# PROJECT IGNORANCE

Comprehensive walkthrough to the BeagleBone Webserver and data collection

## Abstract

A mini-project, designed and executed by a group of first year Ethical Hacking and Cyber Security students at Coventry University.
Our aim was to create a Wi-Fi hotspot using just a Beagle Bone Black and a Wi-Fi dongle to test hypothesis we may stumble across in later years.
This document can be used as a walkthrough to how we made the Wi-Fi network and server.
In essence, this is a light-hearted walkthrough guide.

Harry Smith
Smithh43@uni.coventry.ac.uk

## Contents

## Setting up Wi-Fi hotspot on a Beagle Bone Black

Hello and Welcome to the Wi-Fi hotspot and Captive portal setup Guide for a Beagle Bone Black (BBB). This guide will run through all elements of the setup process, hence, this guide is slightly long. So, grab a coffee, sit back, give it a read and be prepared for any bug fixing (A Beagle Bone is slightly jammy in some situations, especially when connecting to the internet to download Packages)

It is my aim to cover all aspects of the setup process, so if you know how to do certain steps, feel free to skip ahead. However, if you are to use your own values, names, ip addresses, etc, then that is on your own head.

### Equipment

#### Necessary

1) Beagle Bone Black (non-Wireless version)
2) USB cable provided in the Beagle Bone's box to connect BBB to a PC.
3) A Wi-Fi dongle that has drivers compatible with Debian Wheezy.
    a. I have personally used the TP-LINK TL-WN722N 150Mbs
    b. **Warning:** The TP-LINK TL-WN722N now ships with Realtek hardware, it has no driver support for Debian as of 20/03/2018.
4) A third device to test and view webpage.

#### Recommended

1) A Windows Computer to connect the Beagle Bone to the internet easier.
2) Empty MicroSD card to save flashed images of your eMMC memory.

### Connecting Beagle Bone to the internet

*Note: it is **a lot** easier to do this when using a Windows Computer, for this guide I connected to the Beagle Bone via a USB connection to a Windows PC.*

#### Windows

First, we need to download a terminal emulation program. I personally recommend PuTTY. Windows has a handy feature that will allow us to share our Window's device's internet connection with the Beagle Bone.

Download page here: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

To do this, we can do the following:

1) Connect your BBB to laptop with the USB cable
2) Go to Device manager and find the unknown device (RNDIS Card) the one with the yellow exclamation mark.
3) Right click > update driver
4) Click browse my computer for drivers
5) Click "select from a list of available drivers on my computer
6) Find Microsoft in the left column and select it
7) In the right column, press "U" and install the driver called "USB RNDIS Adapter"
8) You should then have the BBB installed as a new network adapter driver

When you have installed the driver, do the following:

1) Go to your network adapter settings (Settings > network > change adapter settings)
2) Right click your WIFI/Ethernet Interface card, NOT the Beagle Bone and go to properties
3) Select the sharing tab and click the first box "allow other users to connect to the internet through this computers internet connection."
4) In the drop-down box select the BeagleBone you connected and installed a driver for (you can also rename the BBB from the properties of the BBB).
5) Go to the BeagleBone's properties (Right click > Properties)
6) Double Click Internet Protocol Version 4 (IPv4) in the scroll box
7) Check the boxes to enable automatic assignment of both IPv4 addresses and DNS servers.
8) Load up PuTTY and you should be able to SSH in when you type in 192.168.7.2 and connect to the internet.

## Linux

First things first we need to connect to the BBB, to be able to do whatever we want on it. We use the following command

```
ssh debian@192.168.7.2
```

The password is temppwd.

This will give us a secure connection to the beaglebone.

Now, need to ensure the BBB can connect to the internet we do this by running two commands.

```
Sudo ping 8.8.8.8
```

if you see packets returning, good! Move on!

**If not do the following:**

Input the following command to add a default gateway to the Beagle Bone (How it connects to the internet.

```
sudo route add default gw 192.168.7.1
```

Now we try to ping through a DNS

```
Sudo ping www.google.com
```

If either of the two ping commands didn't return any packets, we need to configure the DNS server for your BBB. To do this, we need to do a few things:

- Type the following in your Linux Command Line Terminal (Not your BeagleBone)

```
Cat /etc/resolv.conf
```

  - From which you should see something like
    - Nameserver x.x.x.x
  - Note down this IP address for later.

Now go back to your BBB and type the following:

```
Nano /etc/resolv.conf
```

        - Now type above or below the other text – nameserver <ip you noted down>
        - Press ctrl+o to save
        - Then ctrl+y and ctrl+x to exit
- This above, will net the default DNS server of the Beagle Bone to that of your Linux machine.

    - Try the ping command again
        - If you do not get packets coming back type the following to allow Packet forwarding to your Linux Computer.

```
Echo “net.ipv4.ip_forward=1” >> /etc/sysctl.conf
```

        - Then reboot your BBB:

```
Reboot
```

    - Reconnect

```
ssh debian@192.168.7.2
```

    - Try the ping command again
    - You should be connected and receiving packets.

- If you are still not receiving packets, you need to route your traffic from your internet facing interface to your BBB.

  So, on your Linux PC enter the following three commands to route any packets from usbo (your Beagle Bone connection) to wlano (your Wi-Fi card)

```
Sudo iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
```

```
Sudo iptables -A FORWARD -I wlan0 -usb0 -m state –state RELATED,ESTABLISHED -j ACCEPT
```

```
Sudo iptables -A FORWARD -I usbo -o usbo -j ACCEPT
```

  After this, you have successfully routed your internet.

  To finish and save:

```
Iptables save > /etc/iptables/rules.v4
```

## Installing from the repository

After you have got a connection to the internet from your BBB we need to install numerous different files and binaries so we can setup the Hotspot.

To update your packages to their latest versions, type the following:

```
Sudo apt-get update
```

- **If this command fails you are not connected to the internet and need to go through the steps above. (I recommend the Windows steps as it’s a lot easier)**

- Your Wi-Fi Hotspot and Firewall:

```
Sudo apt-get install hostapd iptables iptables-persistent
```

- Your DCHP Server so users can connect and have an IP address

```
Sudo apt-get install dhcp3
```
**OR**
```
Sudo apt-get isc-dhcp-server
```

(If dhcp3 doesn't work try isc-dhcp-server)

- Your web server

```
Sudo apt-get install apache2 php5 php5-mysql php-pear libapache2–mod-php5
```

- **Note, if any of these fail, try installing them one package at a time.**

## Setting up Wi-Fi Hotspot

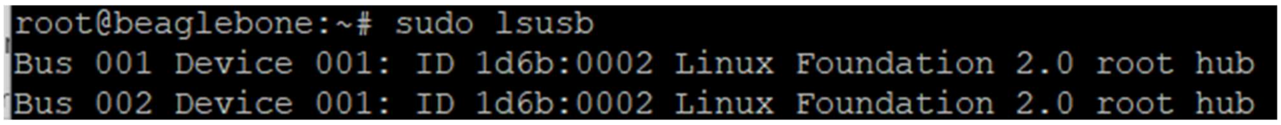
After the packages are installed the first thing to do is setup the Wi-Fi hotspot by editing the network interfaces and hostapd configurations on your BeagleBone.

### Wi-Fi Dongle

First, we need to check that our Wi-Fi Dongle is supported by Debian to do this run the command:

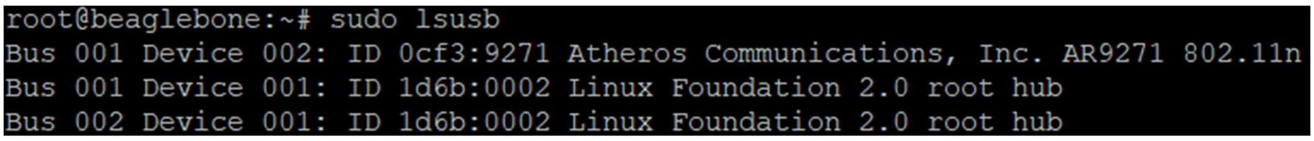
```
Sudo lsusb
```

This will list all known USB ports on the Beagle Bone as well as any device that's connected to them. By default, there are two entries:



```
root@beaglebone:~# sudo lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

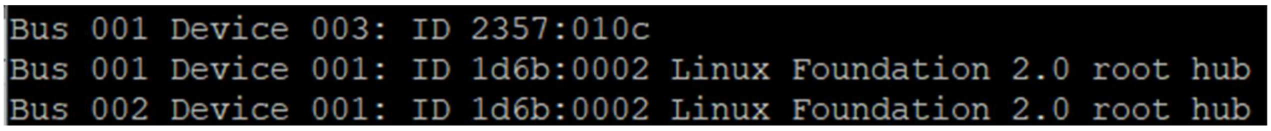
Now, if we connect our dongle we get the following:



```
root@beaglebone:~# sudo lsusb
Bus 001 Device 002: ID 0cf3:9271 Atheros Communications, Inc. AR9271 802.11n
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

This means that the Beagle Bone has successfully identified our Wi-Fi dongle and that it uses the 80211n driver.

This is what it looks like if your dongle is not compatible with Debian:



```
Bus 001 Device 003: ID 2357:010c
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

If this case is yourself, you will need to find a different Wi-Fi dongle, or search around for a driver for your specific dongle that's been compiled by some lovely kind person out there.

In my case, both dongles are in fact the TP-Link TL-WN722N 150mbs, however, as stated before the Realtek hardware in the newly bought dongle does not have a compatible driver. (Boo TP-Link)

## Network Interfaces

A network interface is essentially the way the device can send or receive data from a certain network. We can configure our own interfaces to serve different needs, as seen below.

Even though we have connected the dongle to a USB port, due to Debian having a compatible driver, we can call the dongle wlan0, instead of usb1 (usb0 is the PC -> Beagle Bone connection)

Once you have a dongle that is identified by the Debian OS, we need to setup our interface to serve our Wi-Fi dongle. We can edit the file we need by typing:

```
sudo nano /etc/networking/interfaces
```

Next scroll down to the commented line **Wi-Fi Example.** Uncomment it and edit. We edited ours to look like this:

```
Auto wlan0
Iface wlan0 inet static
  Address 192.168.9.1
  Network 192.168.9.0
  Netmask 255.255.255.0
  Broadcast 192.168.9.255
# hostapd /etc/hostapd/hostapd.conf
```

This means that we have now enabled our wlan0 interface and given it an IP address of 192.168.9.1 and network of 192.168.9.0. So, anyone connecting to our network will connect to and have an address: 192.168.9.xxx (Static or DHCP IP allocation permitting).

Enter the coloured text above (or your own text), then press ctrl+o to save, then ctrl+y and ctrl+x to exit.

## Hostapd

Hostapd is the package used to get the Wi-Fi dongle to appear as a WI-FI network. We must edit a few files here to correctly configure the hotspot.

We used the following as a basis for our Hotspot setup: https://learn.adafruit.com/setting-up-a-raspberry-pi-as-a-wifi-access-point/install-software#configure-access-point

We first must edit a file called /etc/hostapd/hostapd.conf. You can access this by typing:

```
Sudo nano /etc/hostapd/hostapd.conf
```

Find some empty space and input the coloured text below (or your own text).

| | |
|---|---|
| `interface=wlan0` | Interface the dongle is serving |
| `driver=nl80211` | The driver specified above (Standard driver for Access Points AFAIK) |
| `bridge=br0` | Bridge of the Wi-Fi network (where it will connect to) |
| `ctrl_interface=/var/run/hostapd` | The interface which runs the Wi-Fi network |
| `ctrl_interface_group=0` | Group of the interface |
| `ssid=CovFreeWifi` | name of Wi-Fi network |
| `channel=7` | what Wi-Fi channel the dongle operates on (typically 5-14). |
| `hw_mode=g` | The network mode |
| `wpa=2` | type of Wi-Fi security |
| `wpa_passphrase=Password` | Password to login to Wi-Fi network |
| `wpa_key_mgmt=WPA-PSK` | How the network manages logins (WPA – Public Secure Key) |
| `wpa_pairwise=TKIP` | Temporal key Integrity protocol (Encryption Algorithm) |
| `auth_algs=1` | 1 authorisation algorithm |
| `macaddr_acl=0` | Allows all MAC addresses |
| `ignore_broadcast_ssid=0` | Let's the SSID be seen by devices |
| `eapol_key_index_workaround=0` | Needed for Windows PCs so they can connect |
| `beacon_int=100` | How often does a signal get sent (in m/s) |

Save and exit the file as before, ctrl+o, ctrl+y, ctrl+x.

Next, we need to edit the file which tells the BBB where the hostapd config file is

```
nano /etc/default/hostapd
```

Find the line: **DAEMON_CONF=""** edit it so it says
**DEAMON_CONF="/etc/hostapd/hostapd.conf"**

This enables the hostapd config file we wrote to be executed by Hostapd.

Save and exit the file as before, ctrl+o, ctrl+y, ctrl+x.

After this, you can restart hostapd by typing

```
/etc/init.d/hostapd restart
```

If done right, you'll get a little message with a green "ok" and your new Wi-Fi network should be discoverable! Yay!

However, the connected user has no IP address. So, we now have to configure the DHCP Server.

*Note: Alternatively, if you know what devices are to be connecting, you can manually assign them static IPv4 addresses, how to do this is not covered by this guide.*

## Configuring a DHCP server

For the dynamic allocation of user's IP Addresses, we are using isc-dhcp-server, as we installed at the start. For this, we referenced https://fleshandmachines.wordpress.com/2012/10/04/wifi-acces-point-on-beaglebone-with-dhcp/ but added some of our own configurations.

First, we need to edit the configuration file.

```
Sudo nano /etc/dhcp/dhcpd.conf
```

You'll open a file full of commented out lines which run through different configurations for your device.

Go to the bottom and you can add your own, we did the following:

```
subnet 192.168.9.0 netmask 255.255.255.0 {
  Range 192.168.9.10 192.168.9.50;
  Option broadcast 192.168.9.255;
  Option routers 192.168.9.1;
  Default-lease-time 7200;
  Max-lease-time 7200;
  Option domain-name “local”;
  Option domain-name-servers 8.8.8.8, 8.8.4.4;
}
```

This lets us setup a DHCP server, it will assign each new connection to the network an ip address between 192.168.9.10 and 192.168.9.50. We've also made it so the default gateway (routers) is our default gateway to our BBB (192.168.9.1) as well as assigning DNS servers in case the devices connecting do not have one.

Save an exit by pressing the same combination of keys as before: ctrl+o, ctrl+y ctrl+x.

## Iptables

The last step in this setup is to establish routing rules for the network so that the Interface (wlan0) can have an internet connection.

If you have connected your BBB via USB like we have, we are going to route the traffic from wlan0 and through usb0 to get to the internet.

Enter the following commands to do this:

```
Sudo iptables -t nat -A POSTROUTING -o usb0 -j MASQUERADE
```

```
Sudo iptables -A FORWARD -I usb0 -wlan0 -m state –state RELATED,ESTABLISHED -j ACCEPT
```

```
Sudo iptables -A FORWARD -I wlan0 -o usb0 -j ACCEPT
```

Followed by this command to save the configuration to the BeagleBone.

```
Sudo Iptables save > /etc/iptables/rules.v4
```

## Final setup

To finish off the Wi-Fi setup of our BBB and ensure it starts up after a reboot properly, I created a bash script that will run all necessary commands for us at start up, so we don't have to.

You can create this file anywhere, ours is called DefualtGWSetup and is as follows:

```
Nano DefaultGWSetup
```

```
Sudo /sbin/route add default gw 192.168.7.1
Sudo /sbin/init.d/hostapd restart
Sudo /sbin/init.d/isc-dhcp-server restart
```

This small program, when executed, will add the gateway we added before, so our BBB will have internet connection. It will also restart both the Wi-Fi hotspot and DHCP server so it is running fresh.

To have this program run at start-up we run the following command:

```
Sudo crontab -e
```

Once we are in the file we add our file path and file to the end of the document.

```
/<filepath>/<filepath>/DefaultGWSetup
```

Save the file, ctrl+o, ctrl+y, ctrl+x and you are finished!

## Rebooting and Connecting

Now you can safely shutdown your BBB with the command:

```
Shutdown -h now
```

Now with your Dongle plugged in, power on the device and wait for a light to appear on the dongle.

After which you can SSH in and run the file we made above using the command:

```
./DefaultGWSetup
```

(We do this so we get a fresh restart of the hotspot, haven't got to do it every time)

If all goes well, you should have a fully functioning Wi-Fi hotspot.

## Setting up an Apache2 Webserver.

Now that we have a Wi-Fi network setup that allows users to connect and browse the internet from it, we need to set up the webserver so we can get users who connect, to view our own content.

To set up and configure the webpages we will be showing the user, we will be creating a LAMP (Linux, Apache, MySql, Php) server.

## MySQL

To set up an apache server we need to install and securely setup a MySQL server. This will form the backbone of our server and be used to save user data. If you have installed the package the start, skip the first command.

First, we need to install the MySQL-server package. To do this we type the following:

```
Sudo apt-get install mysql-server
```

Next, we have to securely setup the MySQL server. I did this by entering the command:

```
Mysql_secure_installation
```

Say Yes to every question and MySQL should restart and you are good to go!

## Apache

Now, because the Beagle bone already has a server configured. (you can see this if you type in 192.168.7.2 in a web browser with the Beagle Bone connected)

We need to disable this web host so we can display our own webpages. To do this, we need to remove some packages and directories (folders).

```
Sudo apt-get remove npm
Sudo apt-get remove node*
```

This will remove the node.js server the BBB had running previously. Next, we must remove two directories:

```
Sudo Rm -rf /etc/systemd/system/multi-user.target.wants/bonescript
```

```
sudo rm -rf /var/lib/cloud9
```

After this, we should be good to go

```
Service apache2 restart
```

And now we have set up a successful webserver!

You can test this by adding a webpage called index.html or index.php (if you wish to test PHP) in the /var/www directory (folder).

The /var/www directory is the default place Apache accesses its data from. You can change this yourself but for purpose of this guide and simplicity, I did not.

Connect to your network through the Wi-Fi hotspot and navigate to your browser and type the Wi-Fi network's IP address (the one you entered when configuring the hostapd file) followed by: 8080/index.html or: 8080/index.php.

If you do not get the webpage appearing, check the code of your HTML or PHP files. Here are two files you can use if you are stuck

For HTML

```
<html>
<head></head>
<body><h1> Hello world! </h1> <body
</html>
```

For PHP

```
<?php
Phpinfo():
?>
```

## Creating the Database

You should have set up the MySQL database at the beginning of this guide, when you installed and ran Apache2. We are now going to use it again to create a database to deal with user generated data.

Remember, the examples I used here are just that, examples, if you wish to use anything else in place of the Database and Table names, feel free!

I used the following command to start MySQL as root:

```
Mysql -h localhost -u root
```

From here, we input the following commands to create a database and table inside of said database:3

This creates our database called "CovFreeWifi". **(Call your database and table whatever you wish)**

```
CREATE DATABASE CovFreeWifi;
```

While, the next command will create a table with four columns (Fields). Two of which are integers and two are "varchars" or strings.

```
CREATE TABLE WifiData (DB_PrimKey_int, DB_Email varchar(255),
DB_MACAddress varchar (255), DB_LoginCount int,);
```

Followed by this command to modify the column called DB_PrimKey to increment by +1 each time a new record is added. This is our primary key, or unique identifier.

```
ALTER TABLE WifiData MODIFY COLUMN DB_PrimKey int auto_increment;
```

You can now see your table if you enter the command:

```
DESCRIBE * <TableName>;
```

If you are not interested in what I created for the webpages on the server, please click here to skip to the conclusion. Thank you for reading!

## HTML and CSS programming (index.html)

I have now successfully configured an Apache Server and Wi-Fi hotspot on a BeagleBone. The next step would be to start creating content to show when the user connects to our network. Due to this project being about creating a captive portal, we will be doing just that. With three pages that a user is forced to open when they connect to our network.

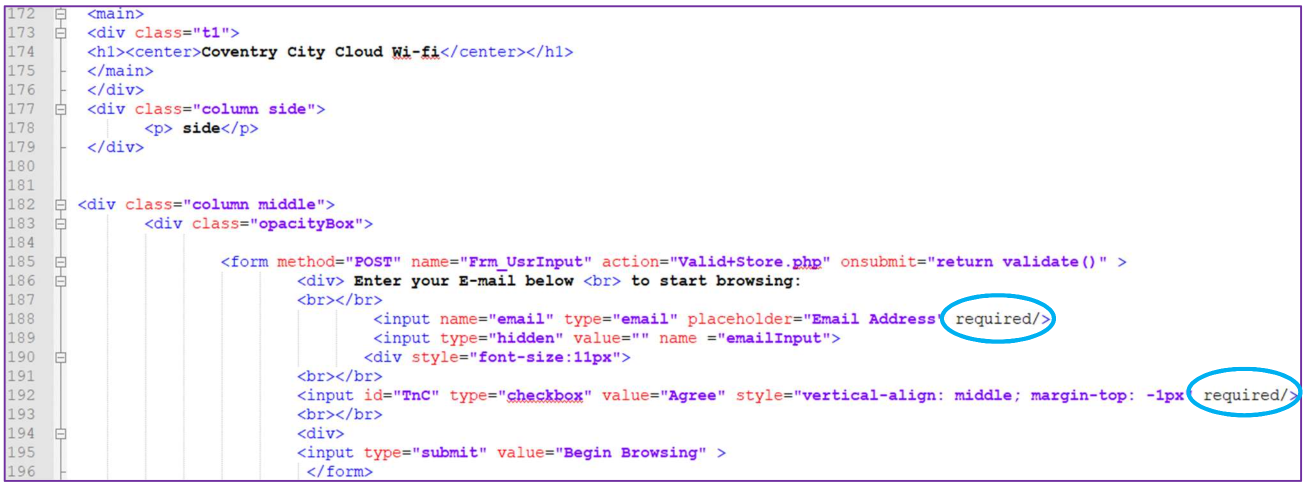
Our first task is to create the Splash Screen/index page, that the user will first be presented with when connecting.

**Note: Your Setup, layout and purpose maybe different to mine, hence change these parts of the guide however you wish! If you wish to use my full or parts of code visit my GitHub:** https://github.com/HarrySmith2/Open-Ended-Project

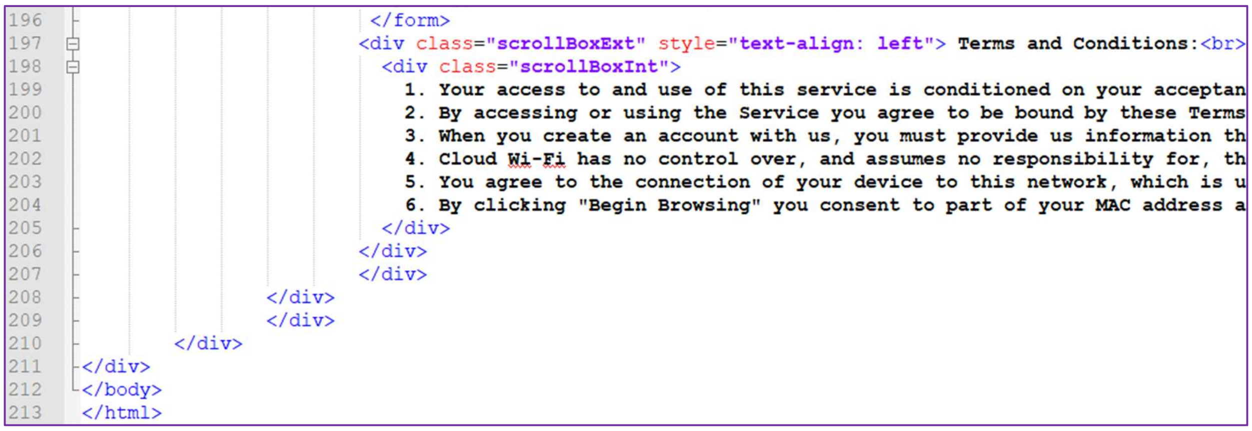
### HTML

The HTML (Hyper Text Mark-up Language) for the Splash/index page is below. One important thing to note is the "required" piece of text (Circled in blue) this is a straightforward way of validating your web form.

For example, because we have set the type of the Email input box to "Email" this means for it to be valid, it needs to be of a valid email format, like: "xxx@.xxx". Hence, if these requirements are not met, a box will appear informing the user to input a correct value. The same goes for the check box that asks if the user accepts the terms and conditions.



```
 <main>
 <div class="t1">
 <h1><center>Coventry City Cloud Wi-fi</center></h1>
 </main>
 </div>
 <div class="column side">
        <p> side</p>
 </div>


<div class="column middle">
        <div class="opacityBox">

                  <form method="POST" name="Frm_UsrInput" action="Valid+Store.php" onsubmit="return validate()" >
                          <div> Enter your E-mail below <br> to start browsing:
                          <br></br>
                                  <input name="email" type="email" placeholder="Email Address' required/>
                                  <input type="hidden" value="" name ="emailInput">
                                 <div style="font-size:11px">
                          <br></br>
                          <input id="TnC" type="checkbox" value="Agree" style="vertical-align: middle; margin-top: -1px' required/>
                          <br></br>
                          <div>
                          <input type="submit" value="Begin Browsing" >
                           </form>
```

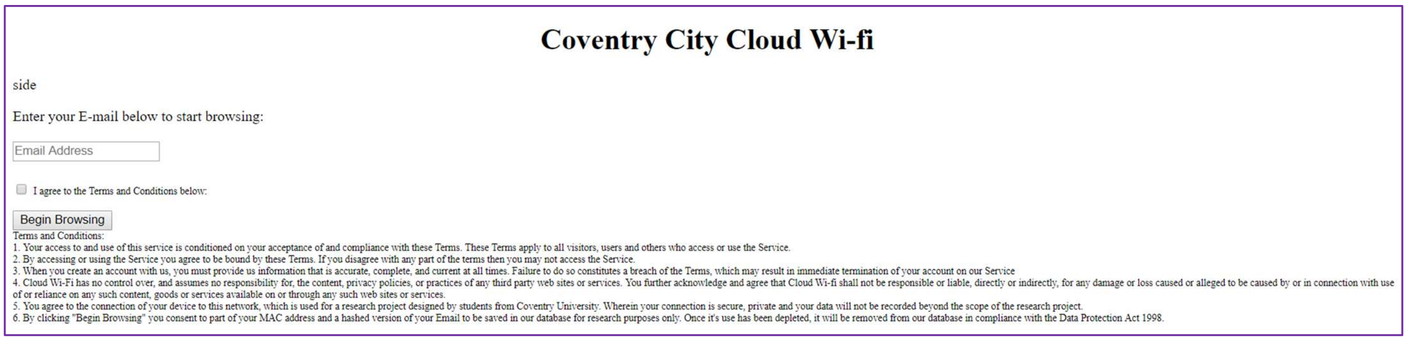


```
                           </form>
                          <div class="scrollBoxExt" style="text-align: left"> Terms and Conditions:<br>
                            <div class="scrollBoxInt">
                              1. Your access to and use of this service is conditioned on your acceptan
                              2. By accessing or using the Service you agree to be bound by these Terms
                              3. When you create an account with us, you must provide us information th
                              4. Cloud Wi-Fi has no control over, and assumes no responsibility for, th
                              5. You agree to the connection of your device to this network, which is u
                              6. By clicking "Begin Browsing" you consent to part of your MAC address a
                            </div>
                          </div>
                          </div>
                  </div>
                  </div>
        </div>
</div>
</body>
</html>
```

Harry Smith

**Remember your code can, and probably will be, different to mine.**

This code will create a webpage, when opened. That looks like this:



Coventry City Cloud Wi-fi

side

Enter your E-mail below to start browsing:

Email Address

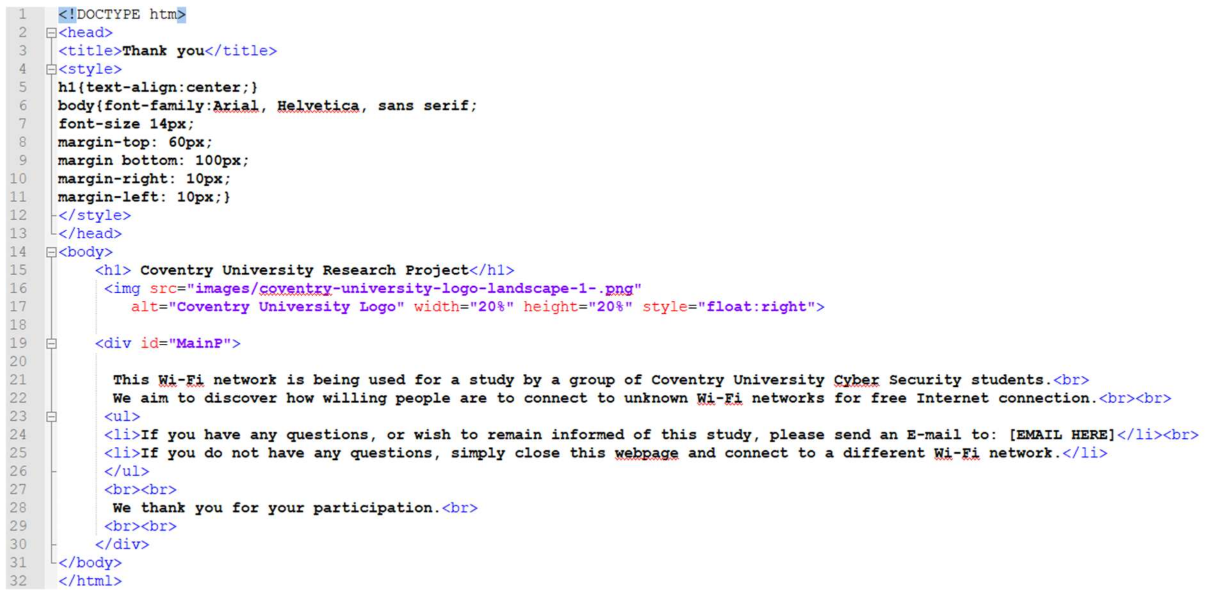
I agree to the Terms and Conditions below:

Begin Browsing

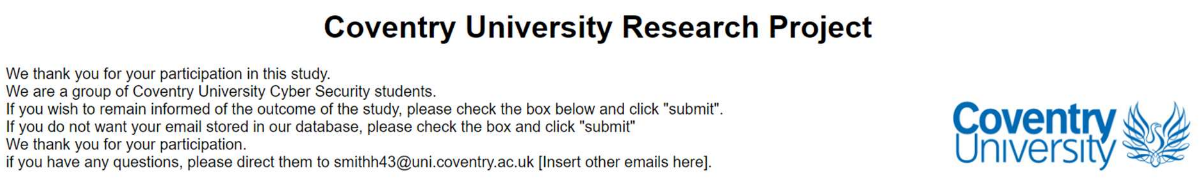
Terms and Conditions:
1. Your access to and use of this service is conditioned on your acceptance of and compliance with these Terms. These Terms apply to all visitors, users and others who access or use the Service.
2. By accessing or using the Service you agree to be bound by these Terms. If you disagree with any part of the terms then you may not access the Service.
3. When you create an account with us, you must provide us information that is accurate, complete, and current at all times. Failure to do so constitutes a breach of the Terms, which may result in immediate termination of your account on our Service
4. Cloud Wi-Fi has no control over, and assumes no responsibility for, the content, privacy policies, or practices of any third party web sites or services. You further acknowledge and agree that Cloud Wi-fi shall not be responsible or liable, directly or indirectly, for any damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods or services available on or through any such web sites or services.
5. You agree to the connection of your device to this network, which is used for a research project designed by students from Coventry University. Wherein your connection is secure, private and your data will not be recorded beyond the scope of the research project.
6. By clicking "Begin Browsing" you consent to part of your MAC address and a hashed version of your Email to be saved in our database for research purposes only. Once it's use has been depleted, it will be removed from our database in compliance with the Data Protection Act 1998.

It is a simple Captive Portal login page which asks the user for their email address and agreement to some terms and conditions.

I also needed to create a webpage that informed the users of what the purpose of this network was for. For this, I simply created a new HTML webpage that is going to be opened after the saving of user's information is finished. Below is the code and a picture of the page:



```
<!DOCTYPE htm>
<head>
<title>Thank you</title>
<style>
h1{text-align:center;}
body{font-family:Arial, Helvetica, sans serif;
font-size 14px;
margin-top: 60px;
margin bottom: 100px;
margin-right: 10px;
margin-left: 10px;}
</style>
</head>
<body>
    <h1> Coventry University Research Project</h1>
     <img src="images/coventry-university-logo-landscape-1-.png"
        alt="Coventry University Logo" width="20%" height="20%" style="float:right">

    <div id="MainP">

      This Wi-Fi network is being used for a study by a group of Coventry University Cyber Security students.<br>
      We aim to discover how willing people are to connect to unknown Wi-Fi networks for free Internet connection.<br><br>
     <ul>
     <li>If you have any questions, or wish to remain informed of this study, please send an E-mail to: [EMAIL HERE]</li><br>
     <li>If you do not have any questions, simply close this webpage and connect to a different Wi-Fi network.</li>
     </ul>
     <br><br>
      We thank you for your participation.<br>
     <br><br>
    </div>
</body>
</html>
```



Coventry University Research Project

We thank you for your participation in this study.
We are a group of Coventry University Cyber Security students.
If you wish to remain informed of the outcome of the study, please check the box below and click "submit".
If you do not want your email stored in our database, please check the box and click "submit"
We thank you for your participation.
if you have any questions, please direct them to smithh43@uni.coventry.ac.uk [Insert other emails here].

I felt that the main Captive Portal page does not look that impressive and anyone logging into our network will immediately think this is not a legit network and immediately disconnect. So, I decided to beautify the webpage with CSS (Cascading Style Sheets)

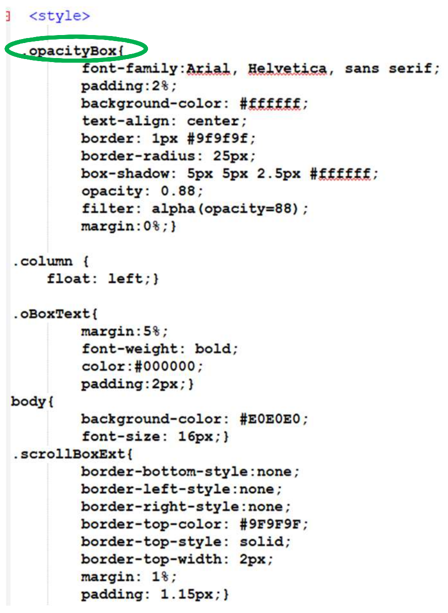
## CSS

Cascading Style Sheets is a language that's interpreted by a web browser, like HTML. It enables a webpage to appear different to just the base HTML. Essentially, it's a way of beautifying the website.

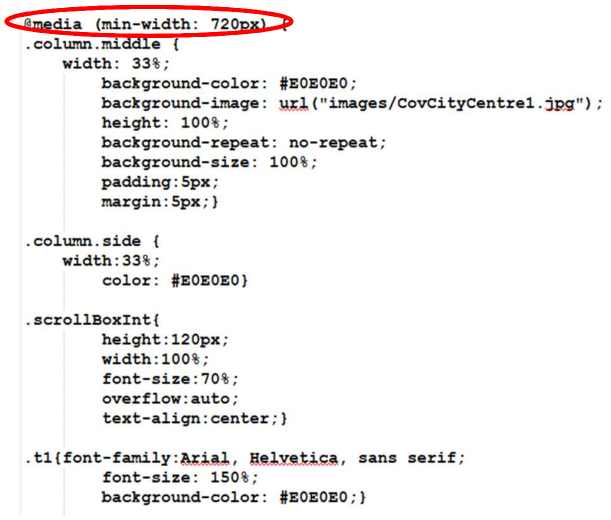
Please note the @media lines (circled in red). This enables additional functionality in the webpage, it allows the webpage to read the screen size it's being displayed on and thus change how things are displayed.

You can see I have three different media tags, one for Phones (max-width 721px), one for tablets (min-width 720px) and one for larger screens and monitors (min-width 1080px).

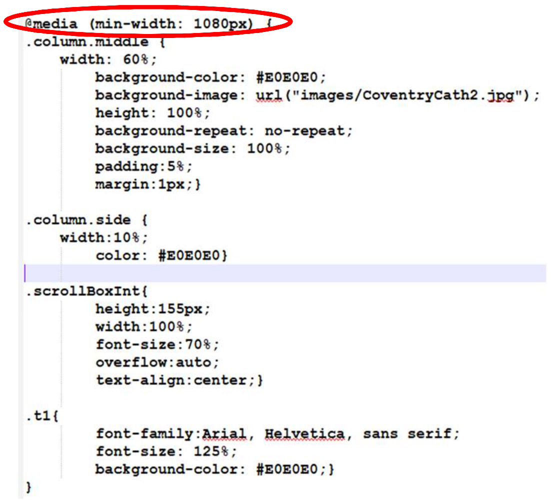
This enables us to scale the page and thus adds a little more functionality and adaptability to the page. (More code on the next page)



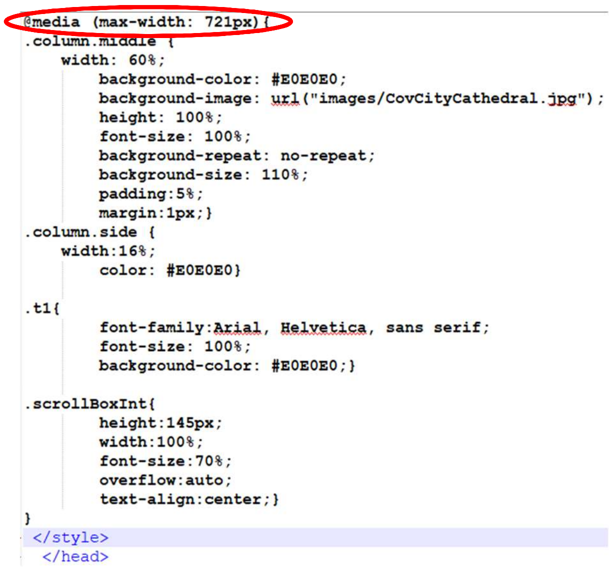
```
<style>

.opacityBox{
        font-family:Arial, Helvetica, sans serif;
        padding:2%;
        background-color: #ffffff;
        text-align: center;
        border: 1px #9f9f9f;
        border-radius: 25px;
        box-shadow: 5px 5px 2.5px #ffffff;
        opacity: 0.88;
        filter: alpha(opacity=88);
        margin:0%;}

.column {
    float: left;}

.oBoxText{
        margin:5%;
        font-weight: bold;
        color:#000000;
        padding:2px;}
body{
        background-color: #E0E0E0;
        font-size: 16px;}
.scrollBoxExt{
        border-bottom-style:none;
        border-left-style:none;
        border-right-style:none;
        border-top-color: #9F9F9F;
        border-top-style: solid;
        border-top-width: 2px;
        margin: 1%;
        padding: 1.15px;}
```



```
@media (min-width: 720px) {
.column.middle {
    width: 33%;
        background-color: #E0E0E0;
        background-image: url("images/CovCityCentre1.jpg");
        height: 100%;
        background-repeat: no-repeat;
        background-size: 100%;
        padding:5px;
        margin:5px;}

.column.side {
    width:33%;
        color: #E0E0E0}

.scrollBoxInt{
        height:120px;
        width:100%;
        font-size:70%;
        overflow:auto;
        text-align:center;}

.t1{font-family:Arial, Helvetica, sans serif;
        font-size: 150%;
        background-color: #E0E0E0;}
```



```
@media (min-width: 1080px) {
.column.middle {
    width: 60%;
        background-color: #E0E0E0;
        background-image: url("images/CoventryCath2.jpg");
        height: 100%;
        background-repeat: no-repeat;
        background-size: 100%;
        padding:5%;
        margin:1px;}

.column.side {
    width:10%;
        color: #E0E0E0}

.scrollBoxInt{
        height:155px;
        width:100%;
        font-size:70%;
        overflow:auto;
        text-align:center;}

.t1{
        font-family:Arial, Helvetica, sans serif;
        font-size: 125%;
        background-color: #E0E0E0;}
}
```



```
@media (max-width: 721px){
.column.middle {
    width: 60%;
        background-color: #E0E0E0;
        background-image: url("images/CovCityCathedral.jpg");
        height: 100%;
        font-size: 100%;
        background-repeat: no-repeat;
        background-size: 110%;
        padding:5%;
        margin:1px;}
.column.side {
    width:16%;
        color: #E0E0E0}

.t1{
        font-family:Arial, Helvetica, sans serif;
        font-size: 100%;
        background-color: #E0E0E0;}

.scrollBoxInt{
        height:145px;
        width:100%;
        font-size:70%;
        overflow:auto;
        text-align:center;}
}
</style>
 </head>
```

A few other things to note about CSS is that it's functionality directly impacts that of elements in the HTML code.

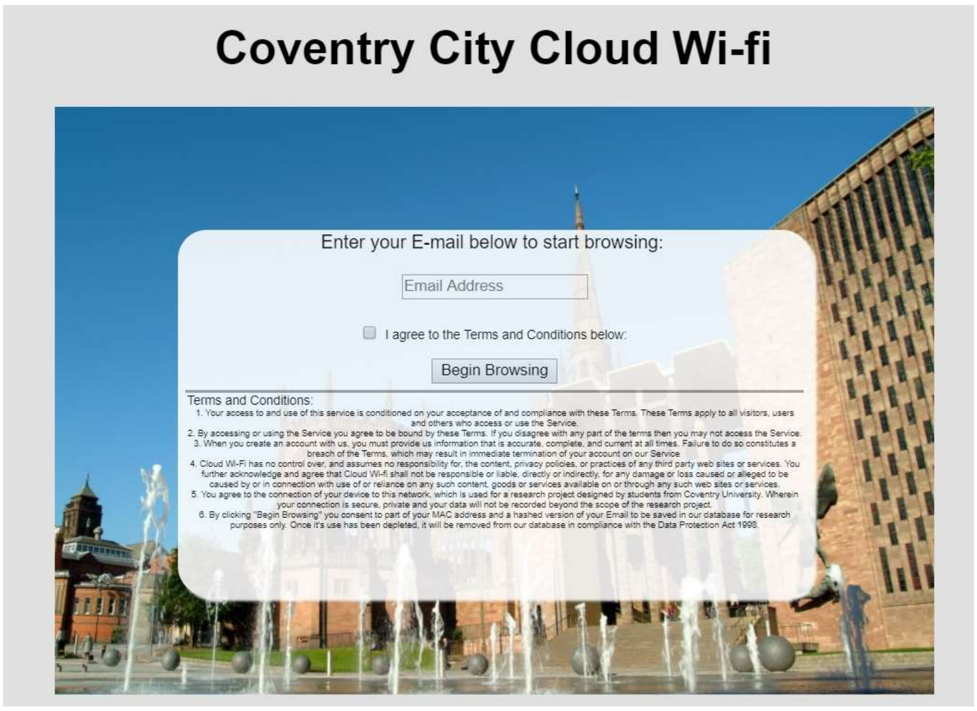
Circled in Green is a name of a class. A class is like a function in typical Python, C++, Java programming, a block of code which can be used repeatedly. However, a CSS class is slightly different in that it changes the features of some HTML code.

For example, circled in Green above, is the opacity box class. This class defines the rounded edge box which houses all the other information on the webpage. Anything edited inside this, will edit any other html objects that also have their class set to "opacityBox".

Another thing to note is the "div" tag in HTML. A div tag is essentially a box, anything inside of that box will be subject to properties of that box and anything else inside. It is a very powerful way of designing a webpage.

As seen above with the columnMiddle and opacityBox div tags, if the columnMiddle class was to be edited and its width reduced, every other div inside of the columnMiddle would also be edited too.

Of course, you do not have to use CSS in this way, or even at all! I just thought it was a subtle way to present my captive portal webpage.


Coventry City Cloud Wi-fi
Enter your E-mail below to start browsing:
Email Address
I agree to the Terms and Conditions below:
Begin Browsing

Terms and Conditions:
1. Your access to and use of this service is conditioned on your acceptance of and compliance with these Terms. These Terms apply to all visitors, users and others who access or use the Service.
2. By accessing or using the Service you agree to be bound by these Terms. If you disagree with any part of the terms then you may not access the Service.
3. When you create an account with us, you must provide us information that is accurate, complete, and current at all times. Failure to do so constitutes a breach of the Terms, which may result in immediate termination of your account on our Service
4. Cloud Wi-Fi has no control over, and assumes no responsibility for, the content, privacy policies, or practices of any third party web sites or services. You further acknowledge and agree that Cloud Wi-fi shall not be responsible or liable, directly or indirectly, for any damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods or services available on or through any such web sites or services.
5. You agree to the connection of your device to this network, which is used for a research project designed by students from Coventry University. Wherein your connection is secure, private and your data will not be recorded beyond the scope of the research project.
6. By clicking "Begin Browsing" you consent to part of your MAC address and a hashed version of your Email to be saved in our database for research purposes only. Once it's use has been depleted, it will be removed from our database in compliance with the Data Protection Act 1998.

## PHP and SQL programming

After creating both the webpage and database, we can now move onto the fun stuff, writing code to capture the user's IP and MAC addresses and if they consent, their Email too.

We will be using two languages:

- PHP (Hypertext Preprocessor) for the server-side processing of the information, page linking and data transfer
- SQL (Structured Query Language) for saving of user's information to database.

A combination of the both will enable us to save both a user's MAC address and Email to a Database.

It's important to note that PHP code will execute on opening of the webpage, NOT on click of a button or submission of a form, so to test this code we need to connect to the Wi-Fi network and navigate to the page. In my case this was typing `192.168.9.1:8080/index.php` into the URL bar.

To begin, we are going to be working on the captive portal html page.

Here, we will be grabbing the User's IP address (the address that our DHCP server allocated to the user) and performing an ARP (Address Resolution Protocol) request on the IP address. This ARP request is a protocol that queries and returns the MAC address of the device's interface card.

This will allow us to identify what device a user is using when connecting to the Wi-Fi network. This is because the first three sets hexadecimal characters is the manufacturer OUI (Organisationally Unique Identifier) and tell us what company the device belongs to.

First, we need to make a copy of the HTML page and change the file type from HTML to PHP, to do this we type:

```
Sudo mkdir /var/www/Old_HTML
```

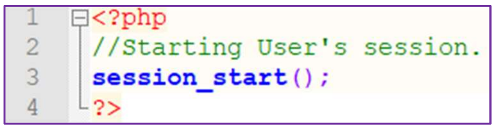
```
Sudo cp /var/www/index.html /var/www/old_HTML
```

```
Sudo nano /var/www/index.html
```

Then press ctrl+o and change the .html ending of the file, to .php, then agree to changing the name of the file by pressing "Y"

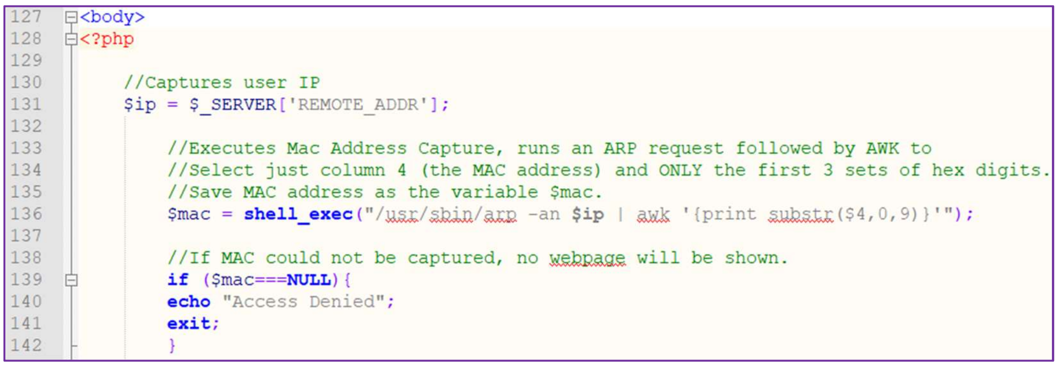
### Captive Portal Page's PHP and SQL (index.php)

Next, we can add the following code to grab the IP address and perform an ARP request. I will go through this line-by-line to explain its functionality in full:



```
1  <?php
2  //Starting User's session.
3  session_start();
4  ?>
```

- **Line 3 –** Initialises the PHP document to serve and use Session variables (Explained more here)



```php
127 <body>
128 <?php
129
130     //Captures user IP
131     $ip = $_SERVER['REMOTE_ADDR'];
132
133         //Executes Mac Address Capture, runs an ARP request followed by AWK to
134         //Select just column 4 (the MAC address) and ONLY the first 3 sets of hex digits.
135         //Save MAC address as the variable $mac.
136         $mac = shell_exec("/usr/sbin/arp -an $ip | awk '{print substr($4,0,9)}'");
137
138         //If MAC could not be captured, no webpage will be shown.
139         if ($mac===NULL){
140         echo "Access Denied";
141         exit;
142         }
```

- **Line 131** - Captures the User's IP address by using the inbuilt PHP method to do so, saves to variable called $ip.
- **Line 136** - Calls a bash shell to execute an ARP request on the IP address. Followed by calling awk (a text editing programming language) to print the 4th column of the ARP reply (the MAC address) and trim to receive the first 9 characters (the first 3 sets of hexadecimal characters). Saves result to variable $mac.
- **Line 139-141 -** If the MAC address could not be resolved and $mac is NULL, do not display the page. Exit PHP code.

This code will execute on interpretation of the php document, or when the webpage is accessed by the User. Therefore, it's important to be as quick as possible to load up. This was the most efficient way of retrieving a MAC address from a user's device.

Before testing this, we have to do one last thing. Enable the user www-data (Apache2's user) to execute ARP requests. To do this we enter the following code:

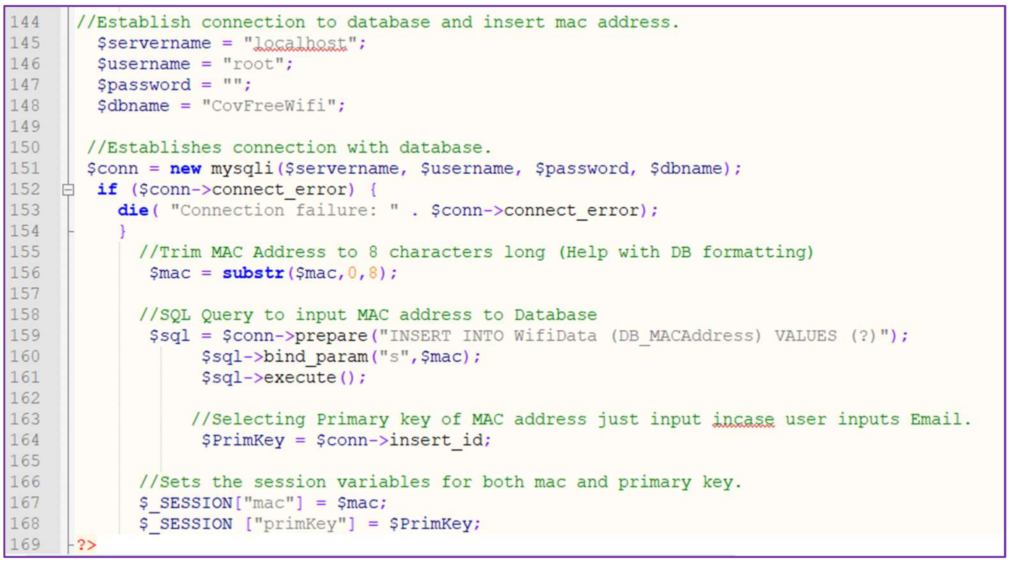
```
sudo chown -R root:www-data /var/www
```

This line will enable www-data to execute anything inside the /var/www directory as root and so let us execute the ARP command.

If you wish to test this, you can append "echo $mac;" to the line under line 136, this will print the value of $mac to the webpage, this will let you see the first three sets of hex digits of your device!

## Saving User's MAC address to Database

Now that we have the User's MAC address, we need to be able to save it to our database.



```php
//Establish connection to database and insert mac address.
  $servername = "localhost";
  $username = "root";
  $password = "";
  $dbname = "CovFreeWifi";

 //Establishes connection with database.
 $conn = new mysqli($servername, $username, $password, $dbname);
  if ($conn->connect_error) {
    die( "Connection failure: " . $conn->connect_error);
    }
      //Trim MAC Address to 8 characters long (Help with DB formatting)
       $mac = substr($mac,0,8);

      //SQL Query to input MAC address to Database
       $sql = $conn->prepare("INSERT INTO WifiData (DB_MACAddress) VALUES (?)");
            $sql->bind_param("s",$mac);
            $sql->execute();

           //Selecting Primary key of MAC address just input incase user inputs Email.
            $PrimKey = $conn->insert_id;

      //Sets the session variables for both mac and primary key.
      $_SESSION["mac"] = $mac;
      $_SESSION ["primKey"] = $PrimKey;
?>
```

- **Lines 145-148 -** Sets the variables used to log into the database.
- **Lines 151-154 –** Attempts to create a new SQL connection to the database, if there is an error with the connection, echo the error and halt connection.
- **Line 156** – Trims the MAC address down to 8 characters, so it appears in the format "*xx:xx:xx*" with no white space.
- **Lines 159 –** Start of the SQL prepared statement to insert the MAC address of the User's device. Notice the "?", this is used as a placeholder for the MAC address. (Placeholders and a prepared statement are used to mitigate an SQL injection vulnerability)
- **Line 160 –** Binds the $mac (User MAC Address) to the "?" in the SQL query. "S" is used to tell MySQL the variable is a string.
- **Line 161 –** Executes the Query.

Now that we have saved the MAC Address, we also need to extract the value of the Primary key for the MAC address. The reason for this is explained more in the section below.

- **Line 164 –** Saves the Primary Key of he inserted MAC address record to a variable.

### *Session Variables*

Finally, this is where sessions come into play, we need to send this data to the next page, which can be done through a POST or GET HTML method, however a Session is much more secure, acts like a class (so we can have multiple instances of each webpage where each instance of a Webpage is like an object of a class) and solely works in PHP, so we have no need for HTML coding here.

- **Lines 167-168 –** Sets both $mac and $primKey to be Session variables.

## Second PHP Document (Valid+Store.php)

Once a user clicks the submit button labelled "Begin Browsing" on the Captive Portal page, it will execute a different webpage called Valid+Store.php. This will hash the user's Email, input it to the database and open our final HTML page.

To transfer the data across webpage I decided to use both a HTML method called POST and a PHP Session.

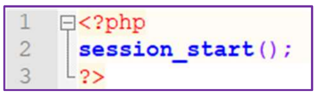
- POST – Secure Version of the HTML method GET which does not publish any data in the URL bar, meaning nobody can grab the plaintext Email address.

- Session - will enable us to have a different version of the webpages running for everyone. Think of it like a class, each new instance of the webpage will be a new session in the same way an object is an instance of a class.

- A session has only been created for both the MAC address and Primary key of said MAC address in the database. This will enable us to protect the information and not reveal the exact database location of a user's MAC address and Email.

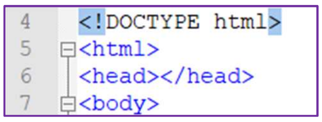
Below is the code for the Valid+Store.php page:

As before **Line 2** is the initialisation of the PHP Session



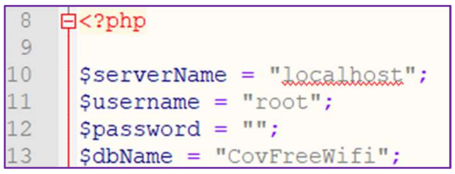
```php
<?php
session_start();
?>
```

Now, to run a PHP script, it needs to be embedded inside HTML tags, so after the first php Session initialisation, we start the HTML document. The rest of the PHP code will be in the body of the HTML page.



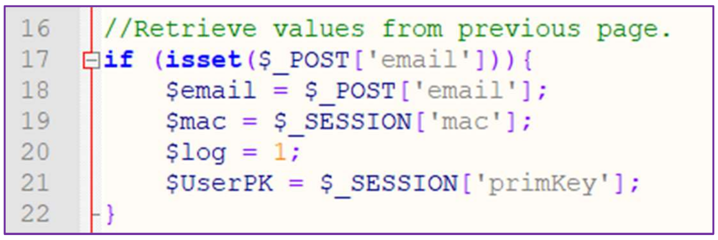
```html
<!DOCTYPE html>
<html>
<head></head>
<body>
```

As before, we set the login variables to their appropriate values:



```php
<?php

$serverName = "localhost";
$username = "root";
$password = "";
$dbName = "CovFreeWifi";
```

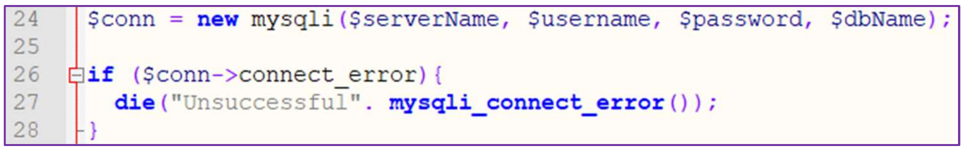
Next, we need to retrieve the data from the Captive Portal/Splash Screen page. To do this we write the following code:



```
16  //Retrieve values from previous page.
17  if (isset($_POST['email'])){
18      $email = $_POST['email'];
19      $mac = $_SESSION['mac'];
20      $log = 1;
21      $UserPK = $_SESSION['primKey'];
22  }
```

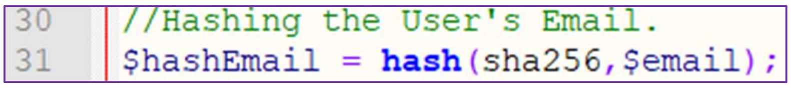
- **Line 16 –** checks that the email field on the Captive Portal page is populated (This should always happen as we did some HTML verification on the page)
- **Lines 18-21** – Sets the variables to their appropriate values
- *Note: $log is used to keep a track of how many times a user has logged in with a certain email.*

As before, we must open the connection to the MySQL database using the following code:



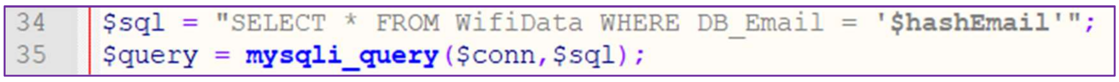
```
$conn = new mysqli($serverName, $username, $password, $dbName);

if ($conn->connect_error){
  die("Unsuccessful". mysqli_connect_error());
}
```

To hash the User's Email, PHP has a handy inbuilt module that will run a hashing algorithm for us. We call the module and tell it to hash the Email with sha256:



```
//Hashing the User's Email.
$hashEmail = hash(sha256,$email);
```

The first of our SQL Queries is below, this will extract all rows that have an Email field that's equivalent to that of our hashed email. There is no need for a prepared statement here as we are not dealing with data the user input.



```
$sql = "SELECT * FROM WifiData WHERE DB_Email = '$hashEmail'";
$query = mysqli_query($conn,$sql);
```

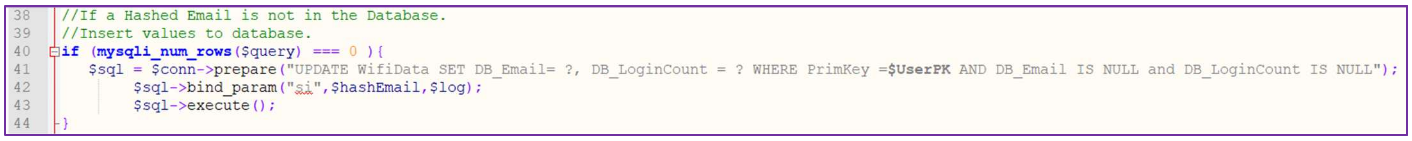
Next, I have an If selection statement that queries the number of rows returned by our query above. If the number of rows (we can see this with a handy PHP module), is equal to 0, we prepare a new SQL query.

This Query updates the record of the Primary key that was extracted back on the previous page, hence the use for the Session variable.

It then binds both the hashed email and a login count of 1 to the query and executes it.

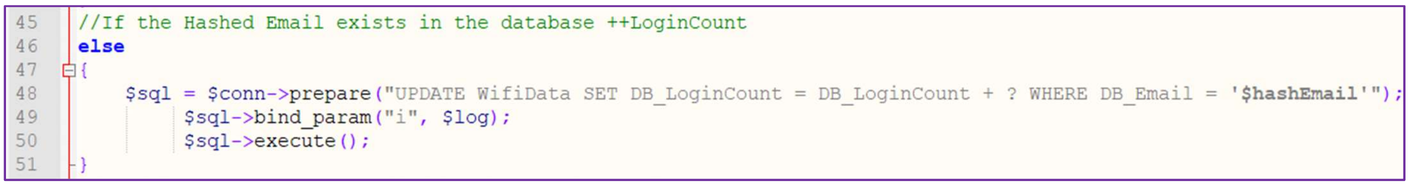
Notice the “IS NULL” section of the query, this tells the SQL to input both the email and login count where the value of the fields is equal to NULL. This next section of code is if the number of rows in

the SQL query is not equal to 0. This mean that the user’s Hashed email is already in the database. Due to a hash algorithm’s output being unique for each input, we can rely on it as a secondary key and thus use it for matching.



```
//If a Hashed Email is not in the Database.
//Insert values to database.
if (mysqli_num_rows($query) === 0 ){
    $sql = $conn->prepare("UPDATE WifiData SET DB_Email= ?, DB_LoginCount = ? WHERE PrimKey =$UserPK AND DB_Email IS NULL and DB_LoginCount IS NULL");
        $sql->bind_param("si",$hashEmail,$log);
        $sql->execute();
}
```

In this case, the Query below will simply update the login count field of the User’s record by +1. I decided to use a prepared statement here too, just to keep some kind of uniformity.

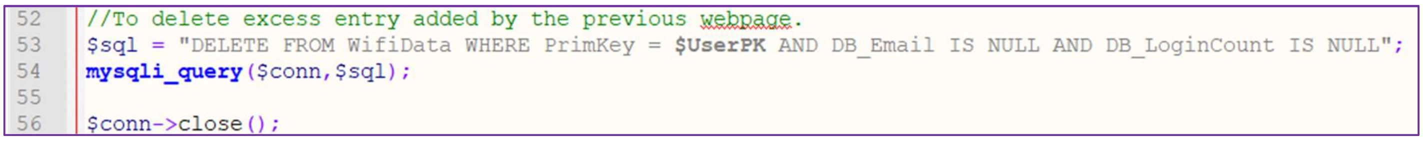


```
//If the Hashed Email exists in the database ++LoginCount
else
{
    $sql = $conn->prepare("UPDATE WifiData SET DB_LoginCount = DB_LoginCount + ? WHERE DB_Email = '$hashEmail'");
        $sql->bind_param("i", $log);
        $sql->execute();
}
```

Finally, the last section of our PHP code is this a query to delete a field from the Database. This record holds just the MAC Address and a NULL value for both the Email and Login count and is generated when the Captive portal page is opened. We delete this because, if a user has entered their email and it is now in the database, there is no need for a field with just their MAC Address in it.

We do not need this field as it’s a duplicate entry of one’s MAC Address.

The reason this works successfully is that it only deletes the field where the Primary key is the same as the Primary key generated from the Captive Portal page and where the Email and Login Count is NULL. If a user does not input their email address, this code never runs (as we don’t access this page) and so we keep the field.

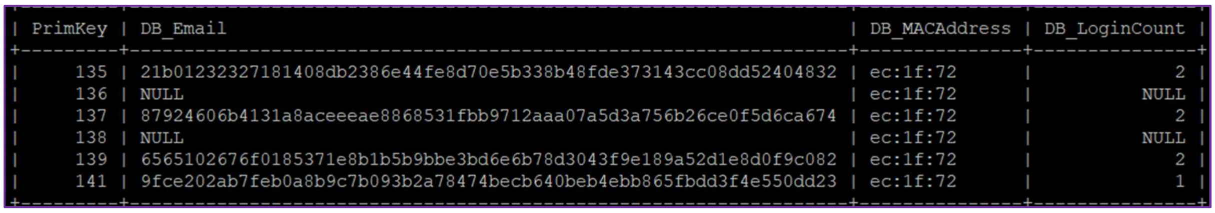


```
//To delete excess entry added by the previous webpage.
$sql = "DELETE FROM WifiData WHERE PrimKey = $UserPK AND DB_Email IS NULL AND DB_LoginCount IS NULL";
mysqli_query($conn,$sql);

$conn->close();
```

Now if we connect to our webpage, and use an Email to login, we will have a result in the database that looks similar to records 135,137,139 and 141. While if we just open the Captive portal page and close it, our code will generate records 136 and 138.

You can see the Database like this if you login to MySQL as before and type:

```
SELECT * FROM <tableName>;
```

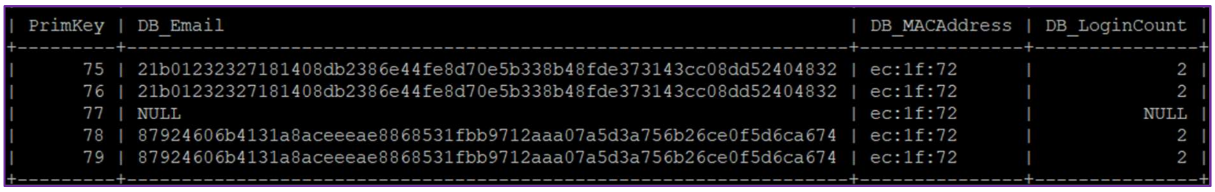


```
+---------+------------------------------------------------------------------+---------------+---------------+
| PrimKey | DB_Email                                                         | DB_MACAddress | DB_LoginCount |
+---------+------------------------------------------------------------------+---------------+---------------+
|     135 | 21b01232327181408db2386e44fe8d70e5b338b48fde373143cc08dd52404832 | ec:1f:72      |             2 |
|     136 | NULL                                                             | ec:1f:72      |          NULL |
|     137 | 87924606b4131a8aceeeae8868531fbb9712aaa07a5d3a756b26ce0f5d6ca674 | ec:1f:72      |             2 |
|     138 | NULL                                                             | ec:1f:72      |          NULL |
|     139 | 6565102676f0185371e8b1b5b9bbe3bd6e6b78d3043f9e189a52d1e8d0f9c082 | ec:1f:72      |             2 |
|     141 | 9fce202ab7feb0a8b9c7b093b2a78474becb640beb4ebb865fbdd3f4e550dd23 | ec:1f:72      |             1 |
+---------+------------------------------------------------------------------+---------------+---------------+
```

### *Why the Primary key?*

The reason we take the primary key from the previous page is that will be the entry of the MAC address for that specific user. If we do not know the specific Primary key of the user's MAC Address, we do not know what record to append both the Email and Login count to.

As you can see below, the MAC address is the same for all entries as is the hashed Email. However, if we did not get the Primary key of the MAC Address just entered, we would get multiple entries for each Email.

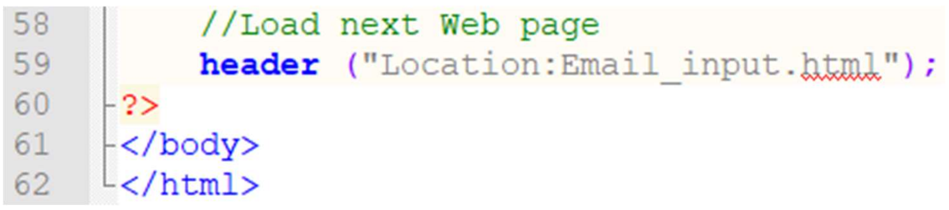


```
| PrimKey | DB_Email                                                         | DB_MACAddress | DB_LoginCount |
+---------+------------------------------------------------------------------+---------------+---------------+
|      75 | 21b01232327181408db2386e44fe8d70e5b338b48fde373143cc08dd52404832 | ec:1f:72      |             2 |
|      76 | 21b01232327181408db2386e44fe8d70e5b338b48fde373143cc08dd52404832 | ec:1f:72      |             2 |
|      77 | NULL                                                             | ec:1f:72      |          NULL |
|      78 | 87924606b4131a8aceeeae8868531fbb9712aaa07a5d3a756b26ce0f5d6ca674 | ec:1f:72      |             2 |
|      79 | 87924606b4131a8aceeeae8868531fbb9712aaa07a5d3a756b26ce0f5d6ca674 | ec:1f:72      |             2 |
+---------+------------------------------------------------------------------+---------------+---------------+
```

Furthermore, the OUI of a MAC address is not unique to each device, therefore we cannot use it in replacement of our Primary Key.

If we were to hash the User's Email on the first page, then we could use the Hash as a primary key, however this is not the intended task of the first page and so a Primary key must be used.

Finally, after we have inserted and removed all the data we need to from the database, we need to open the page which informs users what has just happened. The code below will open the new html file in the same tab as the Captive Portal webpage.



```
    //Load next Web page
    header ("Location:Email_input.html");
?>
</body>
</html>
```

## Finishing touches

Now we should have a fully working Wi-Fi hotspot that captures people's MAC addresses, and with permission a hashed version of their Email. From here there is only a few more things to do.

We need to create a new webpage called generate_204.html. This is due to Android phones, when connecting to a Wi-Fi network with a captive portal, will attempt to generate a 204 error code, this will tell a user that they need to sign into a network.

Essentially, when they tap to sign into the network, they will first visit the generate_204.html webpage, this will then route you to the captive portal page. We can simply do this with an HTML meta tag, see the following code:

```
<html>
<head>
        <meta http-equiv="refresh" content="0;URL='<YOUR WEBPAGE NAME">
</head>
<body></body>
</html>
```

Simply place this inside a html file called generate_204.html in /var/www and you are good to go.

Finally, we need to enter a final iptables command to route all traffic that connects to the network, to the Captive Portal page.

```
iptables -t nat -A PREROUTING -p tcp -m tcp --dport 80 -j DNAT --to-destination
192.168.9.1:8080

iptables save > /etc/iptables/rules.v4
```

## Conclusion

There you have it! A Beagle bone that is configured as a Wi-Fi hotspot to serve a webserver that collects user's information.

We personally plan to use this system to test a question:

*"Users of which make of device are more likely to login to an unknown Wi-Fi network to get an internet connection?".*

The project is a cumulative effort of a few Cyber Security students, we are a group of six, however four of us worked solely on the BeagleBone and programming side.

I - Harry Smith – worked on setting up the BeagleBone webserver and the HTML, CSS, PHP and SQL for the webpages as well as this write up of the project.

While my other team members, Owen Lloyd-Jones and Tom Newman investigated differing types of form validation and Luke Reading researched how to set up a database.

Below is both my LinkedIn profile link as well as my GitHub. All code I wrote will be on there for you to use, interpret and change as you see fit.

GitHub - https://github.com/HarrySmith2/Open-Ended-Project

Harry Smith LinkedIn - www.linkedin.com/in/harryejsmith

Owen Lloyd-Jones LinkedIn - https://www.linkedin.com/in/owenlloydjones/

Thank you again for your time in reading this mini project

- Harry Smith, Owen Lloyd-Jones, Tom Newman, Luke Reading

## Research Sources

**Installing Debian on Beagle Bone Black**

- https://learn.adafruit.com/beaglebone-black-installing-operating-systems/

**If you have no internet connection and DNS lookup from BBB:**

- https://www.adminsehow.com/2011/09/gateway-on-a-different-subnet-on-linux/

**Connect BBB to internet on Windows Device**

- https://billwaa.wordpress.com/2014/08/03/beaglebone-black-enable-usb-internet-sharing-from-windows/

**To setup the Wi-Fi access point:**

- https://fleshandmachines.wordpress.com/2012/10/04/wifi-acces-point-on-beaglebone-with-dhcp/

**If the DHCP server setup in the link above does not work:**

- https://learn.adafruit.com/setting-up-a-raspberry-pi-as-a-wifi-access-point/install-software#set-up-dhcp-server

**PHP and SQL Programming help**

- https://www.w3schools.com/php/php_mysql_intro.asp
- http://php.net/manual/en/set.mysqlinfo.php

**Apache**

- https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-debian
- http://kingofprotons.blogspot.co.uk/p/beaglebone-web-server.html