By: Alvin Wan

How To Set Up an OpenVPN Server on Debian 8

49

Posted August 10, 2015 369.3k VPN FIREWALL SECURITY DEBIAN

Introduction

OpenVPN is an open source VPN application that lets you create and join a private network securely over the public Internet. In short, this allows the end user to mask connections and more securely navigate an untrusted network.

With that said, this tutorial teaches you how to setup OpenVPN, an open source Secure Socket Layer (SSL) VPN solution, on Debian 8.

Prerequisites

This tutorial assumes you have the following:

- One fresh Debian 8.1 Droplet
- A root user
- Optional: After completion of this tutorial, use a sudo-enabled, non-root account for general maintenance; you can set one up by following steps 2 and 3 of this tutorial

Step 1 — Install OpenVPN

Before installing any packages, update the apt package index.

```
# apt-get update
```

Now, we can install the OpenVPN server along with easy-RSA for encryption.

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

Step 2 — Configure OpenVPN

The example VPN server configuration file needs to be extracted to `/etc/openvpn` so we can incorporate it into our setup. This can be done with one command:

```
# gunzip -c /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz >
```

Once extracted, open the server configuration file using nano or your favorite text editor.

```
# nano /etc/openvpn/server.conf
```

In this file, we will need to make four changes (each will be explained in detail):

1. Secure server with higher-level encryption
2. Forward web traffic to destination
3. Prevent DNS requests from leaking outside the VPN connection
4. Setup permissions

First, we'll double the RSA key length used when generating server and client keys. After the main comment block and several more chunks, search for the line that reads:

/etc/openvpn/server.conf

```
# Diffie hellman parameters.
# Generate your own with:
#   openssl dhparam -out dh1024.pem 1024
# Substitute 2048 for 1024 if you are using
# 2048 bit keys.
dh dh1024.pem
```

Change `dh1024.pem` to `dh2048.pem`, so that the line now reads:

/etc/openvpn/server.conf

```
dh  dh2048.pem
```

Second, we'll make sure to redirect all traffic to the proper location. Still in `server.conf`, scroll past more comment blocks, and look for the following section:

/etc/openvpn/server.conf

```
# If enabled, this directive will configure
# all clients to redirect their default
# network gateway through the VPN, causing
# all IP traffic such as web browsing and
# and DNS lookups to go through the VPN
# (The OpenVPN server machine may need to NAT
# or bridge the TUN/TAP interface to the internet
# in order for this to work properly).
;push "redirect-gateway def1 bypass-dhcp"
```

Uncomment `push "redirect-gateway def1 bypass-dhcp"` so the VPN server passes on clients' web traffic to its destination. It should look like this when done:

/etc/openvpn/server.conf

```
push "redirect-gateway def1 bypass-dhcp"
```

Third, we will tell the server to use OpenDNS for DNS resolution where possible. This can help prevent DNS requests from leaking outside the VPN connection. Immediately after the previously modified block, edit the following:

/etc/openvpn/server.conf

```
# Certain Windows-specific network settings
# can be pushed to clients, such as DNS
# or WINS server addresses.  CAVEAT:
# http://openvpn.net/faq.html#dhcpcaveats
# The addresses below refer to the public
# DNS servers provided by opendns.com.
;push "dhcp-option DNS 208.67.222.222"
;push "dhcp-option DNS 208.67.220.220"
```

Uncomment `push "dhcp-option DNS 208.67.222.222"` and `push "dhcp-option DNS 208.67.220.220"`. It should look like this when done:

/etc/openvpn/server.conf

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

Fourth, we will define permissions in `server.conf`:

/etc/openvpn/server.conf

```
# You can uncomment this out on
# non-Windows systems.
;user nobody
;group nogroup
```

Uncomment both `user nobody` and `group nogroup`. It should look like this when done:

/etc/openvpn/server.conf

```
user nobody
group nogroup
```

By default, OpenVPN runs as the **root** user and thus has full root access to the system. We'll instead confine OpenVPN to the user **nobody** and group **nogroup**. This is an unprivileged user with no default login capabilities, often reserved for running untrusted applications like web-facing servers.

Now save your changes and exit.

Step 3 — Enable Packet Forwarding

In this section, we will tell the server's kernel to forward traffic from client services out to the Internet. Otherwise, the traffic will stop at the server.

Enable packet forwarding during runtime by entering this command:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Next, we'll need to make this permanent so that this setting persists after a server reboot. Open the `sysctl` configuration file using nano or your favorite text editor.

```
# nano /etc/sysctl.conf
```

Near the top of the `sysctl` file, you will see:

/etc/openvpn/server.conf

```
# Uncomment the next line to enable packet forwarding for IPv4
#net.ipv4.ip_forward=1
```

Uncomment `net.ipv4.ip_forward`. It should look like this when done:

/etc/openvpn/server.conf

```
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

Save your changes and exit.

Step 4 — Install and Configure ufw

UFW is a front-end for IPTables. We only need to make a few rules and configuration edits. Then we will switch the firewall on. As a reference for more uses for UFW, see How To Setup a Firewall with UFW on an Ubuntu and Debian Cloud Server.

First, install the `ufw` package.

```
# apt-get install ufw
```

Second, set UFW to allow SSH:

```
# ufw allow ssh
```

This tutorial will use OpenVPN over UDP, so UFW must also allow UDP traffic over port `1194`.

```
# ufw allow 1194/udp
```

The UFW forwarding policy needs to be set as well. We'll do this in the primary configuration file.

```
# nano /etc/default/ufw
```

Look for the following line:

/etc/default/ufw

```
DEFAULT_FORWARD_POLICY="DROP"
```

This must be changed from `DROP` to `ACCEPT`. It should look like this when done:

/etc/default/ufw

```
DEFAULT_FORWARD_POLICY="ACCEPT"
```

Save and exit.

Next we will add additional UFW rules for network address translation and IP masquerading of connected clients.

```
# nano /etc/ufw/before.rules
```

Next, add the area in red for **OPENVPN RULES**:

/etc/ufw/before.rules

```
#
# rules.before
#
# Rules that should be run before the ufw command line added rules. Custom
# rules should be added to one of these chains:
#   ufw-before-input
#   ufw-before-output
#   ufw-before-forward
#

# START OPENVPN RULES
# NAT table rules
*nat
:POSTROUTING ACCEPT [0:0]
# Allow traffic from OpenVPN client to eth0
```

```
-A POSTROUTING -s 10.8.0.0/8 -o eth0 -j MASQUERADE
COMMIT
# END OPENVPN RULES

# Don't delete these required lines, otherwise there will be errors
*filter
```

Save and exit.

With the changes made to UFW, we can now enable it. Enter into the command prompt:

```
# ufw enable
```

Enabling UFW will return the following prompt:

```
Command may disrupt existing ssh connections. Proceed with operation (y|n)?
```

Answer `y`. The result will be this output:

```
Firewall is active and enabled on system startup
```

To check UFW's primary firewall rules:

```
# ufw status
```

The status command should return these entries:

```
Status: active

To                           Action      From
--                           ------      ----
22                           ALLOW       Anywhere
1194/udp                     ALLOW       Anywhere
22 (v6)                      ALLOW       Anywhere (v6)
1194/udp (v6)                ALLOW       Anywhere (v6)
```

Step 5 — Configure and Build the Certificate Authority

OpenVPN uses certificates to encrypt traffic.

In this section, we will setup our own Certificate Authority (CA) in two steps: (1) setup variables and (2) generate the CA.

OpenVPN supports bidirectional authentication based on certificates, meaning that the client must authenticate the server certificate and the server must authenticate the client certificate before mutual trust is established. We will use Easy RSA's scripts to do this.

First copy over the Easy-RSA generation scripts.

```
# cp -r /usr/share/easy-rsa/ /etc/openvpn
```

Then, create a directory to house the key.

```
# mkdir /etc/openvpn/easy-rsa/keys
```

Next, we will set parameters for our certificate. Open the variables file using nano or your favorite text editor.

```
# nano /etc/openvpn/easy-rsa/vars
```

The variables below marked in red should be changed according to your preference.

/etc/openvpn/easy-rsa/vars

```
export KEY_COUNTRY="US"
export KEY_PROVINCE="TX"
export KEY_CITY="Dallas"
export KEY_ORG="My Company Name"
export KEY_EMAIL="sammy@example.com"
export KEY_OU="MYOrganizationalUnit"
```

In the same `vars` file, also edit this one line shown below. For simplicity, we will use `server` as the key name. If you want to use a different name, you would also need to update the OpenVPN configuration files that reference `server.key` and `server.crt`.

Below, in the same file, we will specify the correct certificate. Look for the line, right after the previously modified block that reads

/etc/openvpn/easy-rsa/vars

```
# X509 Subject Field
export KEY_NAME="EasyRSA"
```

Change `KEY_NAME`'s default value of `EasyRSA` to your desired server name. This tutorial will use the name `server`.

/etc/openvpn/easy-rsa/vars

```
# X509 Subject Field
export KEY_NAME="server"
```

Save and exit.

Next, we will generate the Diffie-Helman parameters using a built-in OpenSSL tool called `dhparam`; this may take several minutes.

The `-out` flag specifies where to save the new parameters.

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

Our certificate is now generated, and it's time to generate a key.

First, we will switch into the `easy-rsa` directory.

```
# cd /etc/openvpn/easy-rsa
```

Now, we can begin setting up the CA itself. First, initialize the Public Key Infrastructure (PKI).

Pay attention to the **dot (.)** and **space** in front of `./vars` command. That signifies the current working directory (source).

```
# . ./vars
```

> The following warning will be printed. Do not worry, as the directory specified in the warning is empty. `NOTE: If you run ./clean-all, I will be doing a rm -rf on /etc/openvpn/easy-rsa/keys.`

Next, we'll clear all other keys that may interfere with our installation.

```
# ./clean-all
```

Finally, we will build the CA using an OpenSSL command. This command will prompt you for a confirmation of "Distinguished Name" variables that were entered earlier. Press `ENTER` to accept existing values.

```
# ./build-ca
```

Press `ENTER` to pass through each prompt since you just set their values in the `vars` file.

The Certificate Authority is now setup.

Step 6 — Generate a Certificate and Key for the Server

In this section, we will setup and launch our OpenVPN server.

First, still working from `/etc/openvpn/easy-rsa`, build your key with the server name. This was specified earlier as `KEY_NAME` in your configuration file. The default for this tutorial is `server`.

```
# ./build-key-server server
```

Again, output will ask for confirmation of the Distinguished Name. Hit `ENTER` to accept defined, default values. This time, there will be two additional prompts.

```
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Both should be left blank, so just press ENTER to pass through each one.

Two additional queries at the end require a positive (`y`) response:

```
Sign the certificate? [y/n]
1 out of 1 certificate requests certified, commit? [y/n]
```

You will then be prompted with the following, indicating success.

```
Output
Write out database with 1 new entries
Data Base Updated
```

Step 7 — Move the Server Certificates and Keys

We will now copy the certificate and key to `/etc/openvpn`, as OpenVPN will search in that directory for the server's CA, certificate, and key.

```
# cp /etc/openvpn/easy-rsa/keys/{server.crt,server.key,ca.crt} /etc/openvpn
```

You can verify the copy was successful with:

```
# ls /etc/openvpn
```

You should see the certificate and key files for the server.

At this point, the OpenVPN server is ready to go. Start it and check the status.

```
# service openvpn start
# service openvpn status
```

The status command will return something to the following effect:

```
Output
* openvpn.service - OpenVPN service
```

```
   Loaded: loaded (/lib/systemd/system/openvpn.service; enabled)
   Active: active (exited) since Thu 2015-06-25 02:20:18 EDT; 9s ago
  Process: 2505 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
 Main PID: 2505 (code=exited, status=0/SUCCESS)
```

Most importantly, from the output above, you should find `Active: active (exited) since...` instead of `Active: inactive (dead) since...`.

Your OpenVPN server is now operational. If the status message says the VPN is not running, then take a look at the `/var/log/syslog` file for errors such as:

```
Options error: --key fails with 'server.key': No such file or directory
```

That error indicates `server.key` was not copied to `/etc/openvpn` correctly. Re-copy the file and try again.

Step 8 — Generate Certificates and Keys for Clients

So far we've installed and configured the OpenVPN server, created a Certificate Authority, and created the server's own certificate and key. In this step, we use the server's CA to generate certificates and keys for each client device which will be connecting to the VPN.

Key and Certificate Building

It's ideal for each client connecting to the VPN to have its own unique certificate and key. This is preferable to generating one general certificate and key to use among all client devices.

> **Note:** By default, OpenVPN does not allow simultaneous connections to the server from clients using the same certificate and key. (See `duplicate-cn` in `/etc/openvpn/server.conf`.)

To create separate authentication credentials for each device you intend to connect to the VPN, you should complete this step for each device, but change the name `client1` below to something different such as `client2` or `iphone2`. With separate credentials per device, they can later be deactivated at the server individually, if need be. The remaining examples in this tutorial will use `client1` as our example client device's name.

As we did with the server's key, now we build one for our `client1` example. You should still be working out of `/etc/openvpn/easy-rsa`.

```
# ./build-key client1
```

Once again, you'll be asked to change or confirm the Distinguished Name variables and these two prompts which should be left blank. Press `ENTER` to accept the defaults.

```
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

As before, these two confirmations at the end of the build process require a (`y`) response:

```
Sign the certificate? [y/n]
1 out of 1 certificate requests certified, commit? [y/n]
```

You will then receive the following output, confirming successful key build.

```
Write out database with 1 new entries.
Data Base Updated
```

Then, we'll copy the generated key to the Easy-RSA `keys` directory that we created earlier. Note that we change the extension from `.conf` to `.ovpn`. This is to match convention.

```
# cp /usr/share/doc/openvpn/examples/sample-config-files/client.conf /etc/openvpr
```

You can repeat this section again for each client, replacing `client1` with the appropriate client name throughout.

Note: The name of your duplicated `client.ovpn` doesn't need to be related to the client device. The client-side OpenVPN application will use the filename as an identifier for the VPN connection itself. Instead, you should duplicate `client.ovpn` to whatever you want the VPN's name tag to be in your operating system. For example: **work.ovpn** will be identified as **work**, **school.ovpn** as **school**, etc.

We need to modify each client file to include the IP address of the OpenVPN server so it knows what to connect to. Open `client.ovpn` using nano or your favorite text editor.

```
# nano /etc/openvpn/easy-rsa/keys/client.ovpn
```

First, edit the line starting with `remote`. Change `my-server-1` to `your_server_ip`.

/etc/openvpn/easy-rsa/keys/client.ovpn

```
# The hostname/IP and port of the server.
# You can have multiple remote entries
# to load balance between the servers.
remote your_server_ip 1194
```

Next, find the area shown below and uncomment `user nobody` and `group nogroup`, just like we did in `server.conf` in Step 1. **Note:** This doesn't apply to Windows so you can skip it. It should look like this when done:

/etc/openvpn/easy-rsa/keys/client.ovpn

```
# Downgrade privileges after initialization (non-Windows only)
user nobody
group no group
```

Transferring Certificates and Keys to Client Devices

Recall from the steps above that we created the client certificates and keys, and that they are stored on the OpenVPN server in the `/etc/openvpn/easy-rsa/keys` directory.

For each client we need to transfer the client certificate, key, and profile template files to a folder on our local computer or another client device.

In this example, our `client1` device requires its certificate and key, located on the server in:

- `/etc/openvpn/easy-rsa/keys/client1.crt`
- `/etc/openvpn/easy-rsa/keys/client1.key`

The `ca.crt` and `client.ovpn` files are the same for all clients. Download these two files as well; note that the `ca.crt` file is in a different directory than the others.

- `/etc/openvpn/easy-rsa/keys/client.ovpn`

- `/etc/openvpn/ca.crt`

While the exact applications used to accomplish this transfer will depend on your choice and device's operating system, you want the application to use SFTP (SSH file transfer protocol) or SCP (Secure Copy) on the backend. This will transport your client's VPN authentication files over an encrypted connection.

Here is an example SCP command using our `client1` example. It places the file `client1.key` into the **Downloads** directory on the local computer.

```
# scp root@your-server-ip:/etc/openvpn/easy-rsa/keys/client1.key Downloads/
```

Here are several tools and tutorials for securely transferring files from the server to a local computer:

- WinSCP
- How To Use SFTP to Securely Transfer Files with a Remote Server
- How To Use Filezilla to Transfer and Manage Files Securely on your VPS

At the end of this section, make sure you have these four files on your **client** device:

- \``client1`.crt\`
- \``client1`.key\`
- `client.ovpn`
- `ca.crt`

Step 9 — Creating a Unified OpenVPN Profile for Client Devices

There are several methods for managing the client files but the easiest uses a *unified* profile. This is created by modifying the `client.ovpn` template file to include the server's Certificate Authority, and the client's certificate and its key. Once merged, only the single `client.ovpn` profile needs to be imported into the client's OpenVPN application.

The area given below needs the three lines shown to be commented out so we can instead include the certificate and key directly in the `client.ovpn` file. It should look like this when done:

/etc/openvpn/easy-rsa/keys/client.ovpn

```
# SSL/TLS parms.
# . . .
;ca ca.crt
;cert client.crt
;key client.key
```

Save the changes and exit. We will add the certificates by code.

First, add the Certificate Authority.

```
# echo '<ca>' >> /etc/openvpn/easy-rsa/keys/client.ovpn
# cat /etc/openvpn/ca.crt >> /etc/openvpn/easy-rsa/keys/client.ovpn
# echo '</ca>' >> /etc/openvpn/easy-rsa/keys/client.ovpn
```

Second, add the certificate.

```
# echo '<cert>' >> /etc/openvpn/easy-rsa/keys/client.ovpn
# cat /etc/openvpn/easy-rsa/keys/client1.crt >> /etc/openvpn/easy-rsa/keys/client
# echo '</cert>' >> /etc/openvpn/easy-rsa/keys/client.ovpn
```

Third and finally, add the key.

```
# echo '<key>' >> /etc/openvpn/easy-rsa/keys/client.ovpn
# cat /etc/openvpn/easy-rsa/keys/client1.key >> /etc/openvpn/easy-rsa/keys/client
# echo '</key>' >> /etc/openvpn/easy-rsa/keys/client.ovpn
```

We now have a unified client profile. Using `scp`, you can then copy the `client.ovpn` file to your second system.

Step 10 — Installing the Client Profile

Various platforms have more user-friendly applications to connect to this OpenVPN server. For platform-specific instructions, see Step 5 in this tutorial.

Conclusion

Congratulations! You now have a working OpenVPN server and client file.

From your OpenVPN client, you can test the connection using Google to reveal your public IP. On the client, load it once before starting the OpenVPN connection and once after. The IP address should change.

By: Alvin Wan

Upvote (49) Subscribe Share

Editor:
Tammy Fox

Related Tutorials

How To Create a Point-To-Point VPN with WireGuard on Ubuntu 16.04

How To Secure Web Server Infrastructure With DigitalOcean Cloud Firewalls Using Doctl

How To Troubleshoot DigitalOcean Firewalls

An Introduction To DigitalOcean Cloud Firewalls

How To Create Your First DigitalOcean Cloud Firewall

79 Comments

Leave a comment...

Log In to Comment

deanhere *August 11, 2015*

0 Great Tutorial. Would you add more information about how to config a subnet behind the OpenVPN server? Any additional hardware or server would be needed? Thanks.

alvinwan *August 14, 2015*

0 From OpenVPN:

> If you wish to have particular client-side subnets routed through the VPN, you must ensure that:
>
> - Your Access Server is properly configured so that the User Permissions

> page has the desired client-side subnets specified for the corresponding users.
>
> - The host of each VPN client that is to act as a gateway must be configured to forward traffic to/from the VPN.
> - Your network routing configuration (for any hosts on the VPN that may use the client-side subnets) is adjusted to account for the client-side subnets on the VPN.

If this is all gibberish, keep your eyes peeled for a future DigitalOcean article.

deanhere *August 17, 2015*

0 @alvinwan actually I meant to setup a server-side subnet, not client-side. One client would have access to all remote servers via one OpenVPN server. Looking forward for your future DigitalOcean article~

robbyb *August 11, 2015*

0 Great guide. I just set up mine a few days ago and wish this was out there then.

I have one issue and was hoping for some help or advice. I can't route traffic with just the 1194 port open in ufw. However if I open 1000:2000 the VPN routes the traffic. Any ideas?

alvinwan *August 14, 2015*

1 Thanks. That's interesting. When port 1194 was opened, did the output for `ufw status` actually include port 1194? What errors or exceptions did the client throw, if any?

guestguy *July 7, 2016*

0 Hey Alvin,
Sorry to bring up an old topic but I'm stuck on the second part of step 8. When I put in the command
"scp root@your-server-ip:/etc/openvpn/easy-rsa/keys/client1.key Downloads/"
it says
"ssh: Could not resolve hostname your-server-ip: Name or service not known"
I done everything correctly buy now I cant get the 4 files needed to go to the next step.
What should I do?

adam5385835edbf *July 9, 2016*

0 You need to replace *your-server-ip* with your actual server IP address. You can find this in your DO user dashboard, or use the 'ifconfig' command over SSH. Your IP address will be listed under eth0->inet addr .

robbyb *August 16, 2015*

0 I got this resolved but am unsure what caused the original issue. When I ran ufw status I could see that it had ssh 22, 1194, and 1000: 2000. What I ended up doing was deleting all my ufw rules and starting over this time I just added 22 and 1194 and everything works fine.

supremacy2k *August 28, 2015*

0 Hi.

I just followed this guide to the letter... everything is running server side.
When i try to connect from my android tablet, i get a connection timed out.

Any ideas?

supremacy2k *August 28, 2015*

0 Solved it.

If you change the name of the server, remember to change the name in server.conf file, to point to the right crt and key files.

scottyvortex *August 29, 2015*

0 thanks for the tute. worked a treat :)
I have run into a problem trying to add another client though.

when I run ./build-key client2

I get this error

-bash: ./build-key: No such file or directory

thanks

npiderman *November 15, 2015*

i bet you arent in /etc/openvpn/easy-rsa/

0

FileLife *September 9, 2015*

0 Hi, I hope I can get some help here:

I did everything step by step, connection to the VPN is working. But I don't have a connection to the internet. My Server: IPv4 adress, static. || Client: IPv6 adress, Ipv4 with Dual Stack Lite.

Server: Debian 8
Client: Windows Server 2012 R2

Memonyk *September 12, 2015*

0 I'm new on this field, I follow all steps written here and after I end, I tried to connect to the vpn network from a windows 7 OS, but in the steps for configuring it ask for username and password and I don't know what user and password to put, all I have ar the files user1.crt (what I installed) and user1.key , how to whe this filles to connect in the vpn network

Thank you for the tutorial

---

me1ramcsr *September 19, 2015*

0 Hi all,
I steped all your tutorial. But default gateway is the same 192.168.1.1 (my home router).
My external ip is not changed.
Please give me advice how to resolve this.

Squallk *September 26, 2015*

0 Hey I just found the problem, don't forget you have 4 files right??? So put you client.ovph in the config directory of open vpn AND put the others in the easy-rsa directory of open-vpn!!! And it Works :D

Squallk *September 26, 2015*

0 Same here! I follow the tutorial step by step, all worked well no server problems or little coherence....so I took the 4 files and after change client.ovpn for other devices....I download openvpn and put the config into it....not works....so I decide taking the other cleint.ovpn (the one that we change after) and put in OVPN, it works!!!! Connection bla bla give me new adress 10.0 blabla......and at this point after a full of joy..........my ip DOES NOT CHANGE she stay the same!!! Even if i'm connect to my VPN.......I will try it tomorrow after reinstall entirely my serv, but

if it not works that tuts is broken.

EDIT: I have found the problem!!! If you have the same issues it's because you have put the 4 files together in the config directory of openvpn gui just place your client.ovph in it and put the others in the easy-rsa directory of open gui!!! Good tutorial best I had found on Internet and I'm french!

AndyInMokum *October 18, 2015*

0 Hi, many thanks for the well written and comprehensive tutorial. I'm using a **Raspberry PiB+** with **Raspbian "Jessie"** installed. Installing **openVPN** and setting everything up in the server, (Steps 1 thru' 7) went according to the tutorial. The servers starts without an issue:

```
root@raspberrypi-vpn:/etc/openvpn/easy-rsa# service openvpn start



root@raspberrypi-vpn:/etc/openvpn/easy-rsa# service openvpn status
● openvpn.service - OpenVPN service
   Loaded: loaded (/lib/systemd/system/openvpn.service; enabled)
   Active: active (exited) since Sun 2015-10-18 13:01:19 CEST; 8s ago
  Process: 2682 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
 Main PID: 2682 (code=exited, status=0/SUCCESS)

Oct 18 13:01:19 raspberrypi-vpn systemd[1]: Started OpenVPN service.
root@raspberrypi-vpn:/etc/openvpn/easy-rsa#
```

My issues start while trying to generate **Certificates** and **Keys** for the **"clients"**. I'm changing the name of: **client1** to **Fujitsu_Siemens1**. I'm working from **/etc/openvpn/easy-rsa** as **ROOT**. Here is the output from when I run: **./build-key Fujitsu_Siemens2**.

```
root@raspberrypi-vpn:/etc/openvpn/easy-rsa# ./build-key Fujitsu_Siemens2
  Please edit the vars script to reflect your configuration,
  then source it with "source ./vars".
  Next, to start with a fresh PKI configuration and to delete any
  previous certificates and keys, run "./clean-all".
  Finally, you can run this tool (pkitool) to build certificates/keys.
root@raspberrypi-vpn:/etc/openvpn/easy-rsa#
```

I obviously have to edit something in **/vars**. I'm confused to what this output is asking me to do to continue. Any assistance would be much appreciated.

itf02afc75349ca *February 23, 2016*

0 sh -x ./pkitool

mkturner *October 31, 2015*

0 Even though it isn't really made clear, since I know how `scp` works, I know that the second half of step 8 (the 'Transferring Certificates and Keys to Client Devices' part) should be run from a client computer. However I'm not sure if Step 9 is on the client or the server. I'd assume the server since at this point there is no client.ovpn on the client, only client1.ovpn.

If so, would't this need to be repeated for each client?
Also if this is the 'better' or more-manageable way to handle all the necessary information, I'm curious as to why don't you introduce it in place of where you instruct us to execute the commands to copy the files over scp?

Thanks In advance for any help,
Marvin

wisedjames *November 8, 2015*

0 Hi Alvin, I am experiencing an issue and wonder i you can help me. Your tutorial is ambiguous and lacking in a clearly defined path at the end of step 7. As Step 8 starts the reader is left in the dark. It isn't clear whether we're meant to go back and clear out the existing files created in previous steps, or indeed if we are supposed to refer back to a previous step and repeat it; nor is it clear what we're supposed to do or WHY we're supposed to do it.

"It's ideal for each client connecting to the VPN to have its own unique certificate and key. This is preferable to generating one general certificate and key to use among all client devices..." Well why is this so? This is not explained. If I have 5 devices connecting through my home router, I may want to install the same cert on all devices- OR, I may want to do the (much more sensible) thing, and just install a single certificate on my router's vpn connection, thus bypassing the need to mess around creating multiple keys, and then installing them on individual devices on my network.

Further, there is an annoying 'note' around this stage of the tutorial that begins:

"Note: By default, OpenVPN does not allow simultaneous connections ..."

This only worsens the situation. WHY are you pointing this out? Do I need to edit the config file or not? Again, it is both unclear and confusing. I am up to Step 8, when I try and generate a new cert for client1 I get an error message "Please edit the vars script to reflect your configuration, then source it with "source ./vars"".

Are you implying that, if you yourself follow your own tutorial on a freshly created droplet, that you do not get this message? I can't believe this is so.

Revision is needed. With great power comes great responsibility, as someone said in some dreadful or other... Please revisit this and either
a/ amend the tutorial to better step 8, or
b/ point out the obvious thing(s) that I'm missing :)

I would suggest following your own tutorial with a new droplet. Whenever I have written something like this in the past I have sat down and worked it though as a reader might, and it is an invaluable experience. Namely because it saves you time- you don't have to spend time afterwards answering questions from people like me!

NothingV *November 20, 2015*

0 I don't understand step 9.. so i can't connect to server..

---

bearcat871 *November 21, 2015*

1 I have to agree that Steps 8 & 9 are not very well explained.

I feel like I have followed the tutorial as well as can be expected but after trying to prepare for multiple clients, I cannot get this to work on my first client, a Windows 7 PC. The connection always times out. Something is up and I can only assume it is something to do with steps 8 & 9 which were not very clear, especially if you intend to connect multiple clients.

---

caesar *January 5, 2016*

0 Dear Alvin & Tammy,

thanks for this great tutorial. I was using it on my virtual server (debian 8.2) up to step 4, when the following error occured using the command "ufw enable":

<^>Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
ERROR: problem running ufw-init
modprobe: ERROR: ../libkmod/libkmod.c:508 kmod*lookup*alias*from*builtin*file() could not open builtin file '/lib/modules/3.16.0-042stab113.11/modules.builtin.b in'*
*modprobe: FATAL: Module nf*conntrack*ftp not found.*
*modprobe: ERROR: ../libkmod/libkmod.c:508 kmod*lookup*alias*from*builtin*file() could not open builtin file '/lib/modules/3.16.0-042stab113.11/modules.builtin.b in'
modprobe: FATAL: Module nf*nat*ftp not found.
modprobe: ERROR: ../libkmod/libkmod.c:508 kmod*lookup*alias*from*builtin*file() could not open builtin file '/lib/modules/3.16.0-042stab113.11/modules.builtin.b in'*
*modprobe: FATAL: Module nf*conntrack*netbios*ns not found.
iptables-restore: line 4 failed
iptables-restore v1.4.21: iptables-restore: unable to initialize table 'nat'

```
Error occurred at line: 13
Try `iptables-restore -h' or 'iptables-restore --help' for more information.
iptables-restore: line 34 failed
ip6tables-restore: line 4 failed
ip6tables-restore: line 65 failed
ip6tables-restore: line 34 failed
sysctl: permission denied on key 'net.ipv4.tcp_sack'

Problem running '/etc/ufw/before.rules'
Problem running '/lib/ufw/user.rules'
Problem running '/etc/ufw/before6.rules'
Problem running '/lib/ufw/user6.rules'<^>
```

Couldn't find any solution via google - your help to solve this issue is appreciated.

Thanks & greetings,
caesar

---

x85MKx *January 11, 2016*

0 Hi,
First of all, great tutorial, many thanks for such a nice work.
Second, just wanted to share what problem I had and what solution I found to solve my case. After completing this tutorial, my connection was established, but I had no internet access, after hours of searching I found out, using command - "ifconfig" that my virtual server had venet0 instead of eth0. Just made a change in before.rules and at last everything started to work.

Hope this help :)

---

dimheld *January 13, 2016*

0 Very good tutorial!

BTW!

How it possible to switch on password request for client side ?

---

itf02afc75349ca *January 23, 2016*

0 good howto thx

i have one question
everything work fine but i have some speed problem

client site is router linksys wrtgl with tomato vpn all is connected and i enable Redirect Internet traffic

when router is connected to speed looks like this
4 mb / 1,5 mb

and when i disconnect i have
25 mb /5 mb

where i the problem ?

i change rsa to 1024.pem an try all option on tomato and server
let me know if some one have similar problem

best

---

komasa *February 17, 2016*

0 Has anyone ever actually successfully used the guide?
The openvpn server never actually gets started. For some reason the systemd file for openvpn in debian-8 is just running /bin/true and not the openvpn server. This is even shown here when it tells you to check with "service openvpn status" that it says "active(exited)" (which makes no sense because it should be running, not exited), even worse it says "ExecStart=/bin/true" right there. The writer doesn't seem to be at fault for this, this looks more like some upstream issue, but that means that following this guide it should be impossible to even start the openvpn server.

Efendi *March 30, 2016*

0 Yes, It does work for me and I am not a good linux expert either. I followed step by step. Yes I was confused on step 8 and 9 and also it wasn't clear when client 1 was created for the first time it does create client.ovpn. Then when created unified I dont know what to do with the first client.ovpn but I use the new one for client and it works. The only thing I am struggle with this is to setup client on my Iphone.

Is there the good way how to setup client on Iphone?

---

guestguy *July 7, 2016*

0 Hi Effendi,
I'm like you I'm stuck on the second part of step 8. When I put in the command "scp root@your-server-ip:/etc/openvpn/easy-rsa/keys/client1.key Downloads/"
it says
"ssh: Could not resolve hostname your-server-ip: Name or service not known"
What should I do?

vissie *August 16, 2017*

0 Hi. I just did a install on Debian Jessie. I had the same issues. Seemingly the service started, no visual errors, but openvpn was not running. a "sudo netstat -uapn | grep openvpn" cam up blank. After hours of trouble shooting I eventually came upon this way of starting the service
systemctl start openvpn@server.service.
This failed! Now I had a error.
journalctl -xn showed me a error on TUN/TAP.

Some more hours later I realised, that due to the fact that I was on a VPS, I had no TUN/TAP access from my provider as a default! I had to enable it via a console option.

I did that, the server rebooted. And now I have my openVPN.

So, 10 points to the author on a great setup guide! Thx dude.

I hope this little bit of info helps someone.

Vissie

dobriain *February 18, 2016*

0 What is the solution to the problem already commented for in Step 8 ??
root@www:/etc/openvpn/easy-rsa# ./build-key MAK-CLIENT_1
Please edit the vars script to reflect your configuration,
then source it with "source ./vars".
Next, to start with a fresh PKI configuration and to delete any
previous certificates and keys, run "./clean-all".
Finally, you can run this tool (pkitool) to build certificates/keys.

itf02afc75349ca *February 23, 2016*

0 sh -x ./pkitool

miragetom *July 13, 2016*

0 Yes I am wondering about this as well.
Same problem for me too.

Load More Comments

Community Tutorials Questions Projects Tags Newsletter RSS

Distros & One-Click Apps Terms, Privacy, & Copyright Security Report a Bug Write for DOnations Shop