

# Priori-Fi

Marcos Castrillo Sarmiento

Final report: 21/06/2017

# INDEX

1. Introduction	.....	3
2. Main goals	.....	4
3. User interface	.....	5
4. Development	.....	7
4.1. First programming steps:	.....	7
4.2. Programming	.....	8
5. Conclusion	.....	10

# 1. Introduction:

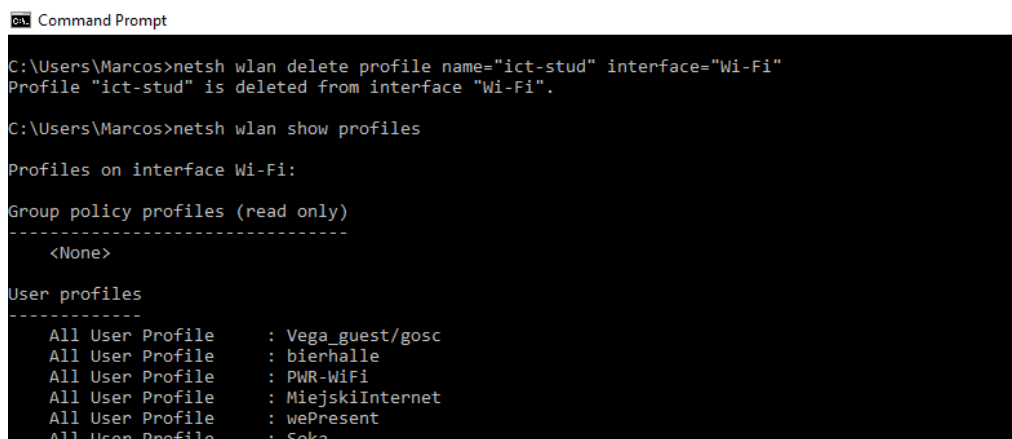
A priori, configure the priority of Wi-Fi networks could seem an irrelevant issue. Although, it's not an option which should be ignored. Enterprises and universities usually have more than one Wi-Fi available network, and those members who need to work using a specific network **should be able to do it easily**, not manually every time they use their computer.

A quick example of how insecure this is: a university teacher has saved two Wi-Fi networks on his laptop, the exclusive one for teachers the university and an open Wi-Fi created by some students of the faculty. His computer connects him to the open one and he doesn't notice about it.

He uses his credentials to log into his university's system, which is not using https. By chance, these students are sniffing all the traffic in the network, being able to steal his credentials and access to the university system as the teacher. Shouldn't this be avoided?

Since the release of Microsoft Windows 8, Windows operative system has a **lack of an appropriate way to manage Wi-Fi priorities**. It happens in Windows 10 as well.

The only way of configure them is using the command prompt (*see [Image 1](#) below*).



```
Command Prompt
C:\Users\Marcos>netsh wlan delete profile name="ict-stud" interface="Wi-Fi"
Profile "ict-stud" is deleted from interface "Wi-Fi".

C:\Users\Marcos>netsh wlan show profiles

Profiles on interface Wi-Fi:

Group policy profiles (read only)
-----
    <None>

User profiles
-----
    All User Profile : Vega_guest/gosc
    All User Profile : bierhalle
    All User Profile : PWR-WiFi
    All User Profile : MiejskiInternet
    All User Profile : wePresent
    All User Profile : Soka
```

*Image 1: Managing a network with CMD*

This method what has some disadvantages:

- Poor user interface.
- Average Windows user doesn't know how to use it.
- Not user-friendly at all.

The scope of this project is to create a software to fulfil this lack of user interface.

## 2. Main goals:

This project has a main goal - to create a simple but useful and user-friendly interface to manage Wi-Fi networks on Windows 8 and Windows 10.

The next **options** are available on the program:

- List of available saved networks.
- List of available networks.
- List of saved networks.
- Show the SSID.
- Show the type of Wireless Security.
- Show the signal strength
- Configure the signal threshold.
- Show the signal speed.
- Move up and down the priority of the network.
- Refresh the list of networks.
- Connect to any kind of Wi-Fi.
- Forget Wi-Fi networks.







### 3. User interface:

The following options will be displayed on the main menu:

- At the left of the window, a list of all the saved Wi-fi networks available at the moment, as well as the networks the user is connected to, if any (see [Image 2 below](#)).

The features of each saved network are described below:

- **Wi-Fi:** SSID (name) of the network.
- **Signal strength:** Percentage of signal received at the moment.
- **Security:** Type of wireless security used by the network (open, WEP, WPA, 802.1x EAP, etc.).
- **Signal threshold:** Percentage of minimal signal received needed to connect to the network. It allows to avoid networks with low signal strength.
- **Priority:** Criteria to connect to a network: it will try to connect to the networks with higher priority first. The highest priority is 1, the lowest is 10.

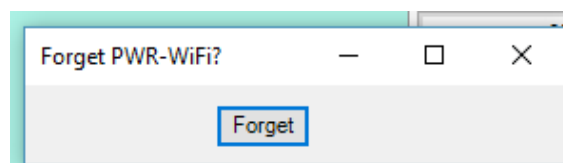
Wi-Fi	Strength	Security	Threshold	Priority
PWR-WiFi	82% 	Open	0 	1 
wePresent	96% 	Open	0 	10 

*Image 2: Example of list of available saved networks*

If the computer is connected to any Wi-Fi, it appears at the top of the list, with blue name.

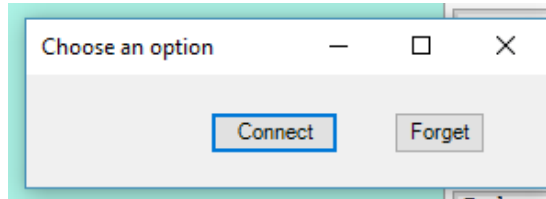
When the user clicks on the name of the Wi-fi, there are two possibilities:

- **Connected network:** If it's the network the computer is connected to, a window to forget the network is showed (see [Image 3 below](#)).



*Image 3: Forget Wi-Fi window*

- **Saved network:** If it's a network known by the computer, a window to connect/forget the network is showed (see [Image 4](#) below).



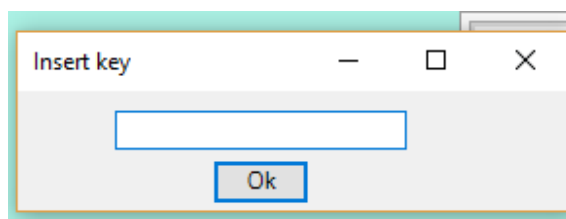
*Image 4: Connect to/forget Wi-Fi window*

- At the right of the window, a list of all the available Wi-fi networks available at the moment, if any (see [Image 5](#) below).

Wi-Fi	Strength	Security
eduroam	90%	WPA2
Y550-L01 6233	60%	WPA2
xboxson	54%	WPA2
	68%	WPA2
wePresent007	34%	WPA2
Orange Internet FB557C	56%	WPA2
Laboratorium Tyfloinformat	4% .	WPA2

*Image 5: Example of list of available networks*

When the user clicks on the name of the Wi-fi, a window to connect to the network is showed, if needed (see [Image 6](#) below). The type of security of the network is detected so if it's needed, it asks for the proper key.



*Image 6: Connect to Wi-Fi window*

## 4. Development:

### 4.1. First steps: programming language

The first task in order to start developing the software was to select a proper programming language. After a research, the chosen language was **C++**, as it's a general-purpose programming language, better than C.

Then I had to choose an IDE and a compiler. The IDE chosen was **Code::Blocks**, because it is very light, user-friendly and I had previously worked with it. I used **MinGW** as compiler.

I started to take the first steps programming the User Interface in C++, and then I realize it wasn't a good idea due to it's a tedious work. After more investigation on network programming, I found out that my best option was **C#**.

Also, I noticed that Code::Blocks is too simple to develop this kind of software. The best option was to choose **Microsoft Visual Studio** with Microsoft .NET Framework.

Then I found a .NET class library which allows you to control Wi-Fi (802.11) network adapters installed in your Windows machine. This library uses the Native Wi-Fi API, available since Windows Vista and Windows XP SP2. Older versions of Windows are not supported, but as the goal of the project is Windows 8/10, it's perfect to work on it.

In order to implement the user interface of the program, I used **Windows Forms**. It's an easy way to personalize it, with a lot of features to customize the appearance as much as you want.

## 4.2. Programming

As soon as I started programming I realised I had a lot of work to do. It was my first time developing a program for Windows, as well as the first time working with Wi-Fi profiles.

The first task was to get a list of all the networks and store them somehow. I created different arrays of Wi-Fis for the different types of networks (available, connected, saved) in order to work with them.

I started saving all of them in a XLS document, and accessing to the file like if I was using Excel. This method turned out to be hard and not elegant at all.

The solution was to use a XML file instead. I created a **database** of the thresholds and priorities set by the user, so they are still there when the program is used again. This document is generated automatically if it doesn't exist when the user launches the program. (see [Image 7](#) below).

```
<?xml version="1.0" encoding="utf-8"?>
<Thresholds>
  <TP-LINK_1234>20</TP-LINK_1234>
  <PWR-WiFi>80</PWR-WiFi>
</Thresholds>
```

*Image 7: Example of thresholds.xml*

Using the Native Wi-Fi API, it wasn't too difficult to get the information I needed about the networks. **Displaying** them in a proper way was harder, because Windows Forms turned out not to be the best and most simple way to do it. (see [Image 8](#) below).

Form1

SSID			
PWR-WiFi	%		<input type="checkbox"/>
eduroam	%		<input type="checkbox"/>

SSID	Signal strength	Wireless security
PWR-WiFi	96%	Open
wePresent	100%	Open
PWR-WiFi	90%	Open
Wrota Baldura	38%	WPA2_PSK
eduroam	86%	802.1x EAP
xboxson	38%	WPA2_PSK
HUAWEI P8 lite	40%	WPA2_PSK
wePresent	100%	Open
	86%	WPA2_PSK
	86%	WPA2_PSK
wePresent007	36%	WPA2_PSK
AndroidAP	38%	WPA2_PSK
TP-LINK_F7BDD6	26%	WPA2_PSK

*Image 8: First view of the main menu*



Obviously, the information has to be continuously **refreshed**. At the beginning, I thought about implementing a button to refresh it, but doing it automatically every few seconds seemed a better choice. I had some problems with the refreshing, because it was slow and the user had to see how it was refreshed during some seconds, making the UI way worse.

I fixed it duplicating both panels and refreshing just the panel I wasn't showing, switching between them every few seconds. This reduced the blinking effect on the panels but not totally removed it. A function to stop refreshing it if the user is interacting with the program (writing a key, setting the priority...) was added as well.

At this point, most of the functionality of the program was working correctly, but it had a lot of bugs. As the program requires text inputs from the user, an inappropriate or not expected use of it (leaving it empty, being too short or too long...) could break it, so I had to implement some additional functions to prevent these errors.

After a lot of debugging and tests, the program was working perfectly.

Windows Forms doesn't have an appealing UI, so the remaining tasks were cosmetic. I gave the program a new and better look changing the fonts, background, adding a signal strength indicator, titles for the panels, etc. (see [Image 9](#) below).

CONNECTED AND SAVED NETWORKS					AVAILABLE NETWORKS		
Wi-Fi	Strength	Security	Threshold	Priority	Wi-Fi	Strength	Security
Vega guest/gosc	80%	WPA2	0	1	Vega	48%	WPA2
MiejskiInternet	34%	Open	0	2	neostrada cd8e	46%	WEP
					Czytelnia	42%	WPA2
					SalaTeatralna	40%	WPA2
					NETIASPOT-ECDBD0	32%	WPA2
					Muzeum	58%	WPA2
					Kancelaria	54%	WPA2
					Vega Cam	40%	WPA

*Image 9: Final view of the main menu*

## 5. Conclusion

Working on this project has been a great and productive experience.

The beginning was tough, because I had to learn about Windows Forms, Windows network profiles, C#, Visual Studio. The amount of information I had to look for on the internet was overwhelming. After a lot of work, the program started to show some results, what was rewarding and pushed me to keep working hard.

In summary, I have acquired some skills I wouldn't have if I wouldn't have worked on Priorifi. I also have a program I can keep developing and use as my thesis.