



Raspberry Pi as an Ad Blocking Access Point

Created by Justin Cooper



Last updated on 2013-09-13 10:30:24 PM EDT

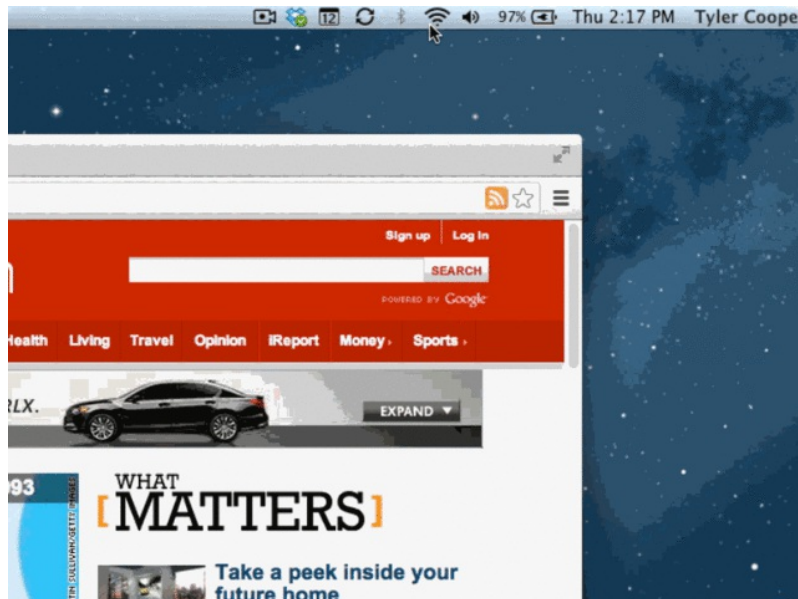
Guide Contents

Guide Contents	2
Overview	3
Preparation	5
Install Software	6
Improving Performance	10
Pixelserv	10
Apache	12
FAQ	15

Overview



This tutorial will show you how to use your Raspberry Pi as a WiFi access point that blocks ads by default for any devices using it. This is really neat in that it would work for your Android or iOS device, your Xbox 360, TiVo, laptop, and more without needing to customize any of those devices other than to use your Raspberry Pi as the access point for WiFi. Using an ad-blocker can be useful for conserving bandwidth, helping out low-power devices, or for keeping your sanity while browsing the web!



I do suggest whitelisting your favorite ad-supported sites so that you can continue to support them (this tutorial shows how to do this as well).

It's not easy keeping up with those wily ad serving companies so we'll also setup a way to keep your ad blocking lists up to date as part of this tutorial.

I used many pages as inspiration in order to set this up, including, but not limited to:

- <http://sfxpt.wordpress.com/2011/02/21/the-best-ad-blocking-method/> (<http://adafru.it/cEG>)
- http://www.debian-administration.org/article/Blocking_ad_servers_with_dnsmasq (<http://adafru.it/cEH>)
- <http://willitscript.com/post/39397777079/raspberry-pi-as-an-ad-blocking-server> (<http://adafru.it/cEI>)
- <http://learn.adafruit.com/setting-up-a-raspberry-pi-as-a-wifi-access-point/overview> (<http://adafru.it/cEJ>)

Each of those links contain useful information, but the next few pages will walk you through in detail each step necessary to successfully block ads with your Raspberry Pi.

Preparation

This tutorial assumes you have your Pi mostly set up and ready to go.

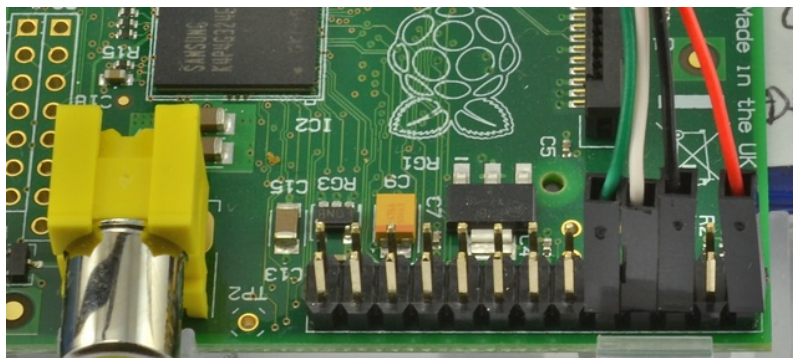
Please follow the tutorials in order to

1. [Install the OS onto your SD card \(http://adafru.it/aWq\)](http://adafru.it/aWq)
2. [Boot the Pi and configure \(http://adafru.it/aUa\)](http://adafru.it/aUa)
Don't forget to change the default password for the 'pi' account!
3. [Set up and test the Ethernet and Wifi connection \(http://adafru.it/aUB\)](http://adafru.it/aUB)
4. [Set up your Raspberry Pi as an Access Point \(http://adafru.it/cg6\)](http://adafru.it/cg6)
5. [Connect with a USB console cable \(optional\) \(http://adafru.it/aUA\)](http://adafru.it/aUA)

When done you should have a Pi that is booting Raspbian, you can connect to with a USB console cable and log into the Pi via the command line interface. Your Pi should also be already setup as a WiFi access point from step 4.

It is possible to do this tutorial via **ssh** on the Ethernet port **or** using a console cable.

If using a console cable, even though the diagram on the last step shows powering the Pi via the USB console cable (red wire) we suggest not connecting the red wire and instead powering from the wall adapter. Keep the black, white and green cables connected as is.



Install Software

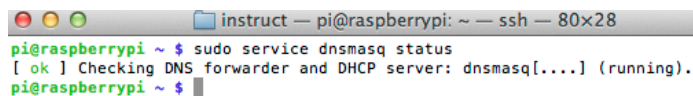
Once you've done all the preparation, setting your Pi up as an ad blocking AP is actually rather simple. The first thing we need to do is install the necessary software, and set it all up.

To start with, uninstall `isc-dhcp-server` and install `dnsmasq`. We'll be using `dnsmasq` as your dhcp server instead of `isc-dhcp-server`. You may want to do a `sudo apt-get update` first, and then execute the following:

```
sudo apt-get autoremove isc-dhcp-server
sudo apt-get install -y dnsmasq dnsutils
```

You can test that it installed properly by checking the status:

```
sudo service dnsmasq status
```



```
pi@raspberrypi ~ $ sudo service dnsmasq status
[ ok ] Checking DNS forwarder and DHCP server: dnsmasq[....] (running).
```

Create and edit a new file for the dhcp server settings:

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

Add the following to that file:

```
interface=wlan0
dhcp-range=wlan0,192.168.42.10,192.168.42.50,2h
# Gateway
dhcp-option=3,192.168.42.1
# DNS
dhcp-option=6,192.168.42.1

dhcp-authoritative
```

Save the file by typing in **Control-X** then **Y** then **return**

These are the minimum settings required to get the dhcp server setup properly. There are a lot more settings available as examples in `/etc/dnsmasq.conf` if you want to configure it further.

Any files added to the directory `/etc/dnsmasq.d` are automatically loaded by `dnsmasq` after a restart. This is a convenient way to override or add new configuration files in Debian.

Next, let's update the dns nameservers to route to our pixelserv IP address first. We'll also go ahead and use the google nameservers.

Open the file with `sudo nano /etc/resolv.conf` and replace the contents with the following:

```
nameserver 192.168.42.49
nameserver 8.8.8.8
nameserver 8.8.4.4
```

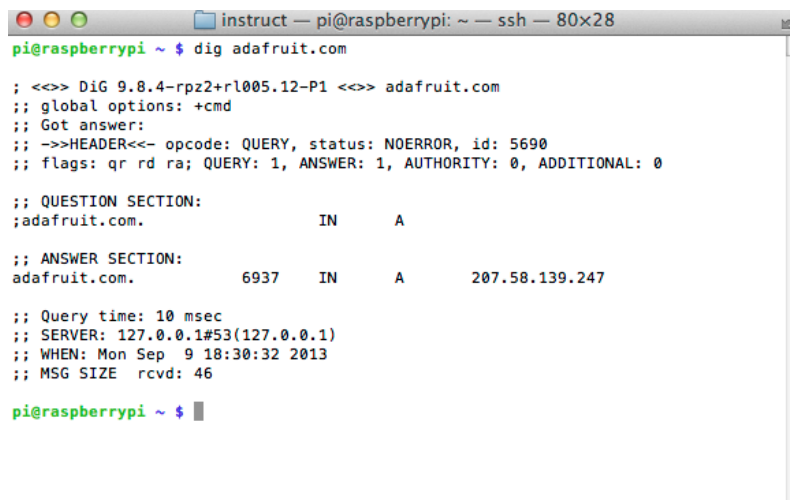
After saving the `resolv.conf` file, let's restart `dnsmasq` to have the settings take effect:

```
sudo service dnsmasq restart
```

Try testing out the dns by using the `dig` command.

```
dig adafruit.com
```

Your results should be similar to the following:



```
pi@raspberrypi ~ $ dig adafruit.com

; <<>> DiG 9.8.4-rpz2+rl005.12-P1 <<>> adafruit.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 5690
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;adafruit.com.                IN      A

;; ANSWER SECTION:
adafruit.com.                6937    IN      A      207.58.139.247

;; Query time: 10 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Sep  9 18:30:32 2013
;; MSG SIZE  rcvd: 46

pi@raspberrypi ~ $
```

Ok, now that we have all of that setup, let's create a script that will redirect known ad servers to an internal IP address. This will essentially cause any requests to generate an HTTP 404.

Open `nano` and create a file with the following command:

```
sudo nano /usr/local/bin/dnsmasq_ad_list.sh
```

Copy and paste the following into the file, and save and exit:

```
#!/bin/bash
```

```

ad_list_url="http://pgl.yoyo.org/adserver/serverlist.php?hostformat=dnsmaq&showintro=0&mim
pixelserv_ip="192.168.42.49"
ad_file="/etc/dnsmaq.d/dnsmaq.adlist.conf"
temp_ad_file="/etc/dnsmaq.d/dnsmaq.adlist.conf.tmp"

curl $ad_list_url | sed "s/127.0.0.1/$pixelserv_ip/" > $temp_ad_file

if [ -f "$temp_ad_file" ]
then
    #sed -i -e '/www\.\favoritesite\.\com/d' $temp_ad_file
    mv $temp_ad_file $ad_file
else
    echo "Error building the ad list, please try again."
    exit
fi

service dnsmasq restart

```

Notice, in the above script there is a line starting with "#sed ". You can uncomment that, and modify it to remove your favorite sites from the ad blocking list so you can continue to support them. You can add as many of those lines as you'd like. One example would be:

```
sed -i -e '/ads\.\stackoverflow\.\com/d' $temp_ad_file
```

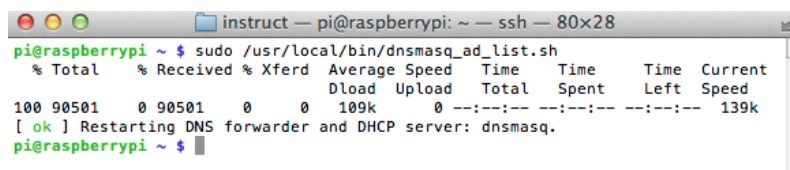
Next allow the file to be executed:

```
sudo chmod +x /usr/local/bin/dnsmasq_ad_list.sh
```

Manually run the script to test that it works:

```
sudo /usr/local/bin/dnsmasq_ad_list.sh
```

You should see the following output:



```

pi@raspberrypi ~ $ sudo /usr/local/bin/dnsmasq_ad_list.sh
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 90501    0 90501    0     0    109k    0 --:--:-- --:--:-- --:--:--   139k
[ ok ] Restarting DNS forwarder and DHCP server: dnsmasq.
pi@raspberrypi ~ $

```

The above script will need to be run as root in order to properly update the configuration file. Basically, what it does it grab a pre-generated list of known ad server domain names from <http://pgl.yoyo.org/adserver> (<http://adafruit.it/cEK>) and save them to a file in /etc/dnsmasq.d/dnsmasq.adlist.conf. It then restarts dnsmasq.

You could manually do this every once in a while, but it's easier if we setup a weekly cron job with the root user.

Open crontab with the following command:

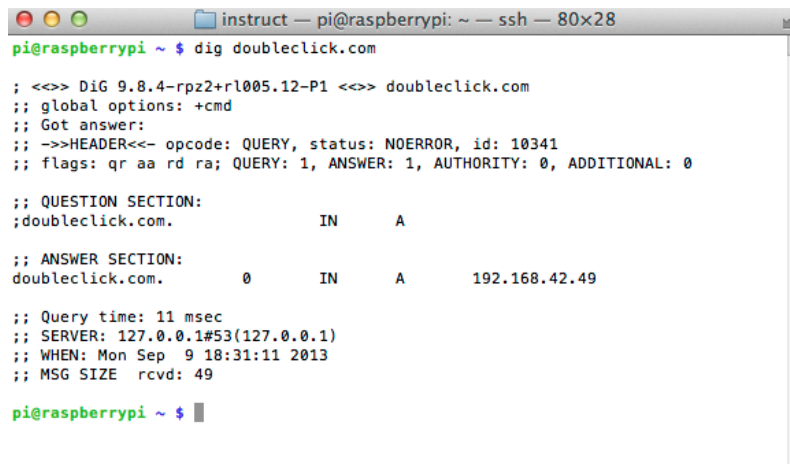

```
sudo crontab -e
```

Add the following line to the end of the file and save it:

```
@weekly /usr/local/bin/dnsmasq_ad_list.sh
```

Congratulations, you should now be blocking ads with your Raspberry Pi. You can test that it's working by executing the following:

```
dig doubleclick.com
```



```
pi@raspberrypi ~ $ dig doubleclick.com

; <<>> DiG 9.8.4-rpz2+rl005.12-P1 <<>> doubleclick.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 10341
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;doubleclick.com.          IN      A

;; ANSWER SECTION:
doubleclick.com.          0       IN      A      192.168.42.49

;; Query time: 11 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Sep  9 18:31:11 2013
;; MSG SIZE  rcvd: 49

pi@raspberrypi ~ $
```

And you should see that it gets routed to the 192.168.42.49 IP address:

One drawback to just stopping here is that we're now timing out on requests that are going to return an HTTP code of 404 for any ad servers. This won't be terribly fast. Another drawback is that any areas with ads will potentially still take up space in the page. We can fix this in the next section.

Improving Performance

One way to improve the performance of the ad blocker is to serve either a 1x1 transparent image, or a blank HTML page. This can be done in a number of ways so we'll show a couple of the options available to you.

Pixelserve

The first, and maybe most common is to use pixelserve. This is a really lightweight perl web server that simply serves a 1x1 transparent GIF to any requests made to it. Thus, anytime ads get redirected by dnsmasq to pixelserve, you'll actually receive a tiny image that won't be visible on the page.

To start with, download the pixelserve file:

```
sudo curl -o /usr/local/bin/pixelserve http://proxytunnel.sourceforge.net/files/pixelserve.pl.txt
```

And also change the permissions:

```
sudo chmod 755 /usr/local/bin/pixelserve
```

Now, open the file with nano:

```
sudo nano /usr/local/bin/pixelserve
```

You can see that the pixelserve is a fairly small perl script. Let's edit it to change the IP address that we're using (192.168.42.49) to redirect the ads to. Find the line with LocalHost, and change it to the following:

```
$sock = new IO::Socket::INET ( LocalHost => '192.168.42.49',
```

Save the file by typing in **Control-X** then **Y** then **return**

You could try running the server now, but you'd get the following error:

```
pi@raspberrypi /usr/local/bin $ ./pixelserve
error : cannot bind : Cannot assign requested address exit
```

We can resolve this by adding this IP address to our wlan0 interface (Thanks to [this site for this fix \(http://adafru.it/cEG\)](http://adafru.it/cEG)!). Open nano and the interfaces file:

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

Update your iface wlan0 inet static section to look like the following. We're adding the last two lines (post-up and pre-down):

```
iface wlan0 inet static
address 192.168.42.1
netmask 255.255.255.0
post-up ip addr add dev wlan0 192.168.42.49/24
pre-down ip addr del dev wlan0 192.168.42.49/24
```

Save the file by typing in **Control-X** then **Y** then **return**

Now reboot your Pi so the settings take effect:

```
sudo reboot
```

Once your system comes back up, try running the server. It won't output anything in the console, but you can try refreshing a page you know that has ads that get blocked:

```
sudo /usr/local/bin/pixelserv
```

It could get annoying having to always run that command. The next logical step would be to create a service that will start the server for us whenever we start our Pi. Let's do that.

First, kill the server you're running by typing "Ctrl-C".

Now, create a new file with nano:

```
sudo nano /etc/init.d/pixelserv
```

Copy and paste the following to that file:

```
### BEGIN INIT INFO
# Provides:      pixelserv
# Required-Start: $network
# Required-Stop: $network
# Default-Start: 2 3 4 5
# Default-Stop:  0 1 6
# Short-Description: pixelserv server for ad blocking
# Description:   Server for serving 1x1 pixels
### END INIT INFO

case "$1" in
start)
echo "pixelserv: starting"
/usr/local/bin/pixelserv &
;;
```

```
stop)
echo "pixelserv: stopping"
killall pixelserv
""
*)
echo "Usage: service $0 {start|stop}"
exit 1
""
esac

exit 0
```

Save the file by typing in **Control-X** then **Y** then **return**

Change the permissions on that file:

```
sudo chmod 744 /etc/init.d/pixelserv
```

Test that the script works by starting and stopping it:

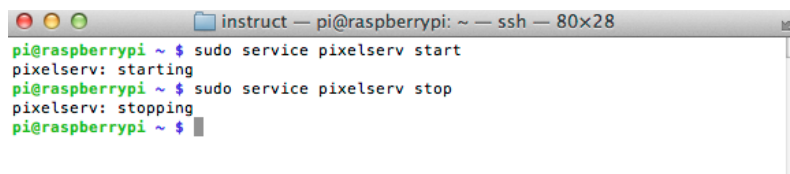
```
sudo /etc/init.d/pixelserv start
sudo /etc/init.d/pixelserv stop
```

Now, enable the script on startup of your Pi:

```
update-rc.d pixelserv defaults
```

You can manually start/stop the pixelserv server by executing the following:

```
sudo service pixelserv start
sudo service pixelserv stop
```



```
instruct — pi@raspberrypi: ~ — ssh — 80x28
pi@raspberrypi ~ $ sudo service pixelserv start
pixelserv: starting
pi@raspberrypi ~ $ sudo service pixelserv stop
pixelserv: stopping
pi@raspberrypi ~ $
```

If you decide you'd rather use the Apache solution, you can disable the pixelserv service on startup:

```
sudo update-rc.d -f pixelserv remove
```

Apache

As an alternative to pixelserv, we can use Apache to serve a blank html file with an HTTP 200 response. Apache is a bit heavier but likely much more stable than pixelserv.

Let's start by installing Apache (make sure to stop and disable pixelserv if you've already installed that):

```
sudo apt-get install apache2 -y
```

By default apache is listening to all IP addresses on port 80. You can change this if you like, but it's not necessary.

Test that apache is picking up our redirected requests:

```
curl doubleclick.com
```

You should get the following:

```
pi@raspberrypi ~ $ curl doubleclick.com
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
pi@raspberrypi ~ $
```

That's a bit more than what we want. Let's modify it so we basically just get an OK from apache instead of any content.

First enable the apache2 rewrite engine by executing the following:

```
sudo a2enmod rewrite
```

Next, let's update the default VirtualHost. Execute the following to open the default VirtualHost file in nano:

```
sudo nano /etc/apache2/sites-available/default
```

Edit the <Directory /var/www/> section to look like the following (we're adding the last three lines):

```
<Directory /var/www/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
    RewriteEngine on
```

```
RedirectMatch 200 (.*)$
ErrorDocument 200 " "
</Directory>
```

Save the file by typing in **Control-X** then **Y** then **return**

Make the same change for the default-ssl file as well (sudo nano /etc/apache2/sites-available/default-ssl).

At this point if you restart Apache, you'll get an error about not being able to determine the server's fully qualified domain name. We can fix that by executing the following command:

```
echo "ServerName raspberrypi" | sudo tee -a /etc/apache2/conf.d/fqdn
```

Ok, now we can restart apache:

```
sudo service apache2 restart
```

And test that the response from Apache has changed:

```
pi@raspberrypi /etc/apache2/sites-available $ curl doubleclick.com
pi@raspberrypi /etc/apache2/sites-available $
```

That's it, you're all setup for super-speedy ad-blocking from your Raspberry Pi!

FAQ

Does this work with SSL traffic?

Yes! This blocks adware and malware sites at the **domain** (DNS) level, so it will work with HTTPS as well as non-SSL pages

Yes! This blocks adware and malware sites at the **domain** (DNS) level, so it will work with HTTPS as well as non-SSL pages

Will this work as a WiFi-to-WiFi bridge? (Or WiFi-to-Cellular)

In theory, if you can get the Access Point setup to work WiFi-to-WiFi then yes the adblocking part will too. *However* we don't have that part of the tutorial written, and most people get best performance with Ethernet-to-WiFi

In theory, if you can get the Access Point setup to work WiFi-to-WiFi then yes the adblocking part will too. *However* we don't have that part of the tutorial written, and most people get best performance with Ethernet-to-WiFi