Building a Raspberry Pi Captive Portal Wi-Fi Hotspot

by ian@cybersecurityguy.com

Introduction

I have a Raspberry Pi that has been lying around for a while, and I decided I would finally do a fun project with it that I could share with my kids to introduce them to a wide range of skills, including Linux, wireless technologies as well as the social aspect of human behavior with technology. This project will take a Raspberry Pi and turn it into what is commonly referred to as a captive portal Wi-Fi hotspot. But first, let's explain what that really means.

As most of us already know, a Wi-Fi hotspot is an open wireless network that you can use to connect to the Internet. Sometimes these are free, and other times they require a form of payment. With a normal Wi-Fi network, you just connect, enter your password, and you're on the Internet. A captive portal is one that blocks you from going straight to the Internet, and instead captures all of your web traffic and redirects you to a web page, where you can agree to their terms and conditions, and sometimes login and/or provide payment. Once you've performed whatever action is required at the portal page, you can then access the Internet for some length of time.

In this project, we will be building a Raspberry Pi to actually do several of these functions. First, we will build it to be a regular Wi-Fi hotspot, and I'll show you where you could use it as your own personal access point. Then I'll show you how you can further modify it to be a captive portal with a single page that provides a list of terms and conditions and an "accept" button. Then I'll show you how you can modify it further to be a fake Wi-Fi public hotspot with a captive portal that actually doesn't connect to the Internet at all. And I'll explain why you might be interested in doing that, for a social experiment.

Project Requirements

Before we begin, let's cover exactly what we'll be using for this project. If you don't use these exact parts, you'll probably be fine, but you will need to make adjustments and find some of your own solutions along the way. You might want to use the same wireless card I specify below, because not all cards can act as Access Point with the appropriate software. So if you are following along and looking to buy parts for this solution, I would strongly encourage you to get this specific card to minimize pain and frustration.

- Raspberry Pi (any model will suffice)
- EDIMax Mini USB Wireless adapter (model EW-7811UN)
- USB keyboard
- HDMI cable for your model of Raspberry Pi
- Monitor with HDMI input

Preparing the Operating System

You might think that the first thing we need to do is to connect all the hardware together, but it's not. The first thing we need to do is to get an operating system installed onto the SD memory card. This is because, without software, there isn't anything we can do with the hardware. Right now we have an SD memory card with nothing on it. We will need to download the operating system onto this memory card so that the Raspberry Pi can load it and function.

There are several different operating systems that we can load onto the Raspberry Pi. They are all based upon a variant of the Linux operating system, whereas you may be normally used to using the Windows or Mac operating systems. The one that we will use in this project is called Raspbian. Rather than re-write what are already great instructions, I'll just refer to them below. Follow the instructions at the link provided below to download NOOBS and install the Raspbian operating system. As part of this, you will install the software you need to write the operating system to the SD memory card, and install it onto the card. Once you've done all of this, and you have successfully written the Raspbian operating system onto the SD memory card, proceed with the rest of my instructions in the next section.

https://www.raspberrypi.org/documentation/installation/installing-images/README.md

Preparing the Hardware

Prepare your hardware as follows:

1. Insert the SD memory card with Raspbian installed into the Pi.
2. Connect the network cable to the Pi. Make sure it will be able to connect to the Internet directly without any special configuration. Technically speaking, the network will need to provide DHCP to the Raspberry Pi. If you don't know what this means, don't worry, you'll almost certainly be fine here.
3. Connect the Wi-Fi USB adapter to the Pi.
4. Connect a USB keyboard to the Pi.
5. Connect an HDMI cable to your monitor and the Pi. Make sure you switch your monitor on and to the correct input.
6. Connect the power to the Pi.

You should see a boot-up sequence or some form of activity. As Raspbian may have changed a bit since this guide was written, just follow whatever questions or prompts you are given until you get to the point where you are provided with a login prompt. That's where we will begin the next section.

Configuring the Pi as a Wi-Fi Hotspot

This is, by far, where most of the meat and complexity lies. We've already taken care of our hardware, so strap in and get ready for the ride. If you're not already familiar with Linux, you're going to be confused with what is going on here. That's OK. I'll explain some of what is happening along the way, just enough so that you have some basic information about what is happening. If I do my job here well enough, hopefully it will get you interested, perhaps even excited, to learn more than what is covered beyond this guide.

Let's begin!

1. You will first need to log in, if you haven't already logged in already. The default login of Raspbian is a username of "pi" and a password of "raspberry". So go ahead and type in the username, press enter, followed by the password and enter. You should get a prompt back that looks somewhat like the following:

```
login as: pi
```

2. Now that we have access to the system, let's change the password. We do that with the "passwd" command. Type that, press enter, and it will ask you for your current password and to enter and confirm your new password. Don't forget your new password, or you will need to reinstall Raspbian onto your SD memory card again and start from the beginning!

```
pi@raspberrypi ~ $ passwd
```

3. Before we go any further, let's confirm that we have a working Internet connection. Let's "ping" Google to make sure we have proper connectivity. This will verify that we have been assigned an IP address, that we are able to resolve DNS hostnames (such as www.google.com, which converts it into an IP address), and that our traffic is flowing all the way out to Google's servers, and that their servers are responding all the way back to us. It's a great way to confirm that everything is working in a single step.

```
pi@raspberrypi ~ $ ping www.google.com

PING www.google.com (74.125.227.24) 56(84) bytes of data.

64 bytes from dfw06s38-in-f17.1e100.net (74.125.227.24): icmp_req=1 ttl=57 time=12 ms
```

4. If everything is working, you'll get a response that shows "icmp_req=" like that shown above, and it will continue to repeat over and over. That's because the PING command will keep sending requests until you interrupt it. You'll need to interrupt it by pressing CTRL-C on your keyboard, after which you'll see a summary of the response, followed by the system prompt.

   If for some reason this doesn't work, check to make sure your network cable is plugged in. If you had a cabling problem, you'll want to reset the power on your Pi and try again. If you still have trouble, you'll probably need to contact someone who understands networking, or try another connection.

As a side note, the numbers shown on the "icmp_req=" line is pretty interesting. The one shown above shows "12.9 ms". That means that it took 12.9 milliseconds to send a network packet to Google's servers, for their server to process it, and for it to get all the way back to the Pi. We call this the "round trip time". That's pretty amazing!

5. Next, make sure all of the software we have installed is up-to-date with all recent patches. We do this by using the "Advanced Packaging Tool", which is used to manage software packages installed on the system. The command-line for this is "apt-get update". This command is very powerful and as such requires administrative, otherwise known as "root" or "super-user" privileges in the Linux world. We can run this, and other powerful commands, by preceding the command with "sudo" (short for "super-user do <something>"). Keep this in mind! Throughout this guide we are going to use the "sudo" command a lot. If you forget to type it, which you probably will at some point, you'll likely get some very weird error message. If you ever get a weird error message you weren't expecting, it's a good idea to double-check that you ran the command as super-user (if it is necessary).

```
pi@raspberrypi ~ $ sudo apt-get update
```

When you run the command below, it will ask you to confirm that you want to download and install all the updated software. You will need to answer that with a "Y" in response.

6. Alright, now we are ready to start installing some other software. Next up is a package called "hostapd". This is used to make a Linux system act as an Wi-Fi "Access Point", or "AP" for short. That means that, instead of the Pi being able to join into a Wi-Fi network, other Wi-Fi devices will be able to see the Pi and be able to connect in to it. We will use the "apt-get" tool again, but this time we're going to tell it to "install" the "host AP" package.

```
pi@raspberrypi ~ $ sudo apt-get install hostapd
```

As another aside, as you start installing more software in Linux, you may notice a lot of packages end in "d", like "hostapd" here. The "d" is short for "daemon", which basically means that it is a service that runs in the background of the system, instead of, for example, an application that displays output on the screen and requires user input from the keyboard. A daemon just runs without needing any of that.

7. We have the hostap daemon installed now, but there is a bug in it that we need to fix. I'll spare you the gory details, but suffice it to say that we need to replace a particular file to get it to work correctly with our wireless adapter. Just go with me on this. First, let's download the file we need. We download the file using the "wget" tool, where the "w" stands for "web".

```
pi@raspberrypi ~ $ wget http://dl.dropbox.com/u/1663660/hostapd/hostapd.zip
```

8. Now that we have this file, we need to "unzip" it, because it is compressed.

```
pi@raspberrypi ~ $ unzip hostapd.zip
```

9. Now we need to copy (“cp”) this file over the original one.

```
pi@raspberrypi ~ $ sudo cp hostapd /usr/sbin/hostapd
```

10. Now that the hostap daemon is installed, we need to configure it for our environment. We will use the “nano” text editor in order to do this. It is not like the text editors you are used to. It looks and feels more complex. So, let’s take this one step at a time. First, let’s launch nano to edit the file.

```
pi@raspberrypi ~ $ sudo nano /etc/hostapd/hostapd.conf
```

Type (or copy and paste) the following lines into the file.

```
interface=wlan0
driver=rtl871xdrv
ssid=FREE PUBLIC WIFI
hw_mode=g
channel=11
macaddr_acl=0
```

Save the file by pressing CTRL-O. It should ask you for the “File Name to Write”. You can accept the default by just pressing enter. Once you do, it should say “[Wrote <#> lines]” near the bottom center of the screen. Exit from nano by pressing CTRL-X. Keep these editing CTRL commands in mind, as we’ll be using them a few more times.

11. Let’s test what we have done so far. Let’s start up hostap and see what happens!

```
pi@raspberrypi ~ $ sudo hostapd /etc/hostapd/hostapd.conf
```

Now check with another wireless device and see if you can find a Wi-Fi network with the name of “FREE PUBLIC WIFI”. If you can, awesome, we’re all good so far. Don’t bother trying to connect to it yet, though. It won’t work properly, because are aren’t done. Press CTRL-C to exit from the hostap daemon. If it didn’t work, or you got any errors along the way, recheck your work between steps 5 and here.

12. As we have now confirmed that we have hostapd working, let’s set it up to always run when the Pi starts up.

```
pi@raspberrypi ~ $ sudo nano /etc/rc.local
```

At the bottom of this file, but above the line that says “exit 0”.

```
hostapd -B /etc/hostapd/hostapd.conf
```

Save and exit the file. When our system reboots the next time, it will automatically run hostapd in the background using the configuration file we customized.

13. Now we will install a DHCP server. DHCP is short for "Dynamic Host Configuration Protocol". In simpler terms, it's the daemon that will provide an IP address, DNS server IP address (we'll talk about that later) and other configuration information to our wireless clients that connect to our AP. Without this information the wireless clients wouldn't be able to communicate with the Pi (or anything else) using the language of the Internet (technically referred to as a network protocol).

```
pi@raspberrypi ~ $ sudo apt-get install isc-dhcp-server
```

14. We need to edit the configuration of the DHCP server. We will use nano again to do this.

```
pi@raspberrypi ~ $ sudo nano /etc/dhcp/dhcpd.conf
```

Find the line that starts with "#authoritative" and remove the "#" in front of it. Then go down to the end of the file, and copy and paste the following text.

```
subnet 10.0.10.0 netmask 255.255.255.0 {
      range 10.0.10.2 10.0.10.254
      option domain-name "schoolscienceproject.com";
      option domain-name-servers 8.8.8.8,8.8.4.4;
      option routers 10.0.10.1;
      interface wlan0;
}
```

Save the file and exit when you are done. You can change your domain name here to whatever you like, just be nice and don't use something that already exists.

15. Now we need to assign the proper IP address to our wireless AP interface. We just finished configuring the wireless client to receive a dynamic IP address from the Pi, but we need to have a fixed (static) IP address that doesn't change for the Pi. It's kind of like having a fixed business address, so our customers know where we are always located.

```
pi@raspberrypi ~ $ sudo nano /etc/network/interfaces
```

Locate the place in the file where it says "iface wlan0 inet manual". Change the "manual" to read "static", and add the following lines immediately below it.

```
address 10.0.10.1
netmask 255.255.255.0
```

Save and exit the file when you are done.

16. We need to enable something called “ip forwarding”. This allows the wireless clients to transfer data not just to the Pi, but through it. Enabling this is what allows traffic to come in one interface (the Wi-Fi) and go out another (the network cable), and vice-versa. We need this in order to get the wireless client data to go through the Pi and out to the Internet and back.

```
pi@raspberrypi ~ $ sudo nano /etc/sysctl.conf
```

    Locate the line in the file where it says “#net.ipv4.ip_forward=1”, and removing the leading “#”. Save the file and exit.

17. Now here is a bit of advanced trickery that is more difficult to wrap one’s head around. It enables something called Network Address Translation, or NAT. This is a topic better left for a later Google search. For now, just type this command.

```
pi@raspberrypi ~ $ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

18. We need to make sure that the change we just make is saved, or persists, between reboots. We need to install another software package in order for this to happen. During installation, you will be asked if you want to save the current rules. Say yes.

```
pi@raspberrypi ~ $ sudo apt-get install iptables-persistent
```

19. At this point, we have installed and configured all of the software we need in order to have the Pi function as a Wi-Fi Hotspot. All we need to do now is reboot it in order to get everything to load. We will use the shutdown command, and tell it to reboot (-r), and to do it immediately (now).

```
pi@raspberrypi ~ $ sudo shutdown –r now
```

20. Wait for the Pi to reboot, and give it a few extra minutes for everything to load properly. Then try connecting with a Wi-Fi device and browse out to the Internet. If it works, congratulations! If it doesn’t, go back and check the steps above.

WARNING!

This configuration was simplified to have the AP be without any encryption. As it is currently configured, anyone will be able to connect to your network and not only browse the Internet, but browse your internal network. If you want to keep using the AP as a personal Wi-Fi hotspot, then you’ll want to configure a password for the network and enable strong security. You can find more information about configuring this in the hostapd.conf file you edited earlier, as well as other online resources.

Configuring the Pi as a Captive Portal

Now that you have your Pi working as a hotspot, let's go further and make it a captive portal. This means that when someone connects, it will present a web page that they must click a button on to continue. They will not be able to access any Internet site until they click the button. This page would normally include a list of terms and conditions for accessing the hotspot, and the button would typically say something like "I accept".

21. Log back in the Pi, and let's install the "nodogsplash" captive portal software. In this series of steps, we will download the software, uncompress, compile and install it.

```
pi@raspberrypi ~ $
          wget https://github.com/nodogsplash/nodogsplash/archive/master.zip
pi@raspberrypi ~ $ unzip master.zip
pi@raspberrypi ~ $ rm master.zip
pi@raspberrypi ~ $ cd nodogsplash-master
pi@raspberrypi ~ $ make
pi@raspberrypi ~ $ sudo make install
```

22. We need to configure nodogsplash by editing the configuration file.

```
pi@raspberrypi ~ $ sudo nano /etc/nodogsplash/nodogsplash.conf
```

Paste the following information into the file, then save and exit.

```
GatewayInterface wlan0
GatewayAddress 10.0.10.1
MaxClients 250
ClientIdleTimeout 480
```

23. Now's the fun part where we can edit the web page to display whatever we want. By default, nodogsplash includes a basic page. Make whatever edits you want, but make sure you retain the part in the file where it lets the user click to join the network.

```
pi@raspberrypi ~ $ sudo nano /etc/nodogsplash/htdocs/splash.html
```

24. Once you've edited your web page, let's start nodogsplash.

```
pi@raspberrypi ~ $ sudo nodogsplash
```

25. Connect to the hotspot, and see if you receive the splash page when you try to browse out to http://www.google.com with a wireless client.

26. Now that we know that nodogsplash is working, we need to set it up to run when the Pi starts. To do that, we need to add it to the /etc/rc.local file.

```
pi@raspberrypi ~ $ sudo nano /etc/rc.local
```

Find the line you added earlier to this file, and add the following line beneath it.

```
nodogsplash
```

27. At this point, your captive portal is working and should continue to work when you reboot! Congratulations!

    If you'd like to keep a log of the devices that connect to your captive portal, including those that click the accept button, you can output status on a regular basis to a log file. We will do this by adding entries to the "crontab", which contains a list of commands that the system will run on a recurring basis. To edit this configuration, use the following command.

```
pi@raspberrypi ~ $ sudo crontab -e
```

Add the following lines to the end of the file. The first line instructs the system to, every 5 minutes, output the status of nodogsplash (using the ndsctl utility) to a file named "/root/results5.txt". A single greater-than symbol would mean that we would overwrite this file every time it runs, but I prefer to append to the file so that we don't accidentally lose any data, because the nodogsplash client status will reset every time the device reboots. I also like to have a backup, so I also have it running every 15 minutes as well, and outputting to a different file name.

```
*/5 * * * * ndsctl status >> /root/results5.txt
*/15 * * * * ndsctl status >> /root/results15.txt
```

Configuring the Pi as a Fake Captive Portal

If you don't have an Internet connection, but you still want your Pi to act as a Wi-Fi captive portal, you can do some more configuration to add the services you need to fake an Internet connection.

28. One of the services that is required for Internet access is DNS, which stands for Domain Name System. Everything on the Internet talks using IP addresses. When you try to access www.google.com, for example, your computer first performs a DNS lookup on this to get back an IP address. Only once it has an IP address does your computer then make a connection to the server, using that IP address. We are going to set up a DNS server on the Pi which responds to every request with the IP address of the Pi itself. So regardless of where the wireless client is trying to connect, it will always direct them back to the IP address of the Pi. We will install "bind", a DNS server .

```
pi@raspberrypi ~ $ sudo apt-get install bind9
```

29. Next we need to configure the DNS server.

```
pi@raspberrypi ~ $ sudo nano /etc/bind/named.conf.local
```

30. Add all of this at the end of the file.

```
zone "." {
        type master;
        file "/etc/bind/db.catchall";
};
```

31. We also need to modify an options file.

```
pi@raspberrypi ~ $ sudo nano /etc/bind/named.conf.options
```

32. Find the line that says "dir=/etc/namedb" and change it to the following.

```
dir = /etc/bind
```

33. And we need to create this file.

```
pi@raspberrypi ~ $ sudo nano /etc/bind/db.catchall
```

34. Add all of this info the file.

```
$TTL    604800
@       IN      SOA     . root.localhost. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL

      IN    NS    .
.     IN    A     10.0.10.1
*.    IN    A     10.0.10.1
```

35. We need to make sure the DNS server starts on boot, so again we need to edit the "/etc/rc.local" file.

```
pi@raspberrypi ~ $ sudo nano /etc/rc.local
```

36. Find the line you added earlier to this file, and add the following line beneath it.

```
named –u bind
```

37. Now we need to reconfigure our DHCP server so that when the wireless clients connect, they are told to use the Pi as their DNS server.

```
pi@raspberrypi ~ $ sudo nano /etc/dhcp/dhcp.conf
```

38. Find the line in the file that says "" and change it to read the following. Recall that the IP address we used for the Pi is 10.0.10.1.

```
option domain-name-servers 10.0.10.1;
```

39. Now disconnect the network cable on the Pi and reboot. Try connecting with a wireless client, and see if you still get the captive portal splash page. If you do, congratulations, you're done!

In Closing

Hopefully this guide has helped you to complete this project successfully. I hope that you've learned something new along the way, and have had fun doing so. If you have any questions or comments, please feel free to drop me a bit at ian@cybersecurity.com.

About Me

I'm an Information Security and Information Technology professional who lives in the Austin, Texas area. I'm a white hat who prefers to create rather than destroy and use my powers for good and not evil. I've attacked and defended companies large and small, with notables including Home Depot, the State of Texas, Amazon, and IBM. I've provided responsible disclosure for multiple high-profile exploits. My area of specialty is broad and ranges from network, server and application security, penetration testing, reverse-engineering and almost everything in between. I've spoken on multiple topics at various security conferences around the country including DerbyCon, BSides Austin/Dallas/San Antonio, HouSecCon and others. I'm passionate about security and technology and learning new things, especially where the two intersect.