

Pi-Powered VPN Client Gateway

with Web based Management Interface

Installation and Configuration Guide

v1.0

VPN Client Gateway Management

Basic

Advanced

Tools

Admin

Current VPN server: us-seattle.privateinternetaccess.com



United States (Seattle)

Choose new VPN server:

 <p>Canada</p>	 <p>United States (Seattle)</p>	 <p>United Kingdom (London)</p>
 <p>Sweden</p>	 <p>France</p>	 <p>Germany</p>
 <p>Switzerland</p>	 <p>Netherlands</p>	 <p>Hong Kong</p>

powered by  **OPENVPN**®

Table of Contents

Introduction.....	1
Requirements.....	1
Part 1: Install the operating system.....	2
Prepare the SD card.....	2
Perform initial setup.....	3
Configure network.....	3
Update Raspbian.....	4
Part 2: Install and configure supporting services.....	6
Install the DNS forwarder.....	6
Install openvpn.....	6
Configure logrotate.....	8
Configure iptables.....	8
Enable IP forwarding.....	8
Part 3: Install web server, php, and web page files.....	9
Install the web server.....	9
Change web root folder permissions.....	9
Test web server and php installation.....	10
Add www-data user permissions.....	12
Install VPN gateway management web page files.....	12
Test the VPN gateway.....	13
Appendix A: additional iptables info.....	15
Appendix B: configuring the list of VPN servers.....	16

Pi-Powered VPN Gateway Setup

Introduction

Most VPN service providers offer several servers to connect to around the world. When I first started using a VPN, I configured OpenVPN on my main home network router. To change which VPN server I was connecting to, I'd edit the OpenVPN server setting on the router admin page and restart the service.

To simplify the process of switching servers, I developed a set of web pages and scripts to allow me to change the VPN server just by opening a web page and clicking on a country flag. This solution worked well, but as it was implemented using optware packages on a router running Tomato firmware, it wasn't a very portable solution; anyone who wanted this functionality would have to purchase a router that supports open source firmware. After a recent router upgrade I decided to port the solution to the Raspberry Pi which offers a much better development environment and much lower cost than a router. The new version is easy to set up and has an improved user interface; I can set it up for friends and family who may not have much technical knowledge.

Using this guide and the associated files, you can set up your own VPN client gateway running on a Raspberry Pi.

*Note: to date, this solution has only been tested with the Private Internet Access VPN provider. It could be modified to work with other providers supporting openvpn service, but that is beyond the scope of this guide.

Requirements

- Raspberry Pi B or B+
- Private Internet Access VPN account
- 8GB (suggested minimum) SD card
- Raspbian operating system image

Part 1: Install the operating system

Prepare the SD card

The Raspberry Pi expects a boot loader and operating system on the SD card. In order to use the Pi, you must first install the operating system. For this implementation we'll use Raspbian (a Debian build optimized for the Pi hardware). Download the Raspbian image; links are provided at the following website:

<http://www.raspberrypi.org/downloads>

Now we'll write the disk image to the SD card. There are several tools that can be used to do this, e.g. flashnull and Win32DiskImager for Windows. This guide covers using the Linux **dd** utility. Care must be taken when using **dd** as it will overwrite any data on the target disk. You could potentially wipe out the data on your PC's hard drive. For additional guides on preparing SD cards for the Pi, visit the following web page:

<http://www.raspberrypi.org/documentation/installation/installing-images/>

*Note: the following operations will overwrite any data stored on the SD card.

First, find the disk name of the SD card on your PC:

1. run the fdisk command. Observe the disk names displayed, e.g. /dev/sda, /dev/sdb, /dev/sdc etc.:

```
sudo fdisk -l
```

2. insert the SD card in your PC flash reader/writer, and run the fdisk command again. The SD card will now appear with a disk name, e.g. /dev/sdd
3. Unmount the SD card if needed (it may have been auto-mounted). Example:

```
umount /dev/sdd1 /dev/sdd2
```

Now write the Raspbian image to the SD card:

```
sudo dd bs=4M if=2014-12-24-wheezy-raspbian.img of=/dev/sdd
```

If the write fails, try using a 1M block size. This will take significantly longer:

```
sudo dd bs=1M if=2014-12-24-wheezy-raspbian.img of=/dev/sdd
```

This operation will take several minutes to complete.

Perform initial setup

Now that the SD card has been prepared, we'll boot up the Pi and perform the initial OS configuration:

1. connect the HDMI cable, Ethernet cable, and USB keyboard to the Pi.
2. insert the SD card into the slot on the Pi.
3. connect the 5v power supply.

*Note: the HDMI cable must be connected *before* booting the Pi, otherwise the display will remain blank until the next reboot.

The Pi will boot and then automatically run the Raspbian setup utility. Perform the following actions:

- Expand filesystem
- Change user password
- Enable boot to command line
- Advanced Settings
 - A2 - set hostname
 - A3 - Memory split (leave at default 64)
 - A4 - enable SSH

Select Finish -> Reboot to complete the initial configuration and reboot the Pi.

Configure network

At the login prompt, log in using the username 'pi' and the password you entered in the setup utility. Run ifconfig to view the Pi's IP address:

```
ifconfig eth0
```

You can now log in to your Pi via ssh from your desktop PC, e.g.:

```
ssh -l pi 10.1.2.50
```

Assign a static IP address

Edit the file /etc/network/interfaces:

```
sudo nano /etc/network/interfaces
```

Here is an example `/etc/network/interfaces` with static IP:

```
auto lo

iface lo inet loopback
iface eth0 inet static
address 10.1.2.50
netmask 255.255.255.0
network 10.1.2.0
broadcast 10.1.2.255
gateway 10.1.2.1

allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

Modify the settings to match your network.

Modify DNS settings

Edit the file `/etc/resolv.conf`. Set the nameservers to the loopback address and 8.8.8.8 (google's DNS server). We'll install the `dnsmasq` DNS forward in the next section of this guide.

```
sudo nano /etc/resolv.conf
```

The file should look like this:

```
nameserver 127.0.0.1
nameserver 8.8.8.8
```

Reboot the Pi:

```
sudo reboot
```

After the Pi has rebooted, verify that the static IP address is being used.

*Note: remember to reserve the Pi's IP address in your home network router to prevent conflicts.

Update Raspbian

Run the following commands to update the Raspbian OS and supporting software to the latest release:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

The upgrade operation will take several minutes.

Part 2: Install and configure supporting services

Now that the base OS is installed, we will install and configure the supporting services needed for this project. These services include a DNS forwarder, openvpn, and logrotate.

Install the DNS forwarder

The DNS forwarder will accept DNS requests from clients and forward them to real name servers (e.g. Google's 8.8.8.8). This will prevent DNS leaks when clients are using the Pi as a router/gateway:

```
sudo apt-get install dnsmasq
```

Edit /etc/dnsmasq.conf. Uncomment the domain-needed and bogus-priv settings:

```
sudo nano /etc/dnsmasq.conf
```

Uncomment the domain-needed and bogus-priv settings:

```
# Never forward plain names (without a dot or domain part)
domain-needed
# Never forward addresses in the non-routed address spaces.
bogus-priv
```

Uncomment the 'interface' setting, set it to eth0:

```
# If you want dnsmasq to listen for DHCP and DNS requests only on
# specified interfaces (and the loopback) give the name of the
# interface (eg eth0) here.
# Repeat the line for more than one interface.
interface=eth0
```

Install openvpn

For this project, we'll use openvpn to provide virtual private network connections. This guide assumes the use of the Private Internet Access VPN service.

Install openvpn:

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

Download the openvpn configuration files from the Private Internet Access site:

```
wget https://www.privateinternetaccess.com/openvpn/openvpn.zip
```

Extract the files in the pi user's home folder:

```
unzip openvpn.zip -d /home/pi/openvpn
```


Copy the files Certificate Revocation List and Certificate Authority certificate files to /etc/openvpn:

```
sudo cp openvpn/crl.pem openvpn/ca.crt /etc/openvpn
```

Create the file /etc/openvpn/auth.txt. This file will contain your Private Internet Access login credentials:

```
sudo nano /etc/openvpn/auth.txt
```

Sample contents:

```
p8213552  
svpa214S
```

The first line is your user name, the second line is your password.

Create the file /etc/openvpn/server.conf:

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

The file contents should be as follows:

```
client  
dev tun0  
proto udp  
remote us-west.privateinternetaccess.com 1194  
resolve-retry infinite  
nobind  
persist-key  
persist-tun  
redirect-gateway  
ca ca.crt  
tls-client  
remote-cert-tls server  
auth-user-pass auth.txt  
comp-lzo  
verb 1  
reneg-sec 0  
crl-verify crl.pem
```

Copy this file to a template (the template will be used by the VPN switcher script):

```
sudo cp /etc/openvpn/server.conf /etc/openvpn/server.conf.template
```

Change ownership of the openvpn configuration files:

```
sudo chown www-data:www-data /etc/openvpn/server.conf  
/etc/openvpn/server.conf.template
```

Configure logrotate

To prevent log files from eating up disk space, we'll use logrotate. This service comes pre-installed in the Raspbian build.

Edit the logrotate configuration file:

```
sudo nano /etc/logrotate.conf
```

I would suggest changing the rotate frequency to daily, and the backlogs to 4. Here is what the top section of the file should look like:

```
# see "man logrotate" for details
# rotate log files daily
daily

# keep 4 days worth of backlogs
rotate 4
```

Configure iptables

The firewall used by Raspbian is configured using iptables. It allows or blocks connections based on a set of rules. In order to allow the forwarding, http, and dns connections needed for this project we'll start with a working set of iptables rules. A script to load these rules is included with the files distributed with this document.

Make the script executable:

```
sudo chmod +x ./vpn_client_gw_script.fw
```

Run the script to load the rules:

```
sudo ./vpn_client_gw_script.fw
```

To save these rules so that they reload at boot time, we'll use the iptables-persistent utility.

Install iptables-persistent:

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

During the installation, the program will prompt you to save the current iptables rules (IPv4 and IPv6). Acknowledge both prompts. The iptables rules are now saved and will be re-loaded at boot time.

Refer to Appendix A for further information on the iptables rules.

Enable IP forwarding

```
sudo nano /etc/sysctl.conf
```

Uncomment the following setting:

```
net.ipv4.ip_forward = 1
```

Part 3: Install web server, php, and web page files

In this section of the guide, we'll install and configure the webserver and related services to allow switching between VPN exit points via a web page.

Install the web server

We'll use the lighttpd web server for this project:

```
sudo apt-get install lighttpd
```

In order to run scripts via the web server, we'll use php:

```
sudo apt-get install php5-common php5-cgi php5 php5-curl
```

Enable the php CGI module in lighttpd:

```
sudo lighty-enable-mod fastcgi-php
```

then force lighttpd to reload its configuration:

```
sudo service lighttpd force-reload
```

Change web root folder permissions

We now need to set permissions on the /var/www directory.

Change the directory owner and group:

```
sudo chown www-data:www-data /var/www
```

allow the group to write to the directory:

```
sudo chmod 775 /var/www
```

Add the pi user to the www-data group:

```
sudo usermod -a -G www-data pi
```

Now log out and log back in to pick up the group permissions.

Test web server and php installation

To test the lighttpd and php installation, we'll create a simple php script that displays the php system info:

```
nano /var/www/phpinfo.php
```

The file contents should be as follows:


```
<?php
phpinfo();
?>
```

Now we can try to run this script from a web browser on a PC. To call the script, enter the following url:

```
http://10.1.2.50/phpinfo.php
```


of course changing the IP address to the address of your Pi.

The web page should update and display the php system info, similar to this:

PHP Version 5.4.35-0+deb7u2


System	Linux vpncfisd 3.12.35+ #730 PREEMPT Fri Dec 19 18:31:24 GMT 2014 armv6l
Build Date	Nov 19 2014 12:31:28
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/cgi
Loaded Configuration File	/etc/php5/cgi/php.ini
Scan this dir for additional .ini files	/etc/php5/cgi/conf.d
Additional .ini files parsed	/etc/php5/cgi/conf.d/10-pdo.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525,NTS
PHP Extension Build	API20100525,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, compress.bzip2, php, file, glob, data, http, ftp, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, tls
Registered Stream Filters	zlib.*, bzip2.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk

This program makes use of the Zend Scripting Language Engine:
 Zend Engine v2.4.0, Copyright (c) 1998-2014 Zend Technologies

Powered By


Add www-data user permissions

Some functions require the www-data user to be able to run commands as root.

Edit the file /etc/sudoers.d/sudoers:

```
sudo nano /etc/sudoers.d/sudoers
```

Add the following lines:

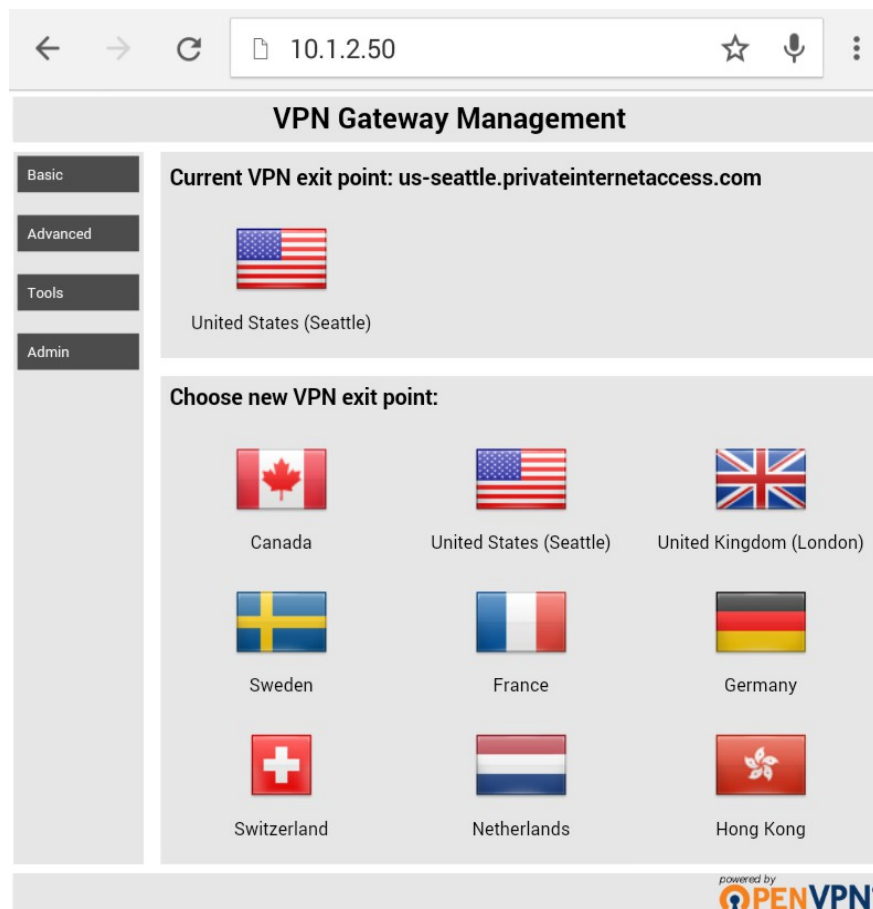
```
www-data ALL=(root) NOPASSWD: /usr/sbin/service
www-data ALL=(root) NOPASSWD: /usr/sbin/update-rc.d
www-data ALL=(root) NOPASSWD: /sbin/shutdown
```

Install VPN gateway management web page files

Extract the vpn_client_gw_www.tar.gz file to /var/www:

```
tar -xvf vpn_client_gw_www.tar.gz -C /var/www
```

That concludes the installation process, you should now be able to access the VPN client gateway management page by accessing the url <http://10.1.2.50> (replacing the IP address with the address of your Pi). You should see the following web page:



Test the VPN gateway

On a client device (tablet or PC) set the gateway and DNS server to the IP address of the Pi, e.g. 10.1.2.50

Open a browser and access the following URL:
<http://ip2location.com>

Review the location information table, in particular the Country and Region/City values. They should show that you are connecting from the U.S.. Here is a sample from my setup:

	Field Name	Value
	IP Address	192.198.202.202
<input checked="" type="checkbox"/>	Country	United States
<input type="checkbox"/>	Region & City	Phoenix, Arizona
<input type="checkbox"/>	Latitude & Longitude	33.44838, -112.07404
<input type="checkbox"/>	ZIP Code	85001
<input type="checkbox"/>	ISP	Secured Servers LLC
<input type="checkbox"/>	Domain	securedservers.com
<input type="checkbox"/>	Time Zone	-07:00
<input type="checkbox"/>	Net Speed	T1
<input type="checkbox"/>	IDD Code & Area Code	+(1) 623
<input type="checkbox"/>	Weather Station	Phoenix (USAZ0166)
<input type="checkbox"/>	MCC, MNC & Carrier Name	-
<input type="checkbox"/>	Elevation	330
<input type="checkbox"/>	Usage Type	DCH

Select a new VPN exit point and then refresh the ip2location.com page. The page should update to display your new IP geo-location.

*Note: it may take several seconds for the vpn client to reconnect to the new location.

Appendix A: additional iptables info

The set of iptables rules supplied with this document were generated using Firewall Builder. If you need to make changes to the firewall rules, you can use this application to do so. The application can be downloaded from www.fwbuilder.org. The Firewall Builder project file is included in the files distributed with this document (vpn_client_gw.fwb). Here is a screenshot taken from Firewall Builder showing how the rules are organized in a grid type view:

VPN Client Gateway / Policy								
	Source	Destination	Service	Interface	Direction	Action	Time	Options
0	Any	VPN Client Gateway	Any	VPN	Inbound	Deny	Any	log
1	lo	Any	Any	lo	Inbound	Deny	Any	log
2	VPN Client Gateway	Any	Any	VPN	Inbound	Deny	Any	log
3	lo	Any	Any	lo	Both	Accept	Any	
4	Any	Any	ESTABLISHED	Any	Both	Accept	Any	
5	VPN Client Gateway	Any	UDP OpenVPN DNS ICMP any ICMP UDP traceroute TCP http	Any	Outbound	Accept	Any	
6	VPN Client Gateway	Any	nfs TCP ssh TCP SMB	eth0	Outbound	Accept	Any	
7	Any	VPN Client Gateway	TCP Transmission Web Interface TCP http DNS ICMP any ICMP TCP ssh	eth0	Inbound	Accept	Any	
8	Any	Any	Any	VPN	Both	Accept	Any	

After making your rule changes, you will need to compile the firewall rules and deploy them to the Pi. Once the rules are updated on the Pi, save the new iptables so that they will be loaded at boot time:

```
sudo su -c 'iptables-save > /etc/iptables/rules.v4'
```

Appendix B: configuring the list of VPN servers

The VPN Gateway Management web page builds the table of available servers dynamically. The list of servers is read from the XML configuration file `/var/www/vpnmgmt/vpnservers.xml`

The section `<basicvpnservers>` contains the list of VPN servers that will be shown on the Basic view of the management web page.

You can change the order in which the servers are displayed by changing their order in this section. You can also remove servers and add additional ones to the basic view. Note that any servers added to this section *must* exist in the `<vpnservers>` section.

As an example, if we change `<basicvpnservers>` to this:

```
<basicvpnservers>
  <servername>us-west.privateinternetaccess.com</servername>
  <servername>us-midwest.privateinternetaccess.com</servername>
  <servername>us-east.privateinternetaccess.com</servername>
  <servername>us-seattle.privateinternetaccess.com</servername>
  <servername>us-california.privateinternetaccess.com</servername>
  <servername>us-texas.privateinternetaccess.com</servername>
</basicvpnservers>
```

the management web page looks like this:

