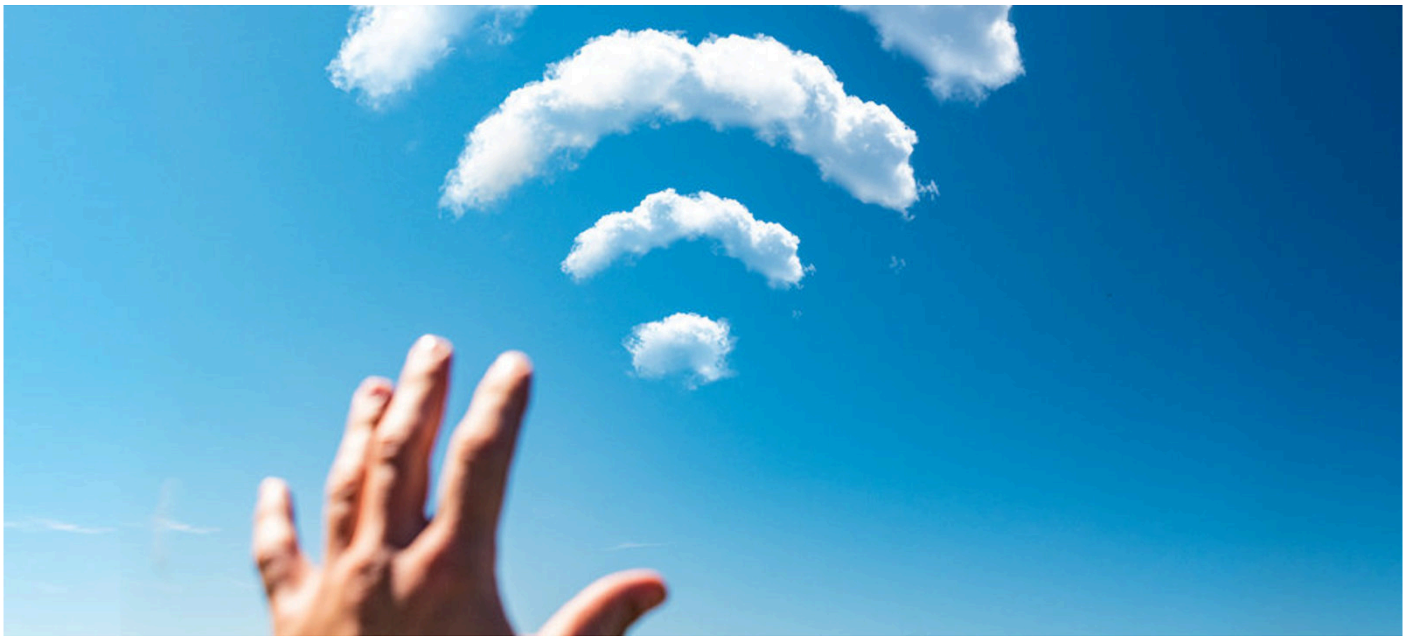
HackMag (https://hackmag.com)

Over-the-air tricks. Simple and effective Wi-Fi pentesting techniques

Date: 04/08/2020 **Author:** Ivan Piskunov (https://hackmag.com/author/g14vano)



In this article, I will demonstrate a few simple and common -although efficient! – Wi-Fi pentesting tricks: hiding your MAC address when you scan a network and attack WPA2, identification of ‘hidden’ networks, bypassing MAC filtering, and jamming access points.

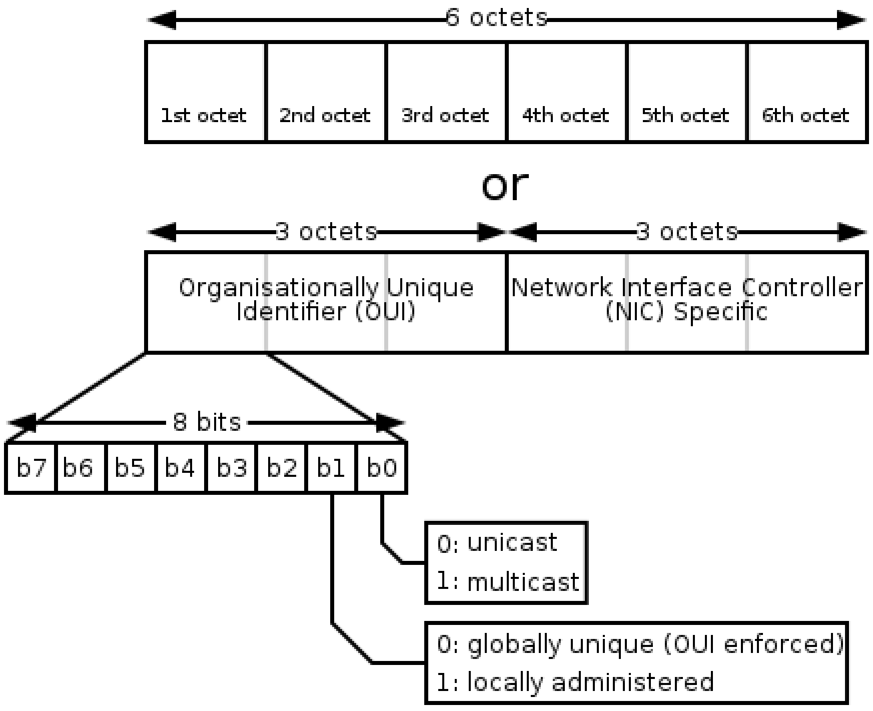
WARNING

All information provided in this material is intended for educational purposes only. The blocking of data transmission and application of techniques described below may by punishable under respective laws. Pentesting cannot be preformed without a written consent from the customer. Remember: the reauthorization data are saved in the router’s log!

Changing and automatically generating a new MAC address at every Wi-Fi connection

The MAC (Media Access Control) address is a unique identifier assigned to each active network unit (e.g. network adapter, router, switch, etc.) or some of their interfaces.

MAC address is embedded into the equipment; its primary purpose is to identify the frame sender and receiver on the network. If a new device connects to the network, the administrator doesn't have to set its MAC address manually.



(https://hackmag.com/wp-content/uploads/2020/08/4855/1.png)

6-octet MAC address

Each network interface has (or at least is supposed to have) a unique MAC address. A device may have several such identifiers; for instance, your laptop has a minimum of two: one for the Ethernet controller and another one for the Wi-Fi adapter. Routers and switches have unique addresses for every port; while in Wi-Fi routers, each wireless interface (e.g. 2.4 GHz, 5 GHz, etc.) has a unique MAC address.

Why change MAC address?

The MAC address uniquely identifies the device and remains unchanged even after the installation of a new OS as it's hardcoded in the network interface microchip.

Pentesters and hackers hide their MAC addresses to prevent the identification of their equipment during the attack. The reason is obvious: if the real MAC address is used, it may be identified during the connection to a network. There are special tools designed to match MAC addresses against geographic coordinates, for instance, the iSniff-GPS script included in Kali.

Practice

I assume that you use Linux. Let's see how your MAC address can be changed without using third-party programs.

Open the terminal and enter the command:

```
$ ifconfig | grep HWaddr
```

If you use Ethernet, type the command below to see the adapters' addresses:

```
$ ifconfig | grep ether
```

To temporarily change your MAC address, you have to disable the respective interface first. For instance, the command below disables the eth1 interface:

```
$ ifconfig eth1 down
```

Now you can generate a new MAC addres:

```
$ ifconfig eth1 hw ether 00:00:00:00:00:11
```

Of course, you may insert any digits in this template.

Now it is time to enable eth1 again.

```
$ ifconfig eth1 up
```

To check whether the changes have come into effect, review the list of MAC addresses again and make sure that the target interface has changed. Note however, that the old MAC address will be restored after the restart.

The optimal variant for a hacker is to change the MAC address with every network connection. Use NetworkManager (https://pkg.kali.org/pkg/network-manager) for that purpose. Starting from version 1.4, this package supports MAC spoofing and many other useful options.

MAC rules are configured separately for the "wired" (ethernet) and "wireless" (wifi) groups.

A wireless adapter can operate in one of the two modes:

- **scanning** – this mode is enabled using the `wifi.scan-rand-mac-address` connection property. By default, its value is `yes`, i.e. an arbitrary MAC address will be set during the scanning. If the value is `no`, this won't happen; and

- **connected to network** – this mode is enabled using the `wifi.cloned-mac-address` property. Its default value is `preserve`.

The following variants are available in the connection mode for the wired interface (`ethernet.cloned-mac-address`) and the wireless interface (`wifi.cloned-mac-address`):

- **explicit MAC address** allows you to spoof a specific MAC address;
- **permanent** uses the factory-assigned MAC address of the device (the default variant);
- **preserve** doesn't change the MAC address of the device upon activation (e.g. if the MAC address was changed by another program, its current value will be used); and
- **random** generates a randomized value upon each connect.

NetworkManager settings are tweaked in the `/etc/NetworkManager/NetworkManager.conf` configuration file. Alternatively, you may add another file with the `.conf` extension (its name does not matter) to the folder `/etc/NetworkManager/conf.d`. I recommend using the latter option because updates usually replace the default config file in NetworkManager, and all changes you have made in it go up in smoke.

Enabling automatic generation of random MAC addresses

If you want to change the MAC address at every new connection but use the same MAC address for connections with the same network, a few strings must be added to the config file:

```
[connection]
ethernet.cloned-mac-address=stable
wifi.cloned-mac-address=stable
```

The properties `ethernet.cloned-mac-address` and `wifi.cloned-mac-address` can be set either separately or together.

To check their values, enter `ip a`; to apply the changes, restart NetworkManager:

```
$ sudo systemctl restart NetworkManager
```

Now it is time to connect to a wireless network and check the MAC address again.

The same addresses will be generated for the same network. However, if you want to generate different addresses every time, use the following settings:

```
[connection]
ethernet.cloned-mac-address=random
wifi.cloned-mac-address=random
```

Spoofing a specific MAC address

Let's say you need a specifid MAC address. First, open `/etc/NetworkManager/conf.d/mac.conf`.

To set a MAC address for a wired interface, add the following strings:

```
[connection]
ethernet.cloned-mac-address=
```

To set a MAC address for a wireless connection, type:

```
[connection]
wifi.cloned-mac-address=
```

Just in case: instead of , you have to enter the required MAC address. And of course, you can configure the wired and wireless connection settings at once.

Important: if you use this method, the MAC address will change only after the connection to a network. Prior to this, the interfaces will have their original addresses. The only exception is Wi-Fi – provided that you have already configured spoofing as shown above. To disable spoofing, add the following strings to the config:

```
[device]
wifi.scan-rand-mac-address=no
```

The changes will come into effect after the restart of NetworkManager.

Other software changing MAC addresses

NetworkManager is not the only program able to change MAC addresses. In fact, many third-party tools and system services can do so. To monitor the results, change the NetworkManager settings:

```
[device]
wifi.scan-rand-mac-address=no
```

Now it won’t spoof the MAC address during the wireless network scanning.

The parameters `ethernet.cloned-mac-address` and `wifi.cloned-mac-address` were not set in the NetworkManager settings; accordingly, the default value (`preserve`) will be used even if the MAC address has been changed by other programs.

The examples below have been implemented in Kali Linux, including tweaking the Wi-Fi adapter settings. A key feature of these techniques is that the changes are lost after a system restart or adapter reconnection.

Changing MAC address with iproute2

I will use the `ip` program included in the iproute2 package. First, I check the current MAC address:

```
$ ip link show
```

The output shows the MAC address after the words `link/ether`. It is necessary to disable the respective interface (in my case, it's wlan0).

```
$ sudo ip link set dev wlan0 down
```

Time to spoof the MAC address. You may set any value you like but remember: the network may be configured to assign addresses only to devices from well-known manufacturers whose MAC addresses meet certain criteria. Therefore, I recommend taking a well-known prefix as the first three bytes and making changes only in the last three bytes.

To change the MAC address, enter the command:

```
$ sudo ip link set dev  address
```

Put your own values instead of the and .

The last step is to switch the interface back to the 'up' mode:

```
$ sudo ip link set dev  up
```

To check the changes, type:

```
$ ip link show
```

The `link/ether` value should match the one you have just set.

Changing MAC address with macchanger

Another variant is the `macchanger` utility. It allows to create MAC addresses resembling those used by certain manufacturers, as well as completely randomized ones. The tool is included in Kali.

Similar to other techniques, the device must not be in use when you change the MAC address; therefore, disable it first:

```
$ sudo ip link set dev  down
```

In my case, the interface is wlan0; replace it with your variant.

To check the MAC values, run the utility with the option `-s`:

```
$ sudo macchanger -s wlan0
```

The output shows the current MAC address, the one hardcoded in the device (in case they are different), and the vendor's name. For instance:

```
Current MAC: 00:c0:ca:96:cf:cb (ALFA, INC.)
Permanent MAC: 00:c0:ca:96:cf:cb (ALFA, INC.)
```

The option `-r` allows to set a random MAC address:

```
$ sudo macchanger -r wlan0
```

The new address will be added to the two output strings shown above.

The option `-e` allows to randomize the MAC address without changing the first three bytes (i.e. manufacturer’s prefix):

```
$ sudo macchanger -e wlan0
```

Finally, if you want to set a new MAC address manually, use the `-m` option:

```
$ sudo macchanger -m  wlan0
```

Don’t forget to put the required address instead of ` `.

Detecting hidden SSID

As an additional security precaution, some hotspot owners configure their hotspots so that they don’t transmit their names (ESSID). Users don’t see such a network on the list of available networks; to connect to it, you have to enter its name manually.

Too bad, this measure doesn’t guarantee security: in some situations, ESSID is still openly transmitted.

Detecting hidden SSID with Airodump-ng

You can intercept an ESSID over-the-air when a client connects to a hidden network. To do so, you should either wait until this happens ‘naturally’ or expedite the process by disconnecting everybody from the access point. This is called deauthentication. The disconnected clients will start reconnecting automatically, and the network will openly broadcast its name.

First, launch airodump:

```
$ airodump-ng
```

When the program detects a new network, you’ll see its BSSID, name length, and the channel it uses. For instance, if the network uses the first channel, specify it as follows:

```
$ airodump-ng wlan0 --channel 1
```

Similar to a handshake interception, you may use the key `-w` followed by the file name prefix. The handshake interception does not prevent the detection of a hidden access point. Then you can either wait until a new user connects to it or deauthenticate all its clients:

```
$ aireplay-ng -0 3 -a  wlan0
```

In this string, `-0` means a mass deauthentication, while 3 is the number of the sent packets.

The result is nearly instant: you will see a string containing the full name of the access point.

Bypassing MAC filtering by using a whitelisted address

Launch the oldie-goodie Airodump-ng. Switch the adapter to the monitoring mode and enter the following commands:

```
$ ifconfig wlan0 down && iwconfig wlan0 mode monitor && ifconfig wlan0 up
$ airodump-ng wlan0
```

You will see the list of networks, numbers of clients connected to them, and their MAC addresses. Any of these ‘legitimate’ addresses can be assigned to your adapter if the target network uses whitelist filtering.

Sometimes, the lists of clients connected to certain access points are not displayed immediately because the program needs time to collect the required information. In such situations, use deauthentication. If the hotspot has at least one client, you will see this after the reconnection. Concurrently, you can intercept the handshakes.

To deauthenticate the clients, terminate Airodump-ng and launch it again specifying the channel used by the target access point.

```
$ airodump-ng wlan0 --channel 1
```

Then send deauthentication packets and see what happens:

```
$ aireplay-ng -0 5 -a  wlan0
```

The deauthentication reveals some of the previously unknown clients. Copy the MAC address of a legitimate client, adjust the settings of your network card – and you are free to launch your carefully planned pentesting attack!

Jamming Wi-Fi networks

Sometimes, pentesting requires to jam a certain access point. I suggest using the LANs (https://github.com/DanMcInerney/LANs.py) utility for that purpose. Not only can it jam Wi-Fi, but also spy on the users and individually retrieve data (including IP and MAC addresses!) contained in ARP (Address Resolution Protocol) tables of the target machine, router, and, if necessary, DNS server.

The jamming range depends on the adapter power; script settings enable you to jam everybody or just one client. All you have to do is download the script, which subsequently downloads required components from repositories and installs them (you’ll be notified is something goes wrong). Finally, the script scans the Aether and collects information on all wireless connections.

```
$ sudo apt install -y python-nfqueue python-scapy python-twisted nbtscan
$ git clone https://github.com/DanMcInerney/LANs.py.git
$ cd LANs.py/
```

Now launch the script to start jamming.

```
$ python lans.py -u -p
```

The `-u` and `-p` keys enable active identification of targets for ARP spoofing and display all important unencrypted data sent or requested by these targets. There is no `-ip` option here; therefore, an ARP scanning of the network is performed, and its results are compared with results collected in the live promiscuous capture mode. In the end, you get the full list of network's clients.

After building the network map and reviewing lists of connected clients, you can press Ctrl + C to stop the search. By the way, you may use Nmap for this purpose as well.

A targeted jamming attack looks as follows:

```
$ python lans.py --jam --accesspoint  -s
```

where:

- **–jam** instructs to jam all or some wireless APs operating at 2.4 HGz and clients within the range; if necessary, additional arguments can be used (see below);
- **-s** specifies a MAC address to skip deauthentication; and
- **–accesspoint** specifies the MAC address of a specific target AP.

Jamming all networks:

```
$ python lans.py --jam
```

Jamming a single access point:

```
$ python lans.py --jam --accesspoint
```

Here you can also specify some additional options:

- **-ch** limits jamming to a single channel;
- **–directedonly** instructs not to send deauthentication packets to the broadcast addresses of APs and only send them to client/AP pairs;
- **–accesspoint** sets a specific AP as a target.

Another efficient Wi-Fi jamming script

The wifijammer (https://github.com/DanMcInerney/wifijammer) utility can also be used to jam Wi-Fi. It is very easy-to-use: without parameters, it will jam everything identified within the adapter's range. To avoid 'friendly fire', you may exclude some MAC addresses using the `-s` option.

Installing wifijammer:

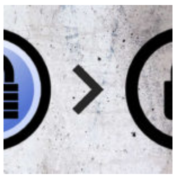
```
$ git clone https://github.com/DanMcInerney/wifijammer.git
$ cd wifijammer/
$ sudo python2 wifijammer.py --help
```

Launching it:

```
$ sudo python2 wifijammer.py -s
```

That’s it. Hopefully, the above techniques would be useful in your pentesting endeavors. Good luck!

Related posts:



2022.06.03 — **Challenge the Keemaker! How to bypass antiviruses and inject shellcode into KeePass memory (https://hackmag.com/coding/keethief)**

Recently, I was involved with a challenging pentesting project. Using the KeeThief utility from GhostPack, I tried to extract the master password for the open-source KeePass database…

Full article → (https://hackmag.com/coding/keethief)

2023.04.20 — **Sad Guard. Identifying and exploiting vulnerability in AdGuard driver for Windows (https://hackmag.com/security/aguard-cve)**

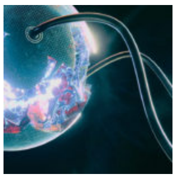
Last year, I discovered a binary bug in the AdGuard driver. Its ID in the National Vulnerability Database is CVE-2022-45770. I was disassembling the ad blocker and found…

Full article → (https://hackmag.com/security/aguard-cve)

2023.02.13 — **First Contact: Attacks on Google Pay, Samsung Pay, and Apple Pay (https://hackmag.com/security/pay-systems-attacks)**

Electronic wallets, such as Google Pay, Samsung Pay, and Apple Pay, are considered the most advanced and secure payment tools. However, these systems are also…

Full article → (https://hackmag.com/security/pay-systems-attacks)



2023.02.13 — **Ethernet Abyss. Network pentesting at the data link layer (https://hackmag.com/security/ethernet-abyss)**

When you attack a network at the data link layer, you can 'leapfrog' over all protection mechanisms set at higher levels. This article will walk…

Full article → (https://hackmag.com/security/ethernet-abyss)

2022.01.11 — **Persistence cheatsheet. How to establish persistence on the target host and detect a compromise of your own system (https://hackmag.com/security/persistence-cheatsheet)**

Once you have got a shell on the target host, the first thing you have to do is make your presence in the system 'persistent'. In many real-life situations,…

Full article → (https://hackmag.com/security/persistence-cheatsheet)

2022.06.03 — **Vulnerable Java. Hacking Java bytecode encryption (https://hackmag.com/security/java-bytecode-encryption)**

Java code is not as simple as it seems. At first glance, hacking a Java app looks like an easy task due to a large number of available…

Full article → (https://hackmag.com/security/java-bytecode-encryption)

2022.01.13 — **[Step by Step. Automating multistep attacks in Burp Suite (https://hackmag.com/security/burp-stepper-intruder)]**

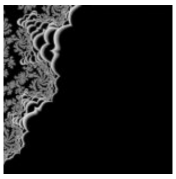
When you attack a web app, you sometimes have to perform a certain sequence of actions multiple times (e.g. brute-force a password or the second authentication factor, repeatedly…

Full article → (https://hackmag.com/security/burp-stepper-intruder)

2022.01.12 — **First contact. Attacks against contactless cards (https://hackmag.com/security/credit-cards-nfc)**

Contactless payment cards are very convenient: you just tap the terminal with your card, and a few seconds later, your phone rings indicating that…

Full article → (https://hackmag.com/security/credit-cards-nfc)



2023.02.12 — **Gateway Bleeding. Pentesting FHRP systems and hijacking network traffic (https://hackmag.com/security/gateway-bleeding)**

There are many ways to increase fault tolerance and reliability of corporate networks. Among other things, First Hop Redundancy Protocols (FHRP) are used for this…

Full article → (https://hackmag.com/security/gateway-bleeding)

2023.07.07 — **VERY bad flash drive. BadUSB attack in detail (https://hackmag.com/security/very-bad-usb)**

BadUSB attacks are efficient and deadly. This article explains how to deliver such an attack, describes in detail the preparation of a malicious flash drive required for it,…

Full article → (https://hackmag.com/security/very-bad-usb)