DIY Captive Portal for Public Education

Aaron Brighton

Introduction

DISCLAIMER: This document is provided in the hopes that it can be developed on to provide your institution with a working Captive Portal system at an affordable and reasonable cost. That being said I make no guarantee that this solution will work and thus will not be responsible for any incurred costs or damages caused by following the instructions. I have not had the chance to deploy this solution on a large network with a large number of clients, as such cannot guarantee any performance.

In this guide, I will assume the individual implementing such a solution has knowledge of networking (including the network to be deployed on), building computer systems, active directory, linux, web server, and web scripting.

Don't let the expectations scare you, the expectations are just to show you what areas this setup touches on. You do not need to be a expert in any of the areas mentioned, simply you need to have some knowledge and experience dealing with these areas.

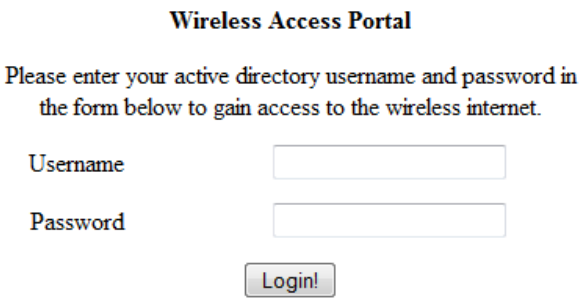
Problem

The problem we will be taking on in this guide is the issue many educational institutions face today, with the expectation of providing wireless internet access in their schools. Public institutions often do not have the money to spend on expensive equipment, run cables to access points, or pay contractors to come in and setup a captive portal system. Luckily for you, the same level of functionality and security can be brought to your institution without the need for all the expensive equipment, cables or labour.

What is a Captive Portal?

I have mentioned the term “captive portal” a few times so far, are you asking yourself what a captive portal is? Good question, let me explain.

A captive portal is a technique that forces an HTTP client (a web browser such as Internet Explorer, or Firefox) to display a web page to a client that has connected to a wireless network. This web page is used as an authentication tool, meaning you will not be permitted to access the network or internet further until you have proceeded with being authenticated by that web page. This web page could ask you to do a number of things, one would be to accept the terms of use (often used in a HotSpot area, like a coffee shop), however for our purposes the web page would require the user to authenticate using a username and password, before gaining access to the network and/or internet.



Goal

The goal of this guide is to provide an affordable DIY solution for providing your educational institution with wireless internet that requires students and/or faculty to authenticate with their active directory username and password.

- Wireless Internet Connectivity throughout your institution.
- Only users on the Domain Controller for the institution, may authenticate.
- Until authenticated, permit no network access, except to the authentication server.
- After authenticated, open up access to the internet, while maintaining limited private network access (to protect the network).
- Provide expansion for logging packets to active directory users.
- Provide expansion for a management console to be built on top of setup.

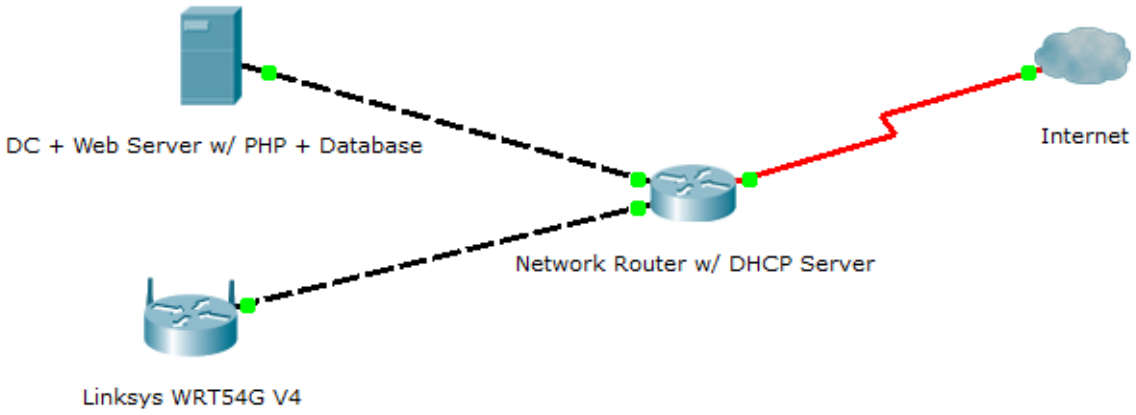
Minimum Required Devices

Below are the minimum required devices/items, for employing a solution similar to the solution we take you through in this guide.

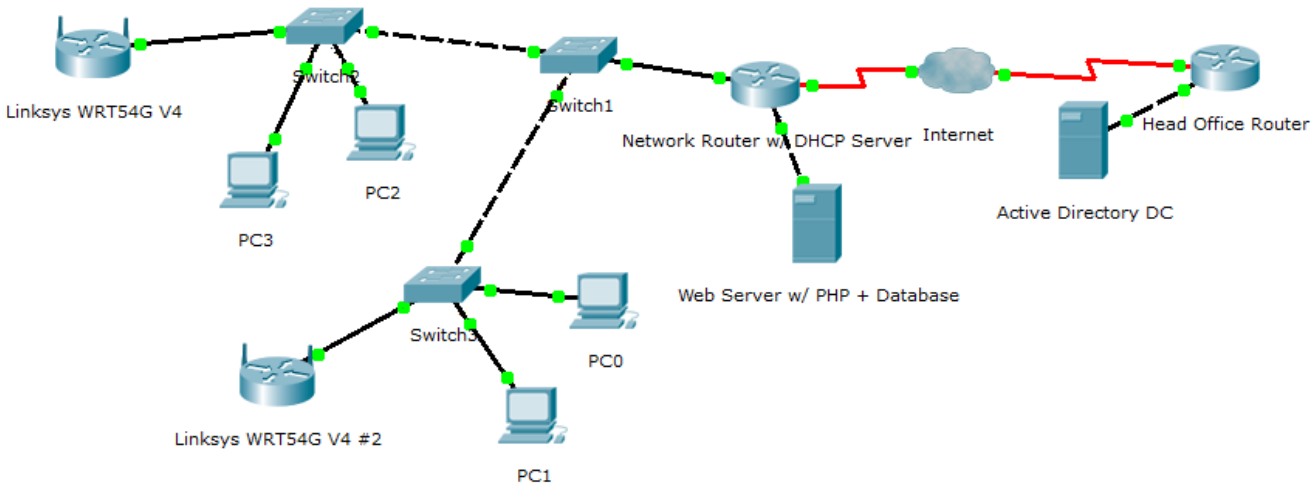
- Existing private network with DHCP.
- Linksys WRT54G or compatible routers (with 4 MB Flash), with DD-WRT Standard Firmware.
- Active Directory Domain Controller
- Web Server w/ PHP support, and Database Server.

The above items are the **minimum** required devices for employing the following setup, with some minor modifications. The WRT54G router is merely an example router, any other router with a minimum 4 MB Flash that is compatible with the free DD-WRT firmware should also prove to work fine.

Using these devices, your most basic network structure would look like the following.



Now that you have seen the most basic network structure, let's face the facts your institutions network is no where near as simple as that, you might even have VPN branching multiple locations. The following image, represents what a large institution's network may look like and how each device would likely be implemented.



The above image as you can see is clearly more complex, however is much more realistic. On the right side of the Internet cloud you have the Head Office for your institution which could be on the other side of the city, connected over the Internet using a virtual private network. On the left side of the cloud you have a school or campus, you've got the gateway router with a DHCP server, for that school or campus. You've also got the Authentication server for that school, it still authenticates users with the Domain Controller at head office though, as you can see all we did was split the DC from the Authentication server. Another thing that should be noted is that the Access Points (WRT54G's) are connected directly to the same basic switches that the normal PC's are connected, no need to run separate cables from a wiring closet to the access points, you'll see how this works later.

Now that I have explained what devices are required, and the network layout for implementing this sort of captive portal system, what kind of bill do you think your institution will have to flip for this sort of system? Well let's look at what pieces of hardware you need, and how much they will set you back.

- The WRT54G series of routers, will set you back approximately $40, per router.
- You'll need a box to use as the Wireless Authentication Server, you should look at a P4 (2.6 GHz), with 512 MB RAM, 10 GB Hard Drive. That shouldn't set you back more than $100.

You've just spent $140 for a full blown wireless solution, not covering your entire institution, just add $40 for each new WRT54G you purchase. Another option to save even more, if you don't have a high volume of wireless clients, you could simply use one WRT54G and purchase some basic access points and have them simply repeat the wireless network put out by the WRT54G.

Not bad eh?

In the next section I will walk through the detailed steps to getting all this hardware to work for us.

Implementation

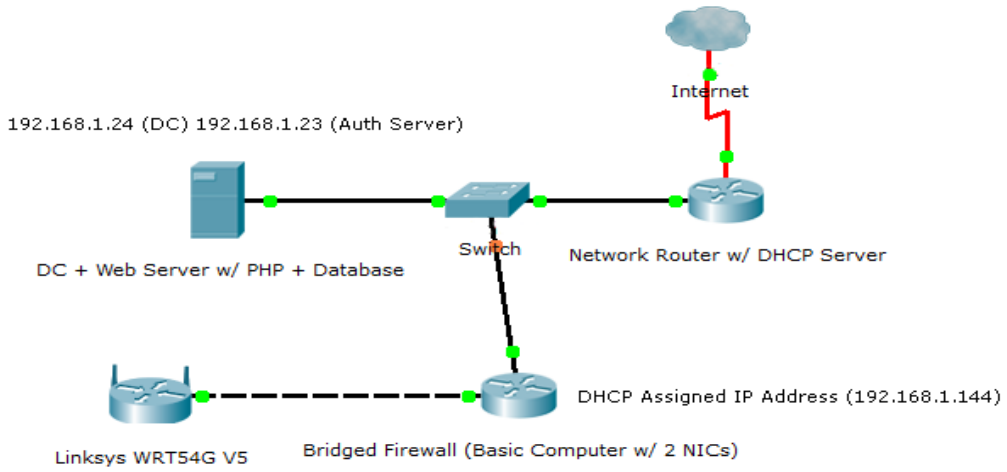
The implementation that I will be using is slightly different than the above implementation due to a couple simple reasons, first reason is that I do not have a huge network spanning multiple locations over a VPN, the second reason is that the WRT54G router I am using is v5, which means it has a 2MB Flash instead of a 4MB flash, with a 2MB flash I can only install a micro build of DD-WRT it will not be able to support the extra firewall features that a 4MB device would support. To deal with this issue, I have an old P3 with two network interface cards (NICs) installed, I will bridge them and then firewall that bridge.

These are the devices I will be using to implement this Captive Portal.

- **Existing Private Network w/ DHCP & Internet (192.168.1.0/24)**
- **DC/Wireless Authentication Server**
  - P4 (1.3 Ghz)
  - 512 MB RAM
  - 20 GB Hard Drive
  - Host: Windows Server 2003 (Domain Controller)
  - Guest (Virtual Machine): Debian Linux (Lighttpd Web Server+PHP5+MySQL)
- **Bridged Firewall (In Place of WRT54G v1-4 Built In Bridging Firewall)**
  - P3 (551 Mhz)
  - 128 MB RAM
  - 8.5 GB Hard Drive
  - Debian Linux
- **WRT54G v5**
  - 2 MB Flash
  - DD-WRT V24 Micro Firmware

Below is a network overview of the environment and devices I will be using to demonstrate and walk

you through this guide.



Step 1: Configure Access Point

The first device we should configure will be the access point (wrt54g) that way we know it is working properly before moving on to the more complicated things. It is not the goal of this tutorial to teach you how to flash a routers firmware, nor is it the goal of this tutorial to reteach what has already been taught so I am going to direct you to the following two guides for flashing your WRT54G and configure it to be used as a basic access point.

http://www.dd-wrt.com/wiki/index.php/WRT54G#WRT54G

http://www.dd-wrt.com/wiki/index.php/Wireless_Access_Point

Step 2: Authentication Server

For the authentication server you could have this run on IIS on the same server with your Domain Controller, however for this tutorial I will be using Debian Linux to run my Authentication Server, as I feel more comfortable with Linux servers.

I have Oracle VM Virtualbox installed on my DC server, I then have a guest OS with a bridged network interface running inside a virtual machine. This guest will be my Debian Lenny authentication server. Below listed is the software I have installed my Debian Linux authentication server, to facilitate the needs of this Captive Portal setup.

- Lighttpd Web Server (configured to work with PHP5)
- PHP5-cgi
- MySQL Server
- OpenSSH Server

First thing you should do before proceeding is making sure you have a Domain Controller setup with users that you can use to authenticate against, I will let you take care of this step as you most likely

already have a Domain Controller on your network providing you are using Active Directory as your directory service.

Again it is not the goal of this guide to teach you how to install Debian Linux, I trust you can take care of this step, and if not Google is your best friend. I used the Debian Linux net install disk. My installation is just a basic CLI, no Graphical User Interface. On the select & install dialogue of the installation I deselected everything including Standard System.

For this tutorial I'm going to assume that the Domain Controller (Windows 2003) is using the ip address 192.168.1.24, and that the Debian Linux authentication server is using 192.168.1.23 as an ip address.

Let's SSH into our Debian Linux machine, I will use putty for my ssh client you can use what ever ssh client you prefer.

What we are gonna first do is install a MySQL server, the MySQL server will be used to store information on which clients have been authenticated, it is the core information store for this entire Captive Portal system.

```
debian:~# apt-get install mysql-server
```

Continue with whatever prompts are displayed, remember what you enter for the MySQL root password, as you will need it later to setup a database and user. This step may take a few minutes as mysql-server is a hefty package.

After the MySQL Server has finished installing we will want to install our web server. Usually when someone speaks of a web server it is often running Apache, however for our purposes a lighter server is available known as Lighttpd (pron. Lighty). I have chosen to use Lighty, as I am running this Auth server in a Virtual Machine so I want it to be as light as possible.

```
debian:~# apt-get install lighttpd
```

Continue with any prompts, once Lighty has finished installing we want to install PHP5 in its CGI form, so the following command should do this for us.

```
debian:~# apt-get install php5-cgi php5-mysql php5-ldap
```

Again continue with any prompts, now we are going to have to configure Lighttpd, to run PHP scripts through the PHP engine, but first let's install sudo as it will be necessary later when one of our scripts calls on the arp cache.

```
debian:~# apt-get install sudo
```

First thing we need to modify is our MySQL configuration we need to permit connections from hosts

other than localhost, so we modify the MySQL configuration file to allow this, you can use your favourite editor, I will use nano.

```
debian:~# nano /etc/mysql/my.cnf
```

```
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
#bind-address           = 127.0.0.1
```

```
debian:~# /etc/init.d/mysql restart
```

We must edit our php configuration “/etc/php5/cgi/php.ini”.

```
debian:~# nano /etc/php5/cgi/php.ini
```

```
...
cgi.fix_pathinfo = 1
```

Save that file, now we need to add a user for the Lighttpd server and give that user sudo permissions, so that the authentication page can run privileged commands (arp commands). Create a user named www, and at it to the sudo user list.

```
debian:~# useradd www
debian:~# nano /etc/sudoers
```

```
# User privilege specification
root    ALL=(ALL) ALL
www     ALL = NOPASSWD: ALL
```

```
debian:~# /etc/init.d/sudo restart
```

Next thing we are going to do is modify the Lighttpd configuration, let's open up Lighty's configuration file “/etc/lighttpd/lighttpd.conf”, and make the following modifications.

```
debian:~# nano /etc/lighttpd/lighttpd.conf
```

```
...
server.modules          = (
            “mod_access”,
            “mod_alias”,
            “mod_accesslog”,
            “mod_fastcgi”,
...
```

```
## Use ipv6 only if available.
#include_shell "/usr/share/lighttpd/use-ipv6.pl"
...
## error-handler for status 404
#server.error-handler-404 = "/error-handler.html"
server.error-handler-404 = "/redirect.php"
...
## virtual directory listings
dir-listing.encoding        = "utf-8"
server.dir-listing          = "disable"
...
## change uid to <uid> (default: don't care)
server.username             = "www"

## change uid to <uid> (default: don't care)
server.groupname            = "www"
...
fastcgi.server = ( ".php" => ((
                    "bin-path" => "/usr/bin/php5-cgi",
                    "socket" => "/tmp/php.socket"
                )))
```

Save the configuration. Before we proceed to restart our Lighty server we need to change ownership on a few files, as we have changed the user and group of Lighty.

```
debian:~# chown www:www /var/log/lighttpd
debian:~# chown www:www /var/log/lighttpd/access.log
debian:~# chown www:www /var/log/lighttpd/error.log
```

now let's proceed to restart the Lighttpd web server using the following command.

```
debian:~# /etc/init.d/lighttpd restart
```

Now that the web server has been restarted, our setup for this server, using pre-built packages is complete. We need to add all the custom scripts to it, that allow our Captive Portal to work. Let's discern what this server's purpose is in the big picture.

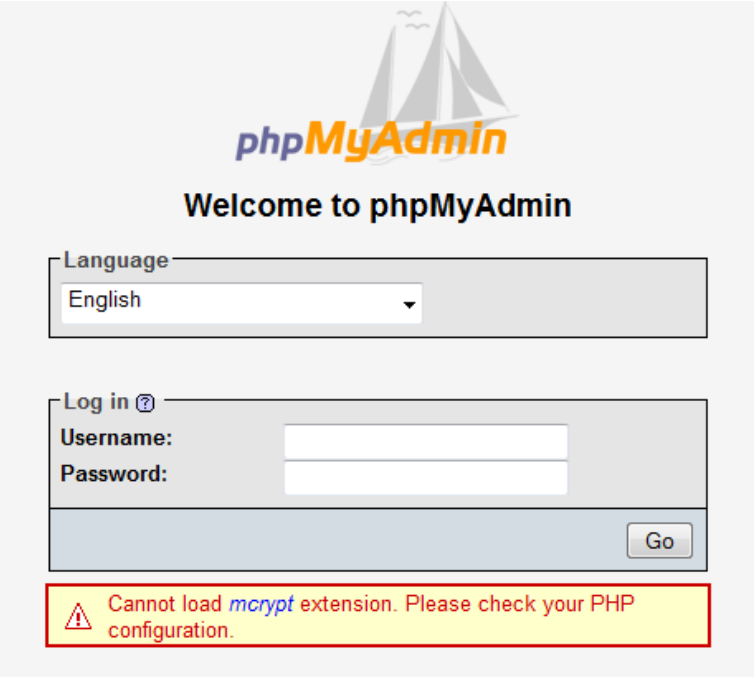
- Provide an Authentication (Login) Interface for the End-User
- Authenticate that user's credentials (Username & Password)
- Periodically expire inactive end-users.

This will require a few scripts to be created, I have wrote some scripts to take care of these tasks, you will find them in the accompanying “authServer” folder under the /var/www folder. You must upload these scripts to the /var/www directory on your Debian server, you could do this in a number of ways and it's not the goal of this tutorial to teach you how to transfer them. You should also delete any files

that are located in this directory by default.

Now another utility that you will find quite useful and I will use it to further this tutorial, is a script called phpMyAdmin. I advise you to download it from here, and upload it to the same /var/www directory on our Authentication server. Once uploaded make sure that the directory it has been extracted to is named "/var/www/phpmyadmin".

Now on the machine your using your ssh client on, open a web browser and enter the IP address of your web server (192.168.1.23) followed by phpmyadmin (ex. "http://192.168.1.23/phpmyadmin/"), you should be presented with a screen similar to the one displayed below.



You should be able to safely ignore the mcrypt error, as we will not be using phpMyAdmin for production use, simply to setup the database for our purposes.

Login using the form seen above with your MySQL username and password. The username will be "root" and password will be whatever you set it as during the MySQL installation earlier.

Again this guide's purpose is not to teach you how to use phpMyAdmin you can find decent tutorials on Google for using phpMyAdmin and managing a MySQL server.

We want to create a new user and database, let's create a user named "macs" and a new database named "macs" that the user macs has privileges to. Keep track of the username and password of the user you created, as well as the database you created.

Now you will want to create a single table in this database named `active_clients`, you should be able to run the below function as an actual query to have the table setup.

```
CREATE TABLE `macs`.`active_clients` (
`id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
```

```
`ip_address` VARCHAR( 15 ) NOT NULL ,
`mac_address` TEXT NOT NULL ,
`username` TEXT NOT NULL ,
`time_authenticated` INT NOT NULL ,
`active` INT NOT NULL
) ENGINE = MYISAM ;
```

Now that we have the database setup and working we can move onto configuring the scripts we uploaded earlier and testing them, we need to modify all but one of the files provided to you. The first script will be the more complicated script to modify but is the most important as it provides the interface between our PHP scripted authentication page and your Active Directory Domain Controller.

```
debian:~# nano /var/www/adLDAP.php
```

Now you want to focus on the section below, as this is where we tell this class how to connect to our Domain Controller, as an Active Directory admin you should be able to decipher what goes where.

```
 /**
    * The account suffix for your domain, can be set when the class is invoked
    *
    * @var string
    */
      protected $_account_suffix = "@dc.pri";

    /**
    * The base dn for your domain
    *
    * @var string
    */
      protected $_base_dn = "DC=dc,DC=pri";

    /**
    * Array of domain controllers. Specifiy multiple controllers if you
    * would like the class to balance the LDAP queries amongst multiple servers
    *
    * @var array
    */
    protected $ domain controllers = array ("win2003.dc.pri");
```

Save that file, now you want to edit something a little simpler “auth.php”, this file is responsible for the Authentication page displayed to the end-users when they are being prompted to authenticate, it is also responsible for processing their credentials and adding them to the allowed list.

```
debian:~# nano /var/www/auth.php
```

Now you want to look for the part of this file that has the MySQL connection information, and replace it with the appropriate information that you just created in the MySQL server.

```
$mysql['host'] = 'localhost';
$mysql['user'] = 'macs';
$mysql['pass'] = 'FJbCSbW3GBMP3QK8';
$mysql['dbname'] = 'macs';
```

One last file to modify and that is the script that takes care of expiring clients who's session has expired and are no longer active on the network.

```
debian:~# nano /var/www/cronStatCheck.php
```

This script also has a section with MySQL connection info and it needs to be replaced.

```
$mysql['host'] = 'localhost';
$mysql['user'] = 'macs';
$mysql['pass'] = 'FJbCSbW3GBMP3QK8';
$mysql['dbname'] = 'macs';
```

Let me explain how this file works, as it is important to understand how clients are determined to still be active or not. This script will be run as a cron job every 5 minutes (or 10 whatever interval you believe is appropriate). This script checks the database and compares the time a user was authenticated by the login script and checks it against the session time limit which is defined within this very file.

```
// 5 Minutes
$maxSessionTime = 5*60;
```

If the user is over the maxSessionTime then a bunch of checks are run, the script checks to see if it can ping off the IP address that was used when they initially authenticated (which shouldn't have changed, in most cases) , if it receives a response to that ping it checks its arp cache to see what mac address corresponds to that IP address, if that mac address is the same as this user's then it maintains their authentication status by setting a new time for time_authenticated in the database. Otherwise it sets the user to not active in the database, requiring them to reconnect when they log back in.

Now let's set this script to run as a cronjob every 5 minutes.

```
debian:~# crontab -e
```

```
# m h  dom mon dow   command
*/5 * * * * /var/www/cronStatCheck.php
```

Save this crontab, now this server is basically up and running let's attempt to authenticate and if we can successfully authenticate via the authentication page on this server then all is well, and we can move on to dealing with the bridge firewall.

Step 3: Bridged Firewall

Now that we have finished up the authentication server, the hard part is over with. The setup to this bridged firewall is very simply in comparison, that being said this step would be made even easier if we had used a WRT54G v1-4, which I strongly recommend you use or a similar 4 MB Flash dd-wrt compatible device.

Now I assume you have Debian Linux installed on this machine with 2 NIC's that are working (i.e. Drivers are all compatible with Linux and installed), first thing we will need to do is bridge our two interfaces (eth0,eth1) will become (br0).
Let's start by installing the bridge-utils package.

```
firewall:~# apt-get install bridge-utils
```

We are going to modify our interfaces file so that we have a bridge brought up whenever we restart.

```
firewall:~# nano /etc/network/interfaces
```

```
# The primary network interface
#allow-hotplug eth0
#iface eth0 inet dhcp

auto br0
iface br0 inet dhcp
        bridge_ports all
```

Now let's restart the box, just so we are sure the network interfaces come up properly.

```
firewall:~# shutdown -r now
```

After our box has come back up, and all networking is working properly and bridged, let's ssh into our bridged box using a client such as Putty. We need to install php-cli, as we have a script that is written in php that will sync the firewall on this bridged firewall to that of the MySQL database we setup earlier.

```
firewall:~# apt-get install php5-cli php5-mysql
```

We now need to configure the firewall, again I have provided scripts, if you look in the firewall folder "/etc/network/if-up.d/" you'll see a file named iptables we need to place this file at "/etc/network/if-up.d/iptables" on the bridged firewall computer. Again I'll leave it up to you to get the files moved there. You'll also need to have the permissions set to 755 as it will need to be executable. This file will be executed every time the network interface goes up.

There are two more files you need to move over to the bridged firewall and they are located in the

/etc folder inside firewall. The first one we are going to copy over is firewall.conf again this file needs to be chmod 755 to allow it to be executed. This file should be looked at as it contains certain things that need to be changed to be adapted to your network. A few IP Addresses are used in this file, you should swap them out for the appropriate addresses for your implementation, as well there is a few MAC addresses present these too should be swapped out for the MAC addresses that your implementation is using. This script sets up the firewall to only allow restricted traffic to the network, for non-authenticated users.

The last file we need to copy over to the /etc/ directory is "getMasterFirewall.php", this script is another key part to this entire Captive Portal system. This script queries the MySQL server we setup earlier and then modifies the firewall allowing authenticated users access and/or blocking expired users. This file should also be set to chmod 755, as it is also going to be executed. This script notably runs constantly however it does take a 3 second break after processing.

Now restart your Bridged Firewall.

```
firewall:~# shutdown -r now
```

There you have it, your Captive Portal should be working. Find a device with a wireless NIC and connect to your wireless network and see what happens.

Final Thoughts

This setup is relatively simple, and obviously needs some polishing, given only a week of work (1-2 hours a day) went into producing this solution for an affordable captive portal, the ruff edges should be understandable.

I'd like to point out the areas of this setup that are open to improvement and are expected in production.

- Customized "Look" for Authentication Page
- Management Console to view, add, and remove active clients. (phpMyAdmin works atm)
- A way to logout, a simple webpage that logs the user out could be added to favourites?
- Longer more permanent logging of authenticated users and when.
- AJAX Improvement to refresh auth page after being authenticated.

I want to speak about that last point for a moment, what I mean by that is, some individuals like to save tabs in their browsers. When they open their browser after connecting to the wireless their tabs are now all displaying a login page... They just lost their saved tabs? With an AJAX addition to the authentication page you could have the page know when the user has authenticated and all those tabs that were redirected to the login page can then be redirected back to their original destination.

The second last point I made was more logging, some institutions are required to keep logs of all the packets coming into and leaving their network and they need to be able to trace those packets to a person. I'm going to assume such an institution currently has a packet logging system, if someone

were to look at those logs they would simply need a script that can query the existing MySQL database for that mac address and who was authenticated during the time of that packet being sent through the network.

We've looked at improvements that could be added to the system easily enough, how about the known bugs that still need to be ironed out?

- If a user has their session killed, yet they still have a connection active with a server somewhere, that connection will stay active, only new connections will be dropped/blocked. **Possible fix:** Addition of appropriate iptable rules to the firewall.conf script.
- If a user has a tab or their homepage using https (SSL), than the webpage will not be redirected to the login script, as we do not have SSL configured on our authentication server. **Possible Fix:** Configure your web server for SSL.
- The way the system checks to see if a user has expired, or is not present on the network any more is by sending a ping to their computer, providing they are located at the same IP address that they used to authenticate. It's possible they changed and set their IP address to something static, or got a new IP address from the DHCP server. The other issue here is that, if the user has a firewall blocking ICMP on their machine, then the expiry script will think they are inactive and unauthenticate them requiring them to reauthenticate.
- Currently the firewall is setup to allow all pings through regardless of whether the user is authenticated or not, this can easily be changed with the addition of a few iptable rules.

I hope this guide has proved useful in your endeavours to bring Wireless internet to more of the world, even if it is restricted to your institutions personnel. Below are the main sources used to develop this solution, as well my contact information if you have questions or would like to use my services.

Sources:

http://www.linuxquestions.org/questions/linux-security-4/iptables-apply-certain-rules-to-certain-mac-addresses-819218/
http://www.howtoforge.com/lighttpd_mysql_php_debian_etch
http://www.dd-wrt.com/wiki/index.php/Wireless_Access_Point
http://www.dd-wrt.com/phpBB2/viewtopic.php?t=75182

Contact

Aaron Brighton
aaron@aaronbrighton.ca