Setting up a Raspberry Pi web accessible torrent box.

Materials:

Permanent:

Raspberry PI 2 Model B with 32 gB SDHC micro SD card

Micro-USB power

Another computer to access

Internet connection (ethernet)

Temporary:

Monitor with cables/converters

USB Keyboard

USB Mouse

Ethernet cable

Step 1 – Installing OS

Follow the raspberry pi website directions to upload the raspberry os onto the micro sd card. Insert it into the pi. Later on we'll exclusively control the Pi by remote connection, but until that is set up we need to use a standard interaction.

1. Connect the keyboard to a USB port of the pi.
2. Connect the Monitor, depending on monitor type this may be a simple HDMI-HDMI cable, or it might require HDMI-DVI conversion.
3. Insert the micro SD card with an image, raspberry, or NOOBS.
4. Connect power to the Pi
5. When prompted, begin the installation for the Pi. Select the Language down the bottom of the screen (English (US)) and keyboard type (us). Select the default OS (Raspbian). Click Install. Once it completes a dialog “OS(es) Installed Successfully” will appear and prompt restart. Go through setup and use defaults for everything except International Localisation (4), where you can change your timezone and keyboard type. If not using noobs, select option 1 to expand the full sdcard size.
6. Once complete, the command prompt will appear, type

Sudo reboot

1. Now update the Pi to the most recent available version:

Sudo apt-get update

Sudo apt-get upgrade

Sudo apt-get dist-upgrade

1. By default the username and password is pi/raspberry, to change the password type

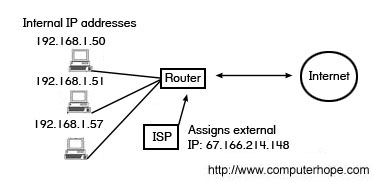
Passwd

1. The pi is pretty good at handling being turned off unexpectedly, but if you want to be nice you can turn it off by typing

Sudo halt

Intermission – Background on network routers/switches

This is just some introduction to those not familiar with how network connection is established. Most homes have one router with many devices connected to it, the router handles all the interactions between the internet and internal devices. Every internal device (phones, computers, tablets, and the router) is assigned an internal IP address – something unique so that it is identifiable, but usually very similar (first 3 digits the same). Likewise, every external device (every router on the internet) is assigned an external IP address – but this is handled through your internet service provider (ISP).



Both internal and external IP addresses can be dynamic (changed once in a while), or static (never changes). Internal IP addresses are configurable by you, you can change if a device is static or dynamic. External IP addresses are configured by your ISP, are usually dynamic by default, and usually costs money to switch it to static.

­­­

Step 2 – Controlling from another computer

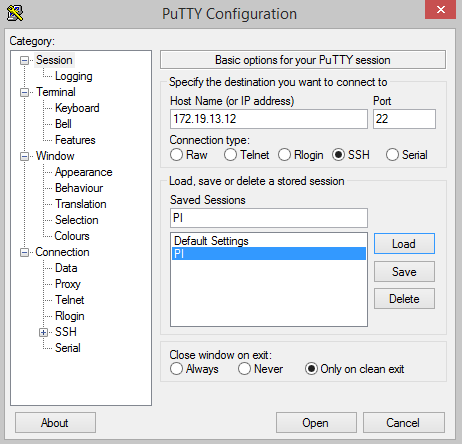
We want control the Pi from another computer so that the Pi is a standalone little box, and allows us to work at our preferred workspace. We will be using SSH (a command-line prompt) and Xrdp (a remote desktop connection). We require that they are on the same network, which will be true if both computers go through the same router.

1. We need to get the IP address of the Pi, which acts as a unique name and location:

ifconfig

The 'inet addr' is the IP address that we are interested in.

1. On the work computer, we need a program to SSH to the Pi, I used PuTTY for Windows.



In the 'Host Name' box, put the Pi's IP address and keep the rest as default, then connect. If successful, a command line terminal will prompt you for a username and password – enter the Pi's (default user: pi, pass: raspberry). You can also save the IP address for later so you don't have to remember it.

Step 3 – Giving the Pi a static IP

REST IS BAD FOR ETHERNET, INSTEAD GO TO ROUTER->CONFIG->ETHERNET, ADD A STATIC IP LEASE LIST, USE A MAC ADDRESS FROM IFCONFIG, USE IPADDRESS OUTSIDE OF DHCP RANGE.

From this point onwards you won't need a monitor, keyboard and mouse attached (except if your internet goes down or something goes wrong with the wifi adapter). If you ever need to reboot type:

sudo reboot

And it should reconnect to the internet automatically after a few seconds.

Background on linux file distribution:

Ls to display current folder

There is a hidden section that can be accessed by doing cd /

Etc: configurations

Hitting tab auto-completes

~

Ls –l: permissions

Step 4 – Accessing files on the network

We'll be using 'samba' to do the work. I followed the guide here: <http://www.howtogeek.com/139433/how-to-turn-a-raspberry-pi-into-a-low-power-network-storage-device/>

1. Install samba

sudo apt-get install samba samba-common-bin

1. Make a directory for you to share

Mkdir PiDrive

1. Create a backup of configurations

sudo cp /etc/samba/smb.conf /etc/samba/smb.conf.old

1. Alter the configuration file:

sudo nano /etc/samba/smb.conf

Scroll down to the 'Authentication' section, and uncomment (remove the '#') for the line:

security = user

Scroll down to the very bottom, we're going to make a new section in the form:

[PiDrive]  
 comment = Raspberry Pi Micro SD  
 path = /home/pi/PiDrive  
 valid users = @users  
 force group = users  
 create mask = 0660  
 directory mask = 0771  
 read only = no

The name in brackets is the name of the folder as it appears on the network share.

Save and quit (Ctrl+X, Y, enter)

1. Restart Samba:

sudo /etc/init.d/samba restart

1. Add a user authentication:

sudo smbpasswd -a pi

OR if you want to create a custom user, create a new one with the following:

sudo useradd NAME -m -G users

sudo passwd PASSWORD

sudo smbpasswd -a NAME

Enter the password when prompted after 'smbpasswd' command.

1. That completes the Pi side of things, now to add the network on your main machine. This will be different depending on your operating system, but in windows you can open 'Computer' then right click and select 'Add network location', 'Choose a custom network location', 'Browse', select 'RASPBERRY PI', then choose what folder you want to link to.

Step 4– Creating a local web server

Now that we can access our Pi from the internal network, lets have it display a web page. We will later make it accessible externally.

1. First make sure everything is up to date with:

sudo apt-get update

sudo apt-get upgrade

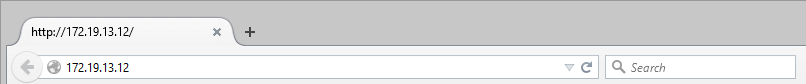
1. We need to install something to do the web server framework for us, which will be Apache, which in turn uses of php5. Let's install it, and everything it needs:

sudo apt-get install apache2 php5 libapache2-mod-php5

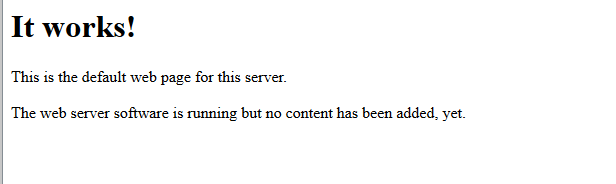
1. Restart Apache:

sudo service apache2 restart

1. This will generate a default web page. We can access it by typing the internal IP address into the URL (top bar) of a web browser.



It should hopefully look like this:



1. The web page can be altered by editing the file found at /var/www/index.html, but we’re going to move that into the samba drive we created earlier.
2. Create a folder for web pages inside the samba drive

Mkdir /PiDrive/webpage

1. Inside the web page, place the script set\_webpage.sh:
2. sudo rm /var/www/\*
3. sudo cp /home/pi/PiDrive/webpage/$1/\* /var/www/
4. sudo chmod 666 /var/www/\*
5. It is important that the file is made from linux, otherwise end of lines cause problems.
6. Then create folders for different website types and versions.
7. Restart the service

Sudo service apache2 restart

1. If you get an error saying something like “could not reliably determine the servers fully qualified domain name”, edit /etc/apache2/apach2.conf and add the line:

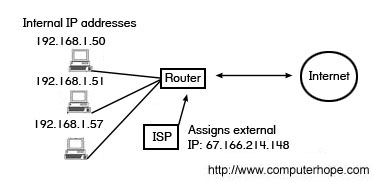
ServerName localhost

Then restart the service.

'sudo' is required as we need admin privileges in order to save over the file. Leave it alone for now, however, as we’ll be doing some extra stuff to make editing it easier.

Intermission – Background on ports

This is some background about ports, and should lead in to the port forwarding we will be doing in the next step. The diagram from earlier may have caused you to ask: How does a device on the internet know how to talk to your internal device?

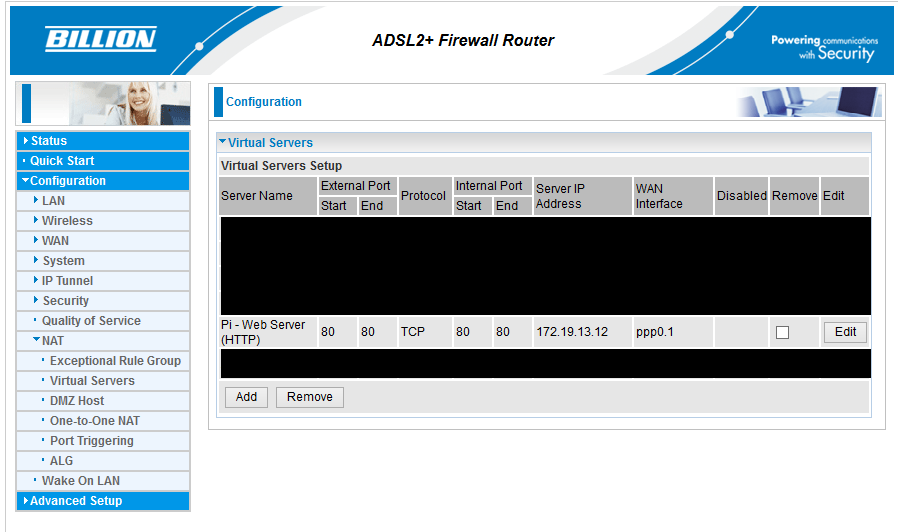


For most home uses, your devices only send requests, e.g device 192.168.1.50 (your computer) wants to load a web page, it sends the request to the router, the router sends the request to a server on the internet somewhere, the server hands a web page back, and the router hands it on to your computer, because it was the one that sent the request. But if its the device on the internet that is sending the request, how does the router know who to give it to? The answer is through ports – which is simply a classification of request types. For example, HTML pages (standard web pages) are typically on ports 80 or 8080, SSH is on port 22, and Call of Duty is on ports 20500, 20510, and 28960. Port forwarding is when we tell the router that all requests with a certain port number should be forwarded to a certain internal IP address, e.g in our router settings, we set port 8080 to forward to 192.168.1.50 (our computer); the next time the router receives an HTML request, it will hand the request over to our computer, and then pass the response back to the device that requested it.

Step 6 – Making the web server externally accessible

We will make the Pi server externally accessible by forwarding port 80. Port forwarding is done by altering the settings in your router, so the procedure will change for different types. I personally have a Billion router, and followed the online instructions for them. Note that a static IP address is necessary for port forwarding.

1. Type your router's IP address (Gateway address from earlier) into the URL of a web browser
2. Enter username and password, then find a way to forward port 80 to the Pi's address (for me it was under Configuration->NAT->Virtual Servers)



1. Find out your external IP address, easiest to do by googling 'my ip address'
2. Type your external IP address into the URL of a web browser. If your page is displayed the port forwarding was successful.

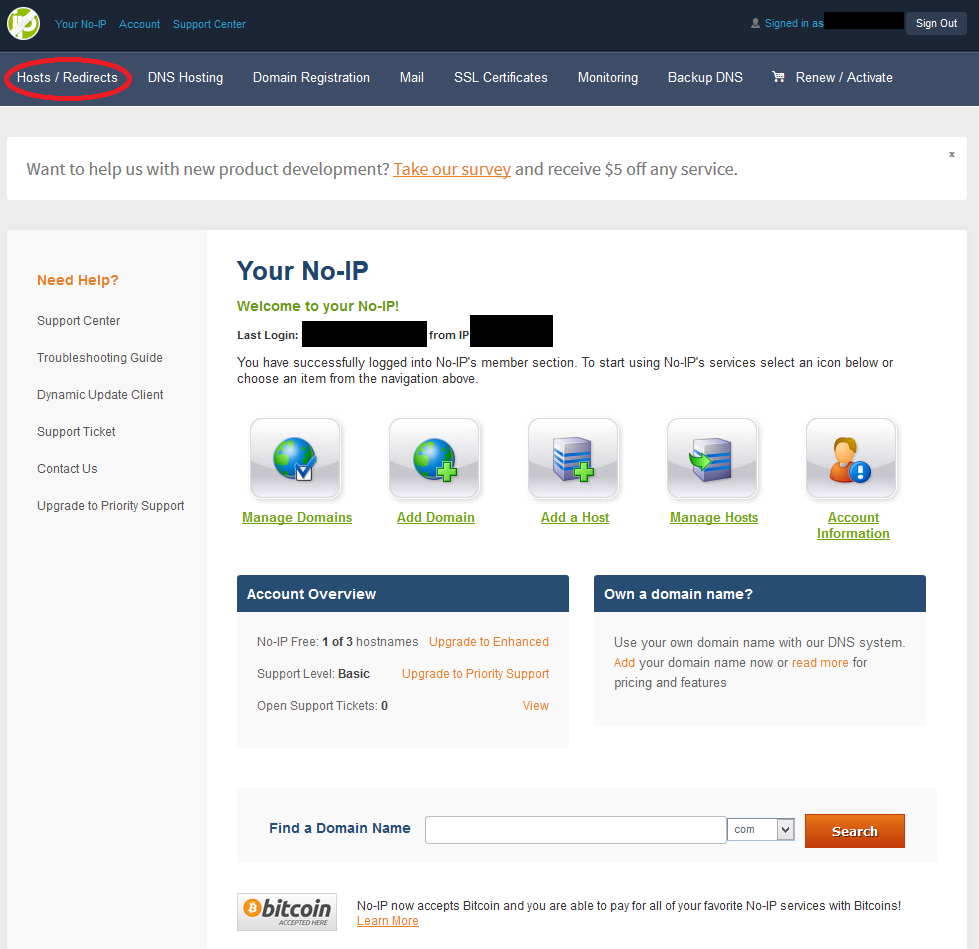
Please note that you should not give out your external IP, it is possible for malicious people to bring down your connection (e.g DDoS). And you also should not be using this method to host heavy traffic – firstly your pi may not be able to handle it, secondly your ISP might think you're hosting a business and come knocking on your door. This method is just intended for personal use.

Intermission – Background on website name servers

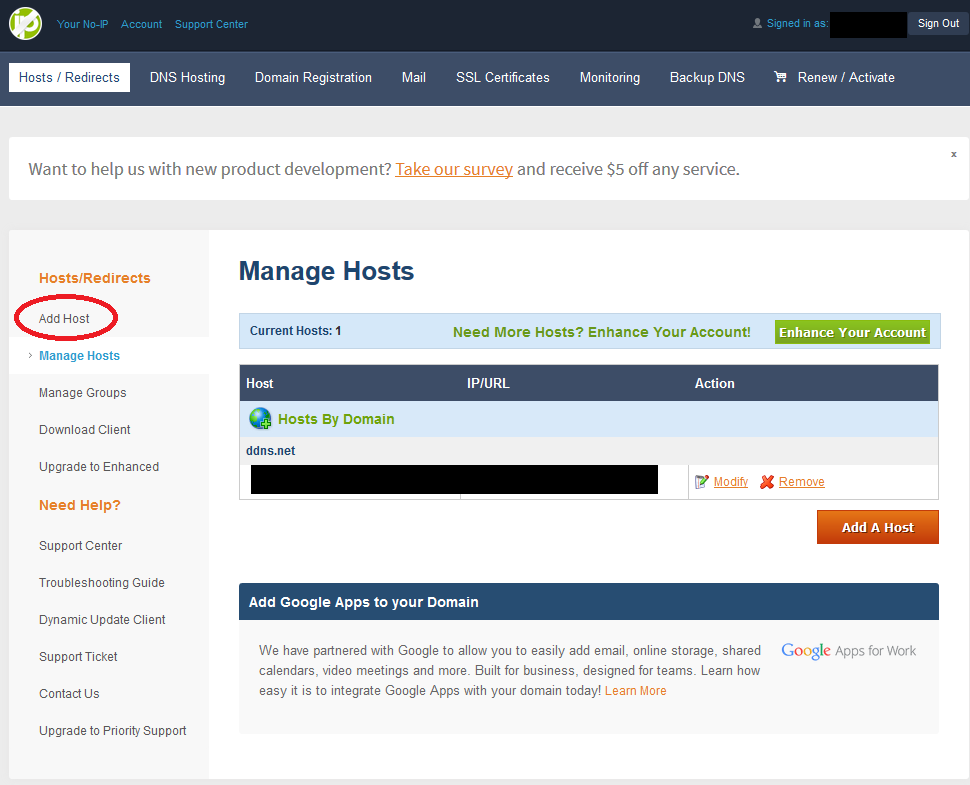
Step 7 – Giving your page a better name

Now that we have an externally accessible web page, we need to give it a name. In this guide, we'll be using noip.com to host a redirect service, and later add dynamic functionality (if you, like me, have a dynamic external address, it will change occasionally – we want to update the redirect service). I recommend looking around to weigh up your options, an amazing service may have been made since this guide was made. At the time of writing, NoIp is free but with limited functionality and a pay upgrade option.

1. Sign up to NoIp: <http://www.noip.com/>
2. At the top bar, select “Hosts / Redirects”



1. On the side, select “Add Host”



1. Enter a desired host name and extension. Select 'DNS Host(A)', enter your IP address, then click “Add Host” at the bottom.
2. Go to the website you assigned, and if successful you will be redirected to your Pi's web page.

Step 8 – Dynamic redirecting (HOLD OFF FOR NOW, AS VPN FUCKS THIS)

As our external IP changes, we need to update the redirect service appropriately. I followed the tutorial found here: <http://raspberrypihelp.net/tutorials/29-raspberry-pi-no-ip-tutorial>. It will prompt you for your username and password, as well as update interval (in minutes) and whether you want something run upon a successful update.

Copy and Pasted:

mkdir /home/pi/noip  
cd /home/pi/noip   
wget http://www.no-ip.com/client/linux/noip-duc-linux.tar.gz

tar vzxf noip-duc-linux.tar.gz

cd noip-2.1.9-1

sudo make  
sudo make install  
sudo /usr/local/bin/noip2

At this point the noip update will not run at startup, so we need to make it do that automatically: <http://www.stuffaboutcode.com/2012/06/raspberry-pi-run-program-at-start-up.html>

The commands are copied below:

sudo nano /etc/init.d/noip

Copy the following into the document:

#! /bin/sh

# /etc/init.d/noip

### BEGIN INIT INFO

# Provides: noip

# Required-Start: $remote\_fs $syslog

# Required-Stop: $remote\_fs $syslog

# Default-Start: 2 3 4 5

# Default-Stop: 0 1 6

# Short-Description: Simple script to start a program at boot

# Description: A simple script from [www.stuffaboutcode.com](http://www.stuffaboutcode.com/) which will start / stop a program a boot / shutdown.

### END INIT INFO  
  
# If you want a command to always run, put it here  
  
# Carry out specific functions when asked to by the system  
case "$1" in

start)  
    echo "Starting noip"  
    # run application you want to start  
    /usr/local/bin/noip2  
    ;;  
  stop)  
    echo "Stopping noip"  
    # kill application you want to stop  
    killall noip2  
    ;;  
  \*)  
    echo "Usage: /etc/init.d/noip {start|stop}"  
    exit 1  
    ;;  
esac  
  
exit 0

Save and close the document with Ctrl-X.

Make the script executable:

sudo chmod 755 /etc/init.d/noip

Start the script to test it:

sudo /etc/init.d/noip start

Stop the script:

sudo /etc/init.d/noip stop

Register script to be run at start-up:

sudo update-rc.d noip defaults

Step 10 – Torrenting

We're going to set up a torrent client that is externally accessible (but password protected), so we can manage and monitor our torrents from anywhere. This is in built in some torrent clients by enabling web-client and accessing our Pi using a particular port (we'll be doing port-forwarding again). I tried two different clients: deluge and transmission. I prefer transmission as it was just easier for me to set up, although there was nothing wrong with deluge. I followed this guide, but with a different folder name: <http://choorucode.com/2014/07/05/how-to-torrent-on-raspbmc-using-transmission/>

1. Make some folders to hold the torrents:

mkdir PiDrive/torrents/completed

mkdir PiDrive/torrents/downloading

'completed' stores the finished files, and 'downloading' stores incomplete files.

1. Install transmission:

sudo apt-get install transmission-daemon

1. Temporarily stop transmission:

sudo service transmission-daemon stop

1. Alter the transmission settings:

sudo nano /etc/transmission-daemon/settings.json

The majority of the settings you see can be changed once we're using the Transmission GUI, but we need to change some now before we get the GUI working. The parameters to edit are:

"download-dir": "/home/pi/PiDrive/torrents/completed",

"incomplete-dir": "/home/pi/PiDrive/torrents/downloading",

"incomplete-dir-enabled": true,

"rpc-password": "raspberry",

"rpc-username": "pi",

"rpc-whitelist-enabled": false,

The 'whitelist' restricts who can access Transmission by IP address, and we want to access it anywhere. You can make the username and password anything, and once we start the daemon the password will be encrypted.

1. Test out transmission by going to your IP address:9091, e.g 172.19.13.69:9091
2. Now download the following: <http://checkmytorrentip.net/torrentip/checkMyTorrentIp.png.torrent>

And add it to the transmission client, this will display your apparent IP address.

1. Now we want to stop this from running at startup:

Sudo service transmission-daemon stop

Sudo update-rc.d transmission-daemon remove

1. From the transmission gui, add the blocklist: <http://ip2k.com/list.gz>

At this point you can torrent something (preferably legal) to see if its working. Later we will add a VPN so that nobody can track what you're downloading. Remember to keep transmission changes, you must restart it.

Step 11 – Making torrenting externally accessible

To access our Pi's transmission client, we have to do the same process we did for the web server, I.e port forwarding. This time, it will be port 9091.

1. Type your router's IP address (Gateway address from earlier) into the URL of a web browser
2. Enter username and password, then find a way to forward port 9091 to the Pi's address (for me it was under Configuration->NAT->Virtual Servers)
3. Find out your external IP address, easiest to do by googling 'my ip address'
4. Type your external IP address into the URL of a web browser followed by ':9091'. If the transmission web-client appears or prompts you for username and password, the redirect was successful.

By default, the NDS redirect (step 7) will already support you putting ':9091' at the end of the URL, so you (hopefully) shouldn't have to do anything more, and you can type:

[http://my-web-server-name.extension:909](http://my-web-server-name.extension:9090/)1

and it should redirect.

Step 12 – Torrenting phone app and browser add-on

Now that we can access our Pi's torrent client from anywhere, we can automatically set it up for our phones and web browsers. The web browser adds an addition right click option “Add to Transmission”, the add-on is suitably called “Add To Transmission”: <https://addons.mozilla.org/en-us/firefox/addon/add-to-transmission/>

In the add-on options set:

URL: <http://WEBSITE-NAME.COM:9091/transmission/rpc>

Username: pi

Password: raspberry

Then if you find a torrent link, just right click it, add to transmission, and the Pi will start downloading it.

As for the phone app, I use Transdrone.

It is simple to set up, just enter the URL, username and password as normal.

Background – IP Tables

<http://ipset.netfilter.org/iptables.man.html>

to see all: sudo iptables –t nat –list

or sudo iptables –nvL –t nat

Step 14 – VPN

A VPN channels ALL internet usage through a server, usually in another country and encrypted. A proxy is similar, but only channels a single program’s internet usage. There are many pros and cons to each, so look it up before deciding what you want. I'll be using a VPN called Private Internet Access, so I'm following the guide here:

<https://bobhood.wordpress.com/2013/07/28/raspberry-pi-creating-a-secure-torrent-client/>

OpenVPN is a free VPN framework that (if the VPN company supports it) allows you to just enter the proper credentials to standardise installation.

1. Install OpenVPN and necessary things.  
    sudo apt-get install openvpn resolvconf
2. Ignore the transmission part of the guide, we've done that already.
3. Create and open a config file:  
    sudo nano /etc/openvpn/pia.conf
4. Copy and paste the text from the guide. For ‘remote’, the ‘nl’ refers to Netherlands, and can be changed to whatever country is supported (Sweden, japan etc).
5. Create the scripts to run upon success or failure of VPN launch (copy the text from the guide into the following files):

sudo nano /etc/openvpn/route-up.sh

sudo nano /etc/openvpn/down.sh

Make the files executable:

sudo chmod +x /etc/openvpn/route-up.sh

sudo chmod +x /etc/openvpn/down.sh

1. Check the files with:

ls -Flavh /etc/openvpn

Executables should be in green.

1. Make the username and password file:

sudo nano /etc/openvpn/userpass.data

Enter PIA username and password

1. IGNORE THIS BIT
2. We’l now be following the guide here: https://dotslashnotes.wordpress.com/2013/08/05/how-to-set-up-a-vpn-private-internet-access-in-raspberry-pi/
3. Get the certificate to use PIA

cd /etc/openvpn

sudo wget <https://www.privateinternetaccess.com/openvpn/openvpn.zip>

sudo unzip openvpn.zip

1. Open a openvpn script to add our password to:

Sudo nano Sweden.ovpn

1. At the bottom add:

Auth-user-pass userpass.data

1. Test that the connection works

Sudo openvpn Sweden.ovpn

1. Note that this is a blocking instruction, so you’ll have to create a new tab (with startx) or a new SSH console to then test if the connection was successful with ‘ipconfig’. If a ‘tun’ segment appears it was successful. You can check your apparent IP address with:

Curl ifconfig.me/ip

1. STOP IGNORING NOW
2. Now to make sure that transmission only runs when we’re connected to the VPN, we’ll be going back to the following guide: <https://bobhood.wordpress.com/2013/07/28/raspberry-pi-creating-a-secure-torrent-client/>
3. Pause the torrent:

Sudo service transmission-daemon stop

1. Remove it from logs and stuff:

Sudo update-rc.d –f transmission-daemon remove

1. Edit the startup and close down scripts from before, and paste in the code from the tutorial.

Sudo nano /etc/openvpn/route-up.sh

Sudo nano /etc/openvpn/down.sh

1. Even though we tell the transmission daemon to close before the tunnel, it might be a bit slow, so we close it earlier:

Sudo nano /etc/openvpn/pia.conf

Add ‘down-pre’ after route-up.sh and before ‘down down-sh’ on its own line.

1. Start up with ‘sudo reboot’, and see if the daemon starts!
2. To stop the vpn and transmission, type:

Sudo service openvpn stop

1. If you change any transmission settings, it must be stopped that way in order to save them.
2. TO SEED WE NEED PORT FORWARDING:
3. Download the script found here:
4. <https://www.privateinternetaccess.com/forum/discussion/3359/port-forwarding-without-application-pia-script-advanced-users>
5. Place it in PiDrive/torrents
6. Add the following after port\_forward\_assignment:

portnum=$json

# Get the isolated portnumber

left=${portnum%%[0-9]\*}

right=${portnum##\*[0-9]}

temp=${portnum#"$left"}

portnum=${temp%"$right"}

transmission-remote -tall --auth pi:ferret -p $portnum

1. And change it so that it has the username and password hardcoded in. Then make it executable:

Chmod +x port\_forward.sh

1. Background on port forwarding – opens up for attacks: protected as long as you do NOT forward the listening port on your router. Watchers might be able to piece together information: port 12345 downloaded torrent1 and torrent2, same person: but this is altered each day.
2. Run script each hour. <https://www.raspberrypi.org/documentation/linux/usage/cron.md>
3. Sudo Crontab –e
4. 0 \* \* \* \* sudo bash /home/pi/PiDrive/torrents/port\_forward.sh
5. Turn on cron logging <http://raspberrypi.stackexchange.com/questions/3741/where-do-cron-error-message-go>
6. Sudo nano /etc/rsyslog.conf
7. Uncomment the line #cron.\* /var/log/cronlog.log
8. Restart rsyslog:

/etc/init.d/rsyslog restart

Step 15 - Excluding HTTP port from VPN

At the moment, web requests to the Pi have their responses filtered through the VPN, so the person who requested it receives the web page from an unknown ip address, and immediately discards it. We need to make it so there are exceptions for certain ports for the VPN.

1. Create a new script

Sudo nano ~/PiDrive/vpnexceptionservice.sh

1. Copy the following contents:

#! /bin/sh

### BEGIN INIT INFO

# Provides: <your script name>

# Required-Start: $all

# Required-Stop:

# Default-Start: 2 3 4 5

# Default-Stop: 0 1 6

# Short-Description: Manage my cool stuff

### END INIT INFO

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/opt/bin

. /lib/init/vars.sh

. /lib/lsb/init-functions

# If you need to source some other scripts, do it here

case "$1" in

start)

log\_begin\_msg "Starting VPN exception on ports 80,443,9091"

iptables -t mangle -A OUTPUT -p tcp -m multiport --sports 80,443,9091 -j MARK --set-mark 0x80

ip route add table 80 to 172.19.13.0/24 dev eth0

ip route add table 80 default via 172.19.13.253

ip rule add fwmark 0x80 lookup 80

log\_end\_msg $?

exit 0

;;

stop)

log\_begin\_msg "Stopping VPN exception on ports 80,443,9091"

# do something to kill the service or cleanup or nothing

iptables -t mangle -D OUTPUT -p tcp -m multiport --sports 80,443,9091 -j MARK --set-mark 0x80

ip route del table 80 to 172.19.13.0/24 dev eth0

ip route del table 80 to default via 172.19.13.253

ip rule del fwmark 0x80 lookup 80

log\_end\_msg $?

exit 0

;;

\*)

echo "Usage: /etc/init.d/<your script> {start|stop}"

exit 1

;;

esac

1. Make the script executable:

Sudo chmod +x vpnexceptionservice.sh

1. Test the script with:

Sudo ./vpnexceptionservice.sh stop

Sudo ./vpnexceptionservice.sh start

1. Make the script boot at startup

Sudo cp vpnexceptionservice.sh /etc/init.d/

Cd /etc/init.d/

Sudo update-rc.d vpnexceptionservice.sh defaults

1. Start and stop the service with:

Sudo service vpnexceptionservice.sh stop

Sudo service vpnexceptionservice.sh start