Guifi.net: Security Analysis of a Heterogeneous Community Network

Patricia Ramos García

Supervised by Andreu Bassols Alcón

Information and Communication Technologies Degree

Barcelona, January 2018

Here it is, at last, the so longed-for product of my own.

Dedicated to my North Stars: Napo & Lemmon.

Abstract

Guifi.net is a heterogeneous community network that brings Internet to rural areas or vulnerable groups. This opens the door to many advances, but encompasses some risks as well. The aim of this project is to assess the general security of Guifi.net from tests performed on a key network element: the router. In particular, MikroTik and Ubiquiti are the most used makes in Guifi.net and hence, the target of this project. Basic, yet important, security settings are tested. On the plus side, the widespread use of firewalls and good management of default credentials stand out from the results. However, the tests also frequently reveal outdated firmware and insecure services, which are potential security threats.

Resumen

Guifi.net es una red comunitaria abierta que facilita la llegada de Internet a zonas rurales o personas vulnerables. Este hecho abre la puerta a muchos avances, aunque también comporta riesgos. El objetivo de este proyecto es evaluar la seguridad general de Guifi.net a partir de pruebas realizadas sobre un elemento de red clave: el rúter. En concreto, MikroTik y Ubiquiti son las marcas más populares en Guifi.net y constituyen, por tanto, la clave de estudio de este proyecto. Su configuración básica, aunque no por ello menos importante, es evaluada. La parte positiva de los resultados destaca por un amplio uso de cortafuegos y una buena configuración de usuarios y contraseñas. Sin embargo, el análisis revela una alta tasa de *firmware* desactualizado y de servicios intrínsecamente inseguros abiertos.

Keywords [Open networks, Security, Hardening.]

Acknowledgements

This project is the outcome of my dedication to the study of Telecommunications. I have got here not by chance, but because I have always fought to be someone who followed their aspirations.

Yet, no one can do this by themselves. I could have hardly done anything without the generosity, endless attention, and unconditional help from *MrIknoweverythingsopleasedonotquestionme*, the invisible but invaluable encouragement in multiple ways from such an impressive family, or the constant praising and acknowledgement of my efforts that only a good friend can possibly do. Even those that have not necessarily being helpful, have given me the opportunity to stand strong.

I am particularly grateful to the UOC for being ahead and offer online education in an environment that promotes self-improvement, self-endurance, and self-motivation every step of the way. And I would also like to thank Andreu Bassols, my final project advisor, for its guidance, reliability, and readiness to help.

Last but not least, the reader, either intentional or casual, thanks for stopping by.

Table of Contents

Aim of this memoir

Guifi.net: Security Analysis of a Heterogeneous Community Network has been developed with people at its core. The Internet has been proven tremendously beneficial in terms of economic, social, political, and technological progress. Nonetheless, these improvements do not necessarily mean security and inclusion for everyone. Vital elements such as fair and free access to broadband, to the resources of the Internet without discrimination, and a survival knowledge level to surf the Net with a minimum of security are far to be guaranteed for everyone.

This project focuses on key security aspects and gives extensive details on what to do to harden a device, following the actual configuration of routers in Guifi.net. Given that it is a task everyone joining Guifi.net has to do, no matter what their background is, it actually partially covers the need for digital education. Those considering to join, or already part of Guifi.net, may benefit the most from this project.

How to read this memoir

This memoir is the tangible result of a end-of-degree project. It is meant to put into practice many of the skills learnt during the degree. Due to its educational and practical component, the interested student may find particularly helpful the content related to the Project's Management. From the Planning to the Closing stages, all are covered and explained in detail in the Appendices.

Every effort has been made to render this memoir both appealing, rigorous, and self-contained. Hence, the reader looking for an introduction to Guifi.net may read Chapter 2. This can lead to further interest on the actual project's goal, which is assessing the network given a paramount player in a network: the router. If that is the case, in Chapter 1 the reader will find the motivations for this work as much as its objectives.

Those looking for a straightforward approach to the core objectives specifically related to Guifi.net might jump to Chapter 3, followed by Chapter 4. There, MikroTik routers are tested and the results plotted and then analysed. The avid reader may want to reproduce the results, improve them, or further develop them. The scripts can be found in the Appendices.

List of Acronyms

AP	Access Point
APC	Association for Progressive
AS	Communications
ASN	Autonomous System
BGP	Autonomous System Number
CAN	Border Gateway Protocol
CLI	Community Access Network
CN	Command Line Interface
CHR	Community Network
CPE	Cloud Hosted Router
DD	Customer Premises Equipment
DIY	Digital Divide
DMZ	Do It Yourself
DNS	Demilitarised Zone
EFF	Domain Name System
EU	Electronic Frontier Foundation
FCC	European Union
GUI	Federal Communications Commission
HCN	Graphic User Interface
HTTPS	Heterogeneous Community Network
IDL	Hypertext Transfer Protocol Secure
IP	Internet Defense League
IPsec	Internet Protocol
IS	Internet Protocol Secure
ISP	Information Society
IT	Internet Service Provider
IXP	Information Technology Internet Exchange Provider
MIB	Management Information Base
NAT	Network Address Translation
NFV	Network Function Virtualization
NHS	National Health System (UK)
NMAP	Network Mapping
OAN	Open Access Network
OID	Object IDentifier
OS	Operating System
OSPF	Open Shortest Path First
P2P	Peer to Peer
SCP	Secure Copy
SDN	Software Defined Network
SFTP	Simple File Transfer Protocol
SMB	Server Message Block protocol
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SSL	Secure Socket Layer
STEM	Science, Tech, Engineering, and Maths
UN	United Nations
UK	United Kingdom
US	United States
VoIP	Voice over IP
VPN	Virtual Private Network
WCN	Wireless Community Network
WDS	Wireless Distribution System
WPA2	WiFi Protected Access (version 2)

1. Introduction

"From its earliest beginnings, the Internet evolved from a set of fundamental principles based on openness, inclusivity, collaboration, and transparency. While its original premise was the voluntary exchange of data across a network of networks, its social, technological, economic and political impact has been profound."

—Internet Society

States the motivations and aims of this work. To do so, it takes into account the current socio-political circumstances. The last section introduces the reader to the contents of the memoir.

1.1. Project scope

Guifi.net: Security Analysis of a Heterogeneous Community Network is a project idea that arises amid the current unforeseeable worldwide geopolitical order, the ubiquity of networks, and the need to bring access to as many people as possible, bridging the Digital Divide [1, 11, 12].

Community networks such as Guifi.net try to be fair and true to the original purpose of the web (and the Internet itself) as Sir Timothy Berners-Lee envisioned it: "*...open and free, [...] where inequities in other parts of society don't play out*" [2]. Precisely, Guifi.net sets itself aside most of networks by being free, open and neutral [3, 4].

However, Guifi.net Foundation ideal of openness and neutrality is at stake as its packet flow is susceptible of being controlled—denied, filtered, or recorded—by external and powerful entities [5]. It is sensible to also consider any possible threat associated with outdated software or poorly deployed security policies. Both are clearly posing vulnerabilities on consumers and citizens.

Specifically, this project evaluates the security of Guifi.net. It is addressed both technically, by testing and analysing key network elements.

1.2. Project objectives

GENERAL OBJECTIVES. Since the Internet came up, back in the 80s, it has been growing steadily to the point where more than half the population is connected to it [6]. Being able to surf the net has a very large economic and social impact. That is the reason why the European Union (EU) has included initiatives in the *Horizon 2020* to reduce the Digital Divide [7] that currently affects 20% of its population [8, 68].

Thanks to community networks like Guifi.net in Catalunya, grey zones—mainly in rural areas—have broadband coverage, and low income people may gain access to it [11]. Furthermore, open networks offer an alternative for citizens that consider the regular Internet Service Providers (ISP) far too expensive. Besides, the latter are becoming a monopoly, posing a serious risk on network neutrality and openness [4].

On top of that, the fast speed at which hackers are evolving leave no other option to many countries than to take action and proactively work to prevent future attacks on large, governmental networks [9]. However, the attacks are not limited to the latter, but do also fiercely affect individual users, often taking advantage of their poor digital education [48, 49].

This environment is especially hostile to vulnerable groups on which discrimination and oppression can be easily inflicted [10]. Moreover, networks are steadily transforming into a more controlled (and unfair) tools, by the application of intrusive control policies.

Those are the grounds on which *Guifi.net: Security Analysis of a Heterogeneous Community Network* is based. Its general objectives are two:

I. An analysis of the weaknesses of the software of Guifi.net, by examining its potential vulnerabilities on key network components such as routers or proxies [13, 14].
II. As a result of the previous analysis, an evaluation of the ways Guifi.net attains (or could improve) fundamental security standards.

SPECIFIC OBJECTIVES. A more in-depth look at the general objectives is given in what follows. They ultimately define the structure and contents of this memoir.

I. **Thorough introduction of Guifi.net.** Guifi.net description as an Open, Neutral and Free network, considering its role for society as a community network and its heterogeneity and diversity.

II. **Testing software**. In order to perform certain network functions, such as routing or accessing services through proxies, Guifi.net employs software. It is the way modern networks are switching to Software Defined Networks (SDN) and Network Function Virtualization (NFV).
In particular, Guifi.net employs ***AirOS*** operating system (OS) for most of its Ubiquiti routers (60%) [14]. As another option, with many users as well (30%) [14], there are MikroTik routers, with **RouterOS** software.

III. **Standard security evaluation**. Results extrapolation. Given such a heterogeneous network—many routing protocols, multiple manufacturers, many types of devices and connections, etc.—this is limited to the assessment of specific elements that are prevalent on the network: routers MikroTik and Ubiquiti.

1.3. Technology in use: advocating for open software

To develop this project, a series of devices, software, and general tools have been used (*Table 1*). Most of them will be highlighted on the text every time they are specifically used. One thing they all have in common is that they are free and open software.

This is an advantage as it reduces project costs and allow programmers to make improvements—crowd-sourced. An important consequence is that we get stronger code—peer-reviewed—and, hence, contribute to reduce its vulnerabilities [54].

System software	Linux (Ubuntu 16.04) RouterOS AirOS	
Application / Firmware Software	PuTTy (SSH connections) MobaXterm (Windows) Secure Shell (SSH) Virtual Private Network (VPN): OpenVPN	CHR OpenVPN SCP [69] FileZilla (SFTP) Virtualbox
Testing software	NMAP (Network Mapping) SNMP Other terminal utilities (Unix)	
Computer languages	Unix Shell Script	
Nodes Guifi.net	UOC (server and supernode) Other network devices at reach	
Production software	Google Drive LibreOffice Nano/Gedit Sharelatex Bubbl.us	

Table 1. Open software employed throughout the project development.

1.4. Outline

The vulnerabilities caused by reckless users help determining whether Guifi.net is secured, from a simple perspective. Chapter 2 introduces the reader to Guifi.net foundations and guides them through to its main characteristics and the potential threats it may face. This leads to Chapter 3, in which Guifi.net is technically addressed to determine whether it achieves a basic security level. Following these results, Chapter 4 focuses on their analysis and a Guifi.net security discussion is made that considers the users and their common practices. Finally, the conclusions, plus benefits from the project, and its future improvements are all covered in Chapter 5.

2. Guifi.net: a heterogeneous community network

"There are few things more powerful than people united."

—Vironika Tugaleva

Guifi.net is described from different angles: both as a community and as a physical network; even as an indispensable tool for people to thrive as part of our society. In addition, its topology, reasons for growth, and potential threats are addressed.

2.1. Guifi.net background

Guifi.net rapidly evolved from an ad-hoc solution into a proper private network. Later on, it received its own Autonomous System Number (ASN), which led to the appearance of fair ISPs. Guifi.net is the largest Heterogeneous Community Network (HCN) in the world, and it is characterised by its openness, free access, and neutrality. It also brings Information Society (IS) and many other benefits to its customers.

The beginnings: the need for broadband

By 2004 many rural areas of Catalunya did not have access to broadband Internet. And even though it was widely available in the cities—mostly offered by Telefónica—, non-lucrative areas were disconnected. This is the context in which Guifi.net is brought about: first, as a personal favour between individuals, consisting in installing wireless devices (*Figures 1, 2*) to establish a connection from a neighbouring area to a rural settlement [19]; later on, as a means of accessing the Internet freely to anyone.

> ***"The origins of this community were technically oriented: from an Internet access-point they designed and developed a mesh network that was able to interconnect different towns in the northern Catalonia through radio waves all included in a few Internet access-node points."*** [25]

Figures 1 and 2. Installation of antennas for Guifi.net [77, 79].

Case study: La Garrotxa

Garrotxa is a rural county in the Pyrenees. Around 2010,town officials and Guifi.net were discussing a possible collaboration, as the former wanted to deploy fibre and bridge the DD within a year. They did not reach to an agreement at first, and five years later in La Garrotxa most of its citizens were not enjoying the benefits of the fibre yet. Luckily, an agreement was eventually reached for Guifi.net to work for a whole year in the town. By 2017 the collaboration is still ongoing and expectations are that by the end of 2018 the whole county will be covered with fibre [79].

Rapid evolution: from private network to autonomous system

In spite of the initial challenges, Guifi.net turned out to be a breakthrough, especially in Catalunya. That was mainly because it helped bridge the Digital Divide (DD), acting as a positive social reinforcement. Guifi.net became a social enhancer for low-income families, dissatisfied customers, and low-density populations.

After some years, *La Generalitat*, city councils and organisations alike have allowed the spread of Guifi.net [19] and contributed to it by offering devices to work as servers for Guifi.net, or even public spaces to place new nodes, hence, allowing people to connect freely to the network [24, 25]. Amid this explosion of nodes arising all over the Catalan domains, several ISPs came into existence after the Guifi.net Foundation was accepted as an Autonomous System (AS) in 2009 (*Figure 3*). From that moment onwards, the Foundation could allow others to work as ISPs to grant Internet access and deliver additional services at a fair cost [21]. By 2017, the number of ISPs was over twenty [79]. As an example, Goufone offers wireless or fibre Internet connections, telephone service with Voice over IP (VoIP), and surveillance, among other services [22].

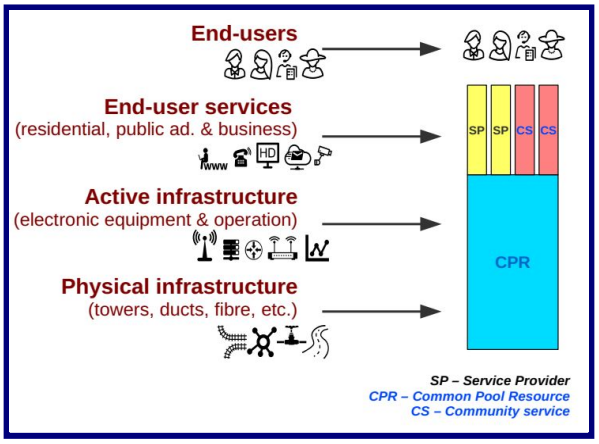


Figure 3. Business model of Guifi.net next to the network stack infrastructure and services [24].

A heterogeneous network: free, open, and neutral

Guifi.net is considered a commons for all its users. That is, the deployment of the network is made thinking about maximising the benefits of technology for the people. Since the beginning, the commons agreement makes it clear that technology is there to achieve the ultimate goal of connecting people [19, 20, 24]. Hence, every stakeholder must comply to keep Guifi.net open, free, and neutral (*Table 2*) [24]. It is open only if *everything* is accessible to everyone, without discrimination (*Figure 4*, [78]). It may be free as long as no one has to pay to be part of the network, or the access is not conditioned to a specific set of hardware and/or software. And neutral, if no restrictions are placed on the type of connection, content, location, time of day, or infrastructure available for communications [50, 75].

guifi.net logo. The text in Catalan reads, "Cut the ties that bind you. A free network for all is possible."

Free. Everyone can benefit from their right to access the Internet—as a human right, without discrimination.	**Open.** Knowledge and network access is public and anyone interested can join. This benefits both the network management and growth, avoiding technological dependencies.	**Neutral.** All the network infrastructure is uncontrolled and undetermined (both software and hardware), and so are its contents and services as well.	**Jointly managed.** Several people take part voluntarily on the network as to install, maintain, and operate its services. Everyone, is expected to approach it as a commons.

Table 2. Guifi.net main characteristics [20].

These traits make it possible to refer to Guifi.net in multiple ways. It is one of the first international Open Access Networks (OAN), as it is free for anyone who would like to join; a Community Access Network (CAN), since it is deployed, managed, operated, and enjoyed by the users themselves; a Wireless Community Network (WCN) [4], because it began relying solely on WiFi connections; or as a HCN, to evidence not only its dependence on people but also the progressive inclusion of fibre as a more reliable, and faster technology [14]. In addition, the term heterogeneous can also refer to the unlimited options for connecting to Guifi.net, both for software and hardware.

Impact on people: Information Society available to all

As a community network, Guifi.net provides knowledge and information to the public. Allowing citizens to access a wide range of resources, such as social networking, web browsing, or mailing, just to name a few, helps to overcome the DD. This means that everyone would be able to discuss topics that matter to them, organise themselves, and share knowledge with others.

> "***Once we have computer outlets on every home, each of them hooked up to enormous libraries and you ask, [...] find out, [...] can follow it up, [...] from your own home, at your own speed, in your own direction, [and] in your own time [...], then everyone will enjoy learning***."
> —Isaac Asimov, 1988.

Naturally, this opens the door to digital education; understood both as online instruction and Information Technology (IT) literacy. Community Networks (CNs) play a key role in this respect not only because they are created with people in mind, but because they offer education resources themselves such as wikis, open network statistics, a whole community of volunteers, multimedia resources, [27, 28, 29] etc.

Guifi.net is neither the only community network nor the first to come up in Spain. Another example is TINET (Tarragona Internet, 2005). It pioneered the technological field by offering a public Internet service, as well as leading the appearance of community networks in Europe. The reason for its foundation was the conviction that the IS should be available to everyone. In fact, the United Nations declared the Internet a human right in 2011 [30].

Its design: users first

This kind of network is structured from the bottom up; that is, the Internet access is offered to the public and, only then, it is expected the market reacts to it. Closed networks work the other way around, as private benefits are stressed above all. While the latter are typically centralised and hierarchical, open networks offer a decentralised and horizontal management [4, 24]. In what follows we will have a more in-depth look into this subject.

2.2. Guifi.net technological overview and topology

TCP-IP network: from radio waves to fibre

Guifi.net uses radio technology to connect devices. Since 2009, fibre connections have been gradually deployed, especially for the backbone links [13, 19]. Fibre is by far faster and more secure [31, 42]. Guifi.net meshed topology [15] is built up with Wireless Distribution System (WDS) and dedicated Point to Point (P2P) links. The former are radio links that extend the WiFi networks and the latter make up the backbone. Altogether, they allow the routing of packets from Guifi.net communications.

Hosts on Guifi.net communicate thanks to IPv4 (Internet Protocol v4) being deployed on the network. Given its potential growth, this private network uses the non-publicly routable addresses:

10.0.0.0/8. As a result, a Network Address Translation (NAT) service is needed at every gateway leading to the Internet.

Centralised management: Internet routing and policies

Packet routing on the Internet is hierarchical (*Figure 5*). At the lowest level (Tier-3 ISP), every ISP handles the traffic within its network, while interconnecting customers from different ISPs through a direct link via an Internet Exchange Point (IXP). Each ISP can apply their policies, putting at stake the privacy and the neutrality of their customers [32].

When the client and the server are not in the vicinity, it may be necessary to contact an upper-tier ISP (Tier-2 ISP). These ones have access to more routes, for they are directly connected to Tier-1 ISPs. Again, whichever organisation with access to this type of information could apply policies that put at stake the aforementioned Internet neutrality and the security itself [4]. The rules that could be applied include shaping, filtering or even blocking contents [33] with total lack of transparency [25]. Luckily, the Internet Governance, which is addressed below, stands for the idea that no person, company, organization, or government has the power to run the Internet [34].

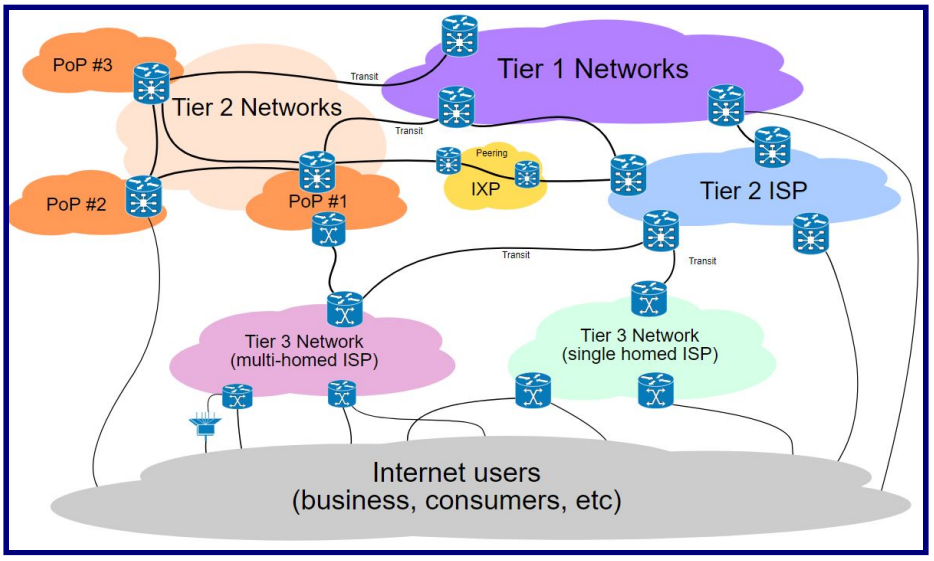


Figure 5. ISP Tiers [74].

Decentralised management: Guifi.net routing and policies

Coming back to Guifi.net, it has a horizontal topology. The routing protocols decide which is the best route depending on proximity, traffic load, and other network characteristics. But, unlike the Internet, this is done in a decentralised way [4]. In spite of this, there exist client nodes and supernodes at Guifi.net. While the clients are usually those that do not have any specific service or use, other than just connecting or allowing the connection to Guifi.net—Customer Premises Equipment (CPE)—, the supernodes provide services and constitute both network access and backbone links [13, 15]. A supernode can be a Domain Name System (DNS) server, a proxy server, a router, an Access Point (AP), or any other device offering connectivity or a service over an existing connection [15].

On the previous paragraphs both Internet and Guifi.net routing have been separately considered. However, when Guifi.net connects to the Internet, all its traffic is handled following the ISP directives. Hence, while “in the wild”, Guifi.net cannot control neither the policies applied by every ISP nor what it is done with the packet flow in the upper tier of the ISP network. In an attempt to elude any possible sneak into the information, secure protocols such as WPA2 (WiFi Protected Access), HTTPS (HyperText Transfer Protocol Secure), SSL (Secure Sockets Layer), or IPsec (Internet Protocol Secure), are employed. However, their effectiveness is not a safeguard against organisations and individuals committed to find backdoors [36, 37].

In this regard, Guifi.net is safe as long as the traffic remains within it, because it runs under an open agreement. Even though it is a private network it is also an AS and, hence, it can allow organisations to operate as ISPs, as long as they meet the Guifi.net requirements. The most widely used routing protocols are Border Gateway Protocol (BGP)—usually between AS—and Open Shortest Path First (OSPF)—generally for internal communications [14].

> "***Guifi.net is a community network that results from a loosely-coupled and decentralised growth that exhibits large local differences, diverse growth, and maturity under a common community license and social network***" [14].

Key function devices

At this point, it is worth mentioning that there are no restrictions imposed on neither the hardware nor the software used to connect to Guifi.net. This is precisely why it is considered an heterogeneous network. Yet, there are ubiquitous devices due to their special characteristics, like Ubiquiti and MikroTik routers. These have many services and configuration options to adapt themselves to different environments and needs [50]. Nonetheless, with the gradual growth of the network, Ubiquiti devices have become the norm as APs for wireless connections to Guifi.net. On the other hand, MikroTik has grown its importance within the cabled connections, thus primarily embodying the routing core of Guifi.net [13, 19].

2.3. Growth of Guifi.net

Spreading both over the territory and in number of users

Guifi.net was created in 2004. Since then, it has been growing at very high rates: at the time of writing this memoir, it holds *34445* working nodes [3] which are mainly deployed in the Spanish territory, although Catalunya and Valencia are the most node-populated areas (*Figure 6*). In fact, by 2011, most of the Catalan territory was covered [25]. But Guifi.net connections are not limited to Spain, as there are currently some nodes in four continents. Hence, it is the largest CN in the world [24]. It is worth mentioning that the connections with long straight lines over long distances that can be seen on *Figure 6* cannot be a wireless link, nor fibre, but VPN connections.

Once Guifi.net Foundation has gained AS status, it is connected to the neutral point in Catalunya: an IXP called CATNIX [43]. It was first connected in 2011 with a MikroTik device running a RouterOS that could handle as much as 300 Mbps both uplink and downlink. Back then, that was enough but, over time, adjustments have been necessary. By 2016, the uplink and downlink needs were of 5 Gbps [13].

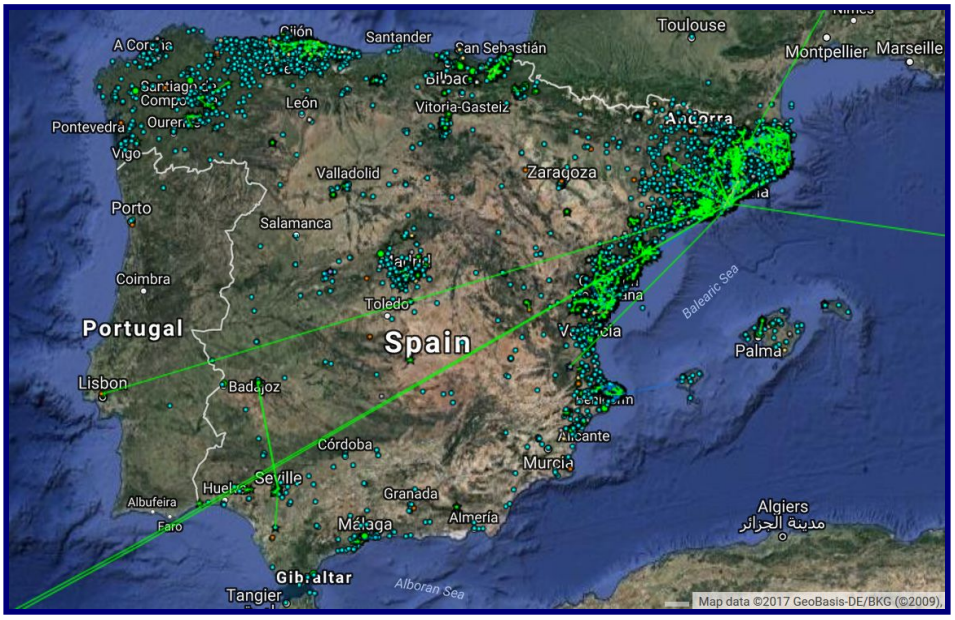


Figure 6. Map depicting Guifi.net nodes and most populated areas [41].

Reasons for growth: a successful project

Guifi.net is open to anyone who voluntarily would like to join the network. Perhaps this is one of the main reasons for its success, apart from the fact that Internet can be accessed either by directly connecting to a nearby AP, through any of the available ISPs, or by deploying a new supernode or isle in the network.

> **"The configuration of all routers is fully automated. The human interaction has been reduced to copy & paste or reflashing procedures. This helps to avoid configuration errors that can create conflicts in the network, and ease the process of setting up nodes, which in turn promotes more participation. [75]"**

Another attractive point is that it is free; the only costs would be those of a host and an antenna (*Figure 7*)—and possibly a contract with an ISP. To ease any complications and favour a smooth network operation, there is a tool called *unsolclic* that helps with the configuration of the most used devices in the network. Adding a device is as simple as filling a form [15], which produces an executable script capable of configuring the device [72, 76]. No wonder Guifi.net is also referred to as the Do It Yourself (DIY) wireless network [78].

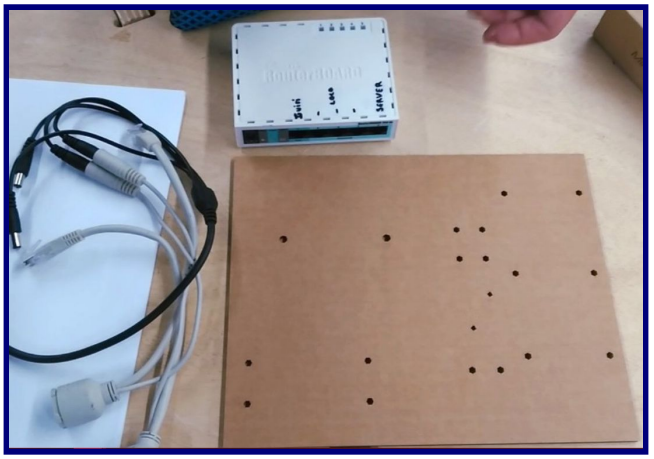


Figure 7. Elements needed to install a MikroTik router outdoors [80].

All of the above, together with a strong sense of community, technical transparency, self regulation, and "*the need of [...] technological achievements that directly promote people's quality of life, [... ,] and encourage sustainable economic development*" is what helps these type of networks thrive [35]. Nonetheless, just like any other network, Guifi.net is also subject to inherent vulnerabilities and threats.

2.4. Threats to Guifi.net

First security steps

As already mentioned, the traffic generated by Guifi.net and destined to the Internet, may be controlled or monitored. When it comes to the incoming traffic, outdated software or poorly deployed security policies can facilitate attacks.

For instance, a weak password to access someone's online banking services on an ill-secured network (like a wireless open network) with outdated software can result in sensitive information leaking to eavesdroppers. Serious consequences are to be expected for the customer: from money loss to stolen identity.

It takes a good antivirus software to keep up-to-date with the latest vulnerabilities and there are a few guidelines that are worth following for network users [81]. These are very simple to follow and yet they constitute a solid security foundation:

1. **Connect to the Internet on secure networks**. If that is not possible, then be aware of the information sent across that network. For example, even though Guifi.net is a private network, the mere existence of gateways make it more vulnerable. Moreover, if the connection is wireless, then all the communications are in the open. WPA2 is there to encrypt the packets on a wireless network. However, this has been proven faulty very recently: if the users do not update their firmware and software on a regular basis, they are exposing themselves to attackers, who can crack this protocol and get access to passwords, private keys or any encrypted data [36, 37].

2. **Update firmware and software**. No software is unbeatable from a security perspective, which makes it crucial to update the systems as soon as a new release is made available. For example, the OpenSSL protocol, used to establish secure sessions, had been proven vulnerable due to the so-called heartbleed vulnerability [26].

3. **Make for strong and non-default passwords**. The usual recommendations are that it has to be no shorter than 8 characters; a combination of alphanumeric, uppercase and lowercase, and as randomised as possible characters; and, ideally, expirable every 72 days.

4. **Close any ports that are not in use**. Especially those with unsafe services such as unencrypted protocols: HTTP/80, Telnet/23, or the proven highly insecure Windows Server Message Block protocol (SMB) on ports 139 and 445 [44]. Even if the user is not connected to the broadband, it is worth mentioning that attacks do not solely belong to the Internet: the attackers can also be directly connected to your own network or in a neighbouring area.

Network vulnerabilities: users at risk

The number of attacks suffered by both organisations, public or private, and individual users is huge. And the back doors left open to perform those attacks are: deprecated OS, old firmware, weak passwords, and risky connections.

One of the most echoed attacks in 2017 was the *WannaCry* case: a "*piece of ransomware designed to extort users, [...] hitting large firms and spreading across networks using holes in*

Windows XP and Windows 7 to propagate far and wide". It made the National Health System (NHS) in United Kingdom (UK) temporarily unavailable and hit companies in Spain such as Telefónica [38].

That was possible after the *EternalBlue* exploit was released by the *Shadow Brokers* hacking group, who allegedly stole it from the National Security Agency (NSA) in the United States (US). After that, Windows released around March 2017 a patch for the exploit, related to the SMB protocol. However, many Windows XP and Windows 7 users failed to update their systems[40].

The scope of similar attacks on a private network like Guifi.net would have less impact: the use of private addressing [14], non-routable on the Internet, makes Guifi.net less prone to this type of threats.

When it comes to surveillance and strong routing policies, the jurisprudence of Guifi.net only affects traffic within its boundaries. Once the packets reach the Internet and are under other ISP's jurisdiction, it is no longer the responsibility of Guifi.net. This is the main threat against openness and neutrality. As a consequence, many are the collectives that have come together to fight for an open and neutral Internet. Some examples are the *Internet Society*, the *Internet Defense League* (IDL), the *Tor Project*, the *Electronic Frontier Foundation* (EFF), *Amnesty International,* or *the Association for Progressive Communications* (APC) to which Guifi.net is a member since March 2017 [45, 46, 47, 56, 79].

2.5. Conclusions

Nowadays, the broadband access is very important and impacts our lives both socially and economically. After Guifi.net became a registered AS, it began to flourish, with many ISPs offering their own Internet services at a fair cost. It can be said that Guifi Foundation has an economic model that autoregulates and focuses on the services it can offer to people more that on the technology itself [19]. This is due to the fact that the technology is individually owned and voluntarily offered. Even non-technical people can join and benefit from it [19, 24]. However, just like any other network, Guifi.net faces some security and privacy threats.

3. Deployment and testing

“It’s really quite simple: The more attack vectors that go unnoticed and the longer we allow attackers time to exploit our systems and infrastructure, the greater their chance for success. It’s on us to close that opportunity.”

—John N. Stewart, Cisco

Guifi.net is approached from a technical perspective. The chapter covers the reasons to assess the security of Guifi.net by analysing routers. Later on, it defines the scope of the study and the different development stages.

3.1. Introduction: the approach

The approach adopted for the security evaluation of Guifi.net is very simple. On the one hand, the success of the communications to and from the Internet is considered to rely heavily on routers. They can figure out the topology of the network and, therefore, are able to choose the best route to reach a packet's destination. If this type of network element happens to be vulnerable, that means it can be attacked and potentially affect many of the communications from and to Guifi.net. Apart from that, any possible loss of information, hijacking of our identity, or theft of personal information should be taken into account. MikroTik and Ubiquiti are the most widely used brands in Guifi.net and, thus, they are the target of the security assessment.

SETTING UP THE ENVIRONMENT. To perform any of the analysis and operations needed to assess the security status of Guifi.net, the following general settings are necessary (*Figure 8*):

1. **VPN connection to Guifi.net**. This is the way to get a private IP address, as if directly connected to Guifi.net. However, a server handles the real connection and sets everything up so that the user can securely access Guifi.net, despite the fact that it is accessing it through the Internet.

2. **UOC server SSH connection**. Once inside Guifi.net, there is administrative access to a virtualized server running Linux from which it is possible to test the network. The connection protocol employed is SSH.

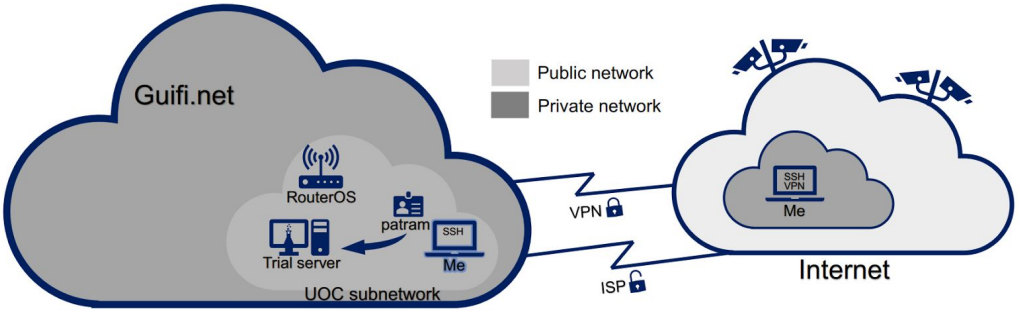


Figure 8. Working environment.

SCOPE. The testing performed on RouterOS is determined by most of the basic security guidelines given in Chapter 2. As it was said, even though they are extremely easy to follow, many users fail to implement them, even whole organisations. On that account, each of the tests examines whether these recommendations are followed on the devices running RouterOS in the selected regions on Guifi.net.

1. Close any ports that are not in use and avoid using unencrypted protocols.
2. Make for strong and non-default passwords for SSH connections.
3. Update firmware and software.

3.2. RouterOS

Guifi.net is a wireless open network that is gradually turning towards cabled connections [14], although it still relies very heavily on radio links [50]. Fibre is usually employed for backbone P2P links. In

particular, it is used to connect to CATNIX, the IXP for Catalunya [13, 14 19, 43]. This is the gateway to the Internet for many networks in the area, and so it is for Guifi.net as well.

MikroTik devices are responsible for the routing between Guifi.net and the rest of the world [13]. They are capable of managing large loads of traffic and many routes [13]. These routers are ready to operate on radio links as well as with cabled communications (*Figure 4*) [51, 73]. In fact, Guifi.net backbone is at least partly built with them. This can be concluded from the fact that the backbone set of IPs is 172.16.0.0/12 and that of RouterOS is 172.28.0.0/14 [53]. That is, the RouterOS range of addresses (172.28.0.0 - 172.31.255.255) overlaps with that of the backbone (172.16.0.0 - 172.31.255.255). However, the user is free to register the devices they are using on Guifi.net. As a consequence, data from the site may not be accurate. In an attempt to check whether the backbone is built up with MikroTik routers, the developed set of tests were run over both range of addresses. It is good news they rendered unsuccessful due to active firewalls.

Yet, the routers analysed here are not part of the backbone. Most of them are interconnected via Ethernet cables, instead of fibre [73]. Particularly, they are employed, as Ubiquiti devices, to enable radio links (*Figure 9*). Moreover, these routers lack in some cases strong security measures [*Chapter 4*].

The software that every single MikroTik router runs is called RouterOS, an OS specifically designed for them. There is a virtualized version as well called CHR [51]. Some of its utilities include *WinBox* or *Console*. While the former is a GUI (Graphic User Interface) manager, the latter is a terminal to manage and configure the router [85].

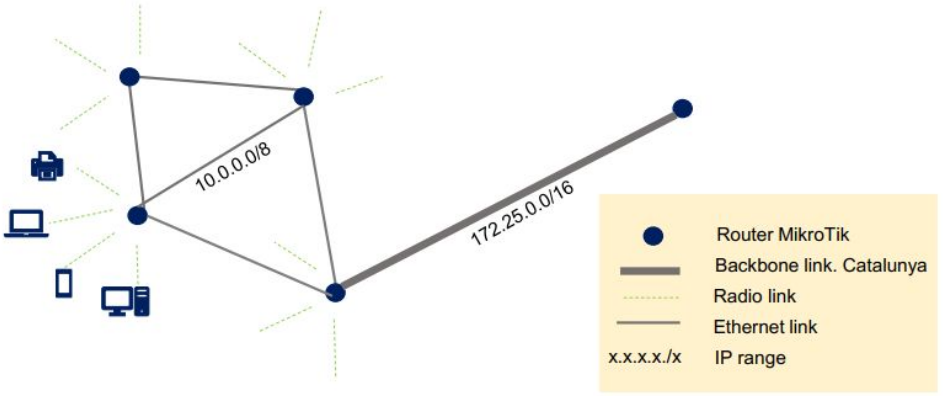


Figure 9. Different connections and uses for MikroTik routers.

3.2.1. Implementation

MikroTik routers are security assessed by running a Shell script called ***RouterOS tests*** that makes use of NMAP, SSH, and SNMP tools. They are implemented in that very order because the results from the NMAP test determine whether there are MikroTik devices within a set of IPs. The SSH login attempt and the firmware detection only make sense on our target: Mikrotik devices.

This script needs two arguments as input: firstly, the path to the file that contains the IPs to scan; secondly, the chosen directory name. The idea is that a group of different towns in Catalunya can be scanned to assess the "health" of their routers. The chosen regions (*Table 3, Figure 10*) [53] have in common a high rate of used addresses per total available host IPs.

As it can be seen in *Figure 10*, these are spread over a large area of the Catalan territory and includes both densely populated large and intermediate cities, and small rural villages. It seems reasonable to assume that such sample can therefore be considered representative of the entirety of Guifi.net.

Town / IP range / Usage [%]					
Barcelona	10.228.192.0/20	9%	**Franqueses del Vallès, Les**	10.139.1.0/24	62%
	10.90.224.0/20		**Garriga, La**	10.139.214.0/24	70%
	10.139.88.0/21		**Llerona**	10.90.63.0/24	28%
	10.139.36.0/22		**Maresme**	10.138.63.0/24	39%
Bigues i Riells	10.90.152.0/23	49%	**Olèrdola**	10.140.16.0/23	46%
Bitem	10.90.175.0/24	23%	**Olot**	10.138.38.0/23	19%
Canet de Mar	10.138.52.0/22	41%	**Sant Pere de Vilamajor**	10.91.216.0/24	33%
Figaró-Montmany	10.139.121.0/24	44%	**Selva, La**	10.139.160.0/22	31%
			Vilafranca del Penedès	10.139.48.0/23	36%

Table 3. IP group selection and their usage percentage (assuming the source is frequently updated) [53]

Figure 10. Map with the scanned regions [60].

After running ***RouterOS tests*** on a town—La Selva, for example—(*Figure 11*) a directory with the region name is created which contains test results and data-handling files. One of the most interesting ones is *mikro_ports.txt*, which lists the MikroTik IPs and their open ports. In particular, *port_22open.txt* stores the IPs with port 22 open. The same pattern is followed for other insecure ports such as 23 and 80, with unencrypted services associated to them. Another file with important results is *firmware.txt*, where all the MikroTik devices are organised under different headings, depending on when they were last updated. Finally, there is a key file called *ssh22_ips.txt*, that lists the MikroTik devices that still have the default credentials for SSH/22.

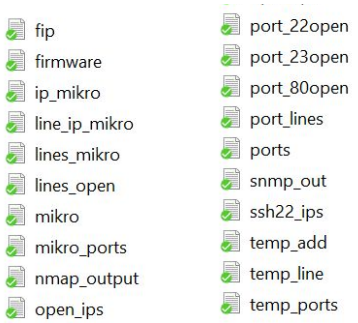


Figure 11. Directory contents after running the script *routerOS tests* for La Selva.

As it has been said, the script ***RouterOS tests*** developed to look for and later scan MikroTik IP addresses is structured in sections, which performs the following tests.

TEST #1: MIKROTIK OPEN PORTS. NMAP allows for OS detection: `nmap -n -Pn -sV -p21,22,23,53,80,81,82,443,2200 -T4 -iL "$file" -oX "$nmapout"` [56, 58, 71]. That way, from the IP addresses provided, those belonging to a MikroTik router can be identified. The next step consists on analysing the NMAP results searching for all MikroTik open ports. Specifically, it looks for FTP/21, …, HTTPS/443. Using the rest of ports as alternatives to the usual HTTP/80 or SSH/22 would reduce the possibilities of a bruteforce attack. A good firewall configuration is also essential [42, 55, 59]. In particular, the configuration of MikroTik devices have Telnet, FTP, WWW (HTTP), and API-SSL (HTTPS) services open by default. They should be turned off if not in use for security reasons.

TEST #2: SSH LOGIN ATTEMPT WITH DEFAULT CREDENTIALS. Once the first test is performed, it is possible to use the router network IP addresses to try to connect via SSH [52]: `sshpass -p "" ssh -o StrictHostKeyChecking=no admin@"$ip_add"`, as long as port 22 is open. The default username-password for MikroTIk routers is *admin-""*. The IPs where the login attempt with the default credentials was successful are kept on a file.

TEST #3: FIRMWARE DETECTION AND EVALUATION. SNMP and the available MIB OIDs (Management Information Base | Object IDentifiers) [64, 65] make it possible to detect firmware from MikroTik devices [39]. These results, associated to each IP, are organised according to their last update year [61, 62, 63] a in file called *firmware.txt.* The firmware version information is written in a file and works as a database for the script.

The Unix utility used to launch the requests is called snmpwalk: `firmware=$(snmpwalk -c public -v1 "$ip_add" "$iso" | cut -d '"' -s -f2 )`. This command surprisingly uses insecure SNMP version 1 to gather the firmware information from a public community.

3.3. AirOS

There are many types of Ubiquiti products, and their uses are broad. The main product lines that may be of interest are AirMAX ac and EdgeMAX [67]. Certain AirMAX ac routers, however, are the most widely employed, for they are economic and flexible. They are wireless devices equipped with antennas, can function both as a bridge or as a router, and have multiple software utilities to meet different needs [23]. They run AirOS. Unlike MikroTik routers, AirOS firmware does not seem to follow a logic of updates and, besides, it is almost impossible to find information regarding the past firmware releases. This is what limits the analysis of AirOS routers in Chapter 4.

These routers are usually working as client nodes, processing and establishing connections from users and creating meshed networks with other devices alike (*Figure 12*). At least one of the nodes of that mesh network is to connect to a supernode, usually a MikroTik [14].

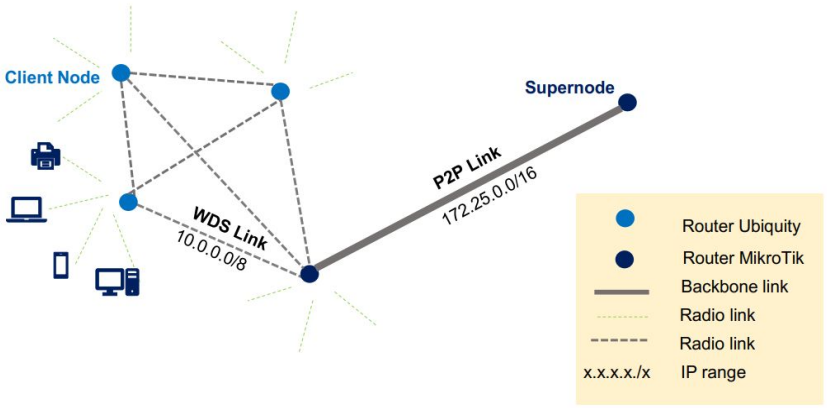


Figure 12. Uses of routers Ubiquiti and MikroTik.

3.3.1. Implementation

The implementation steps followed for the ***AirOS tests*** script are the same as for MikroTik, with some improvements in the algorithms and adaptations due to the particularities of AirOS devices. That is, the regions to scan remain the same as well as the output files, even though some of their names may reference MikroTik. In what follows, attention is paid to the differences.

TEST #1: UBIQUITI OPEN PORTS. The new NMAP command is `nmap -n -Pn -A -p21,22,23,53,80,81,82,443,2200 -T4 -iL "$file" -oX "$nmapout"` [57, 86], as we do not only need service detection (-sV), but a more advanced search that also includes an OS search. Otherwise, the pattern "Ubiquiti" would not be found in the output file and, as a result, neither the target devices.

TEST #2: SSH LOGIN ATTEMPTS WITH DEFAULT CREDENTIALS. Now that Ubiquiti routers are identified, there are several SSH connection attempts per IP. This is due to up to 3 different default credentials to connect to these routers via SSH [68]. The commands are: `sshpass -p "" ssh -o StrictHostKeyChecking=no guest@"$ip_add"`, `sshpass -p "ubnt" ssh -o StrictHostKeyChecking=no ubnt@"$ip_add"`, and `sshpass -p "ubnt" ssh -o StrictHostKeyChecking=no root@"$ip_add"`, where the following user-password pairs have been used: guest-"", ubnt-ubnt, and root-ubnt.

TEST #3: FIRMWARE DETECTION AND EVALUATION. As mentioned before, the information gathered from Ubiquiti past releases is scarce [37, 57, 66]. Therefore, many of the identified routers may fall in a new category called *Unknown Version*. A new OID that identifies the firmware of Ubiquiti devices within the MIB is also needed [70].

3.4. The importance of firewalls

Deciding who comes for dinner tonight is responsibility of the host. Likewise, a user should be able to lay out a series of rules when connecting to a network that clearly state which services are welcome to establish communication. A good configuration of firewalls allow for secure networks and devices. In

medium to large networks this protects the rest of the network from the DMZ (Demilitarised Zone) and the perils of connecting to the Internet.

Apart from routers, web proxies are other elements that face the Internet from the edge of the network. They request and accept HTTP/HTTPS connections between clients and web servers and need a firewall as much as the routers do. And so does any device connected to the Internet. *Iptables* is the firewall that use most of the software running in Unix. That is the case of Squid, the software that works as a Web Proxy in the supernodes of Guifi.net.

3.5. Conclusions

The main tools used to develop the scripts to test both MikroTik and Ubiquiti routers are NMAP, SSH, and SNMP. The three of them work together to provide information based on an input. While NMAP relies on a file with IP addresses or ranges of IP addresses, SSH and SNMP relies on the subsequent selection of the IPs of the target routers. That way, the scans are guaranteed to be performed exclusively on the desired devices.

4. Guifi.net security evaluation

“How safe and secure is the Internet? Well… Would you leave all your [...] belongings on the road and expect them to be there when you came back [...]?”

— Anthony T. Hincks

The tests performed in Chapter 3 lead to the management and analysis of their outcome data. In every result discussion the characteristics of both MikroTik and Ubiquiti are taken into account, plus those of Guifi.net itself.

4.1. Context where results apply

The upcoming analysis of results can be taken as a snapshot of the current security configuration on router devices in Guifi.net when it comes to basic security guidelines. They were performed on IPs from 14 Catalan regions. Namely, Barcelona, Bigues i Riells, Bitem, Canet de Mar, Figaró-Montmany, Les Franqueses del Vallès, La Garriga, Llerona, Maresme, Sant Pere de Vilamajor, La Selva, and Vilafranca del Penedès. These are among the areas in which the largest proportion of Guifi.net IPs is currently in use out of all the available addresses. This choice provides better chances to obtain enough data to make meaningful statistics.

The analysis is done with an emphasis on the usage penetration of unencrypted services such as Telnet and HTTP, the latest firmware, and default credentials on the SSH protocol. The results may denote the users' degree of security awareness or the effectiveness of devices prompting the user to improve or update certain configuration settings.

4.2. Test results

After executing the scripts with tests that look for vulnerabilities on router devices in Guifi.net and later ***RouterOS*** ***data*** and ***AirOS*** ***data*** scripts to arrange those results into meaningful data, it is time to analyse and plot that information. The chosen software is *LibreOffice*, for it provides a *Calc* software with the tools needed to plot colourful column and pie charts.

A first look at the data reveals that an 18% of the roughly 18000 IPs scanned are actually assigned IPs (*Figure 13, left pie chart*); the other values correspond to unassigned or down IPs (most probably unassigned) and reserved IPs, that is, network and broadcast addresses per IP range. The percentage of assigned IPs can be deducted from the usage percentage indicated on *Table 3* for every single region. From those, a non-impressive 57% are firewalled, while the rest are made up with Ubiquiti devices (15%), MikroTik routers (19%), and others such as Linksys or Buffalo makes [14] (*Figure 13, right pie chart*).

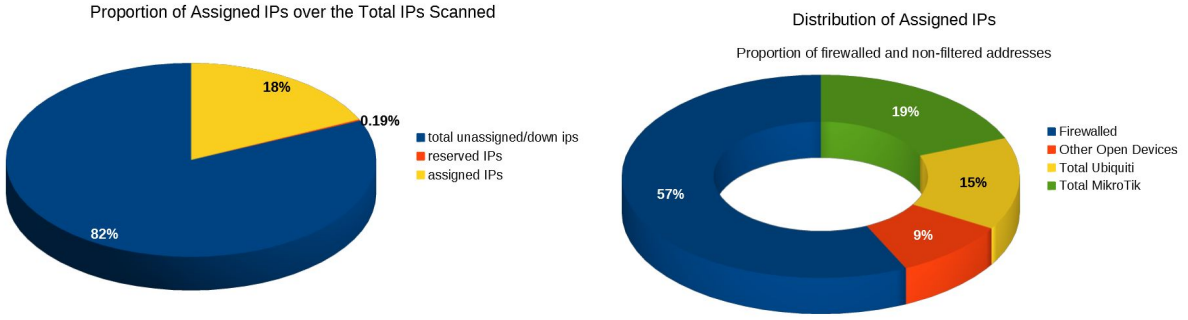


Figure 13. Proportion of MikroTik and Ubiquiti devices over the assigned IPs scanned.

4.2.1. RouterOS

Remarkably, about 2 out of 3 of the MikroTik routers considered tested positive for open 23/Telnet and 80/HTTP ports (*Figure 14*). This is an important security threat, as there are well-known risks associated with these unencrypted services, such as monitoring of web surfing, eavesdropping of username and password, or even potential risk of personal files being stolen or hijacked. As it could be expected, it is also a positive sign to notice that all but 3 SSH connection attempts with the default credentials (username:"admin", password:"") were unsuccessful (that is, less than 1%) (*Figure 15*).

Mikrotik recommends setting the SSH port to an unusual port number (2200 is suggested) to avoid random SSH bruteforce login attempts [59]. What is more, the available options to configure the MikroTik routers are SSH, secure *Winbox*, or HTTPS services. It is recommended to use the latest *Winbox* version for secure access, as well [59].

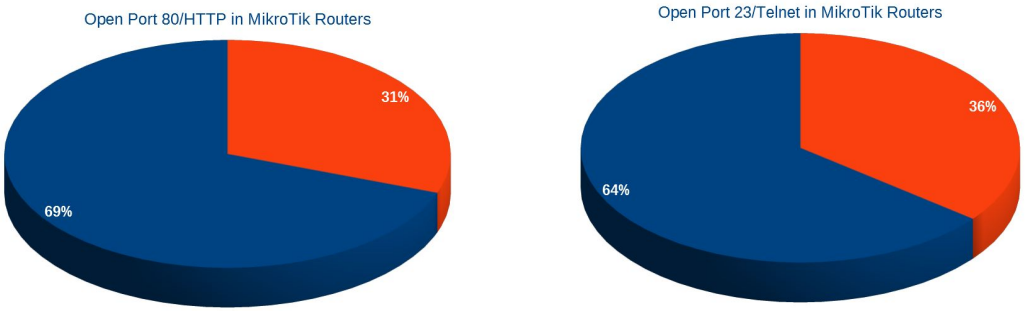


Figure 14. Percentage of routers with ports 80 and 23 opened.

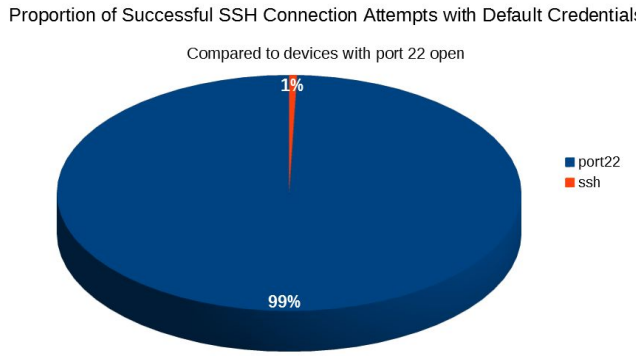


Figure 15. Percentage of open SSH/22 service with default *user-password*.

When it comes to the firmware versions running in the accessible scanned routers, there is also plenty of room for improvement, security-wise (*Figure 16*). First of all, it must be noted that 19% of them had port 161/SNMP either closed or firewalled and hence, no firmware version information could be retrieved. Even in the best-case scenario, for example, assuming that all of these non-responsive routers had up-to-date firmware, only 1 out of 3 routers would have been updated in 2017. However, depending on the actual firmware version of the irresponsive routers, the percentage of outdated and obsolete devices could be as large as 84%. Once again, there are obvious security threats associated with running deprecated firmware like back doors for exploitation.

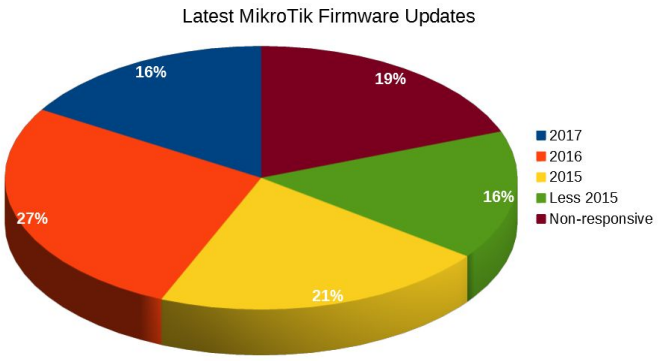


Figure 16. Distribution of firmware updates over the years in MikroTik routers.

It is possible that the reason behind the high number of routers not updated was the initiative *unsolclic* that automates the process for the user. One of its disadvantages is that it does not update the firmware, leaving it exactly as it is once it is installed for the very first time. Besides, there is no way to guarantee that in the *unsolclic* process the newcomer is installing the most up-to-date firmware, due to the need of deploying each *unsolclic* script for every device make and version [76].

In *Figure 17*, the geographical distribution of the results on open 23/Telnet and 80/HTTP ports is shown, together with the total number of MikroTik devices found in each region. Also, in *Figure 18*, the geographic distribution of firmware versions is depicted. First of all, notice that Bitem and Maresme need to be excluded from this geographic analysis due to the low number of accessible devices found (0 and 6, respectively).

As a rough way to assess the security status of a region in a range from 1 (excellent security) to 100 (very insecure), one could average the percentage of open 23/Telnet and 80/HTTP ports and the percentage of routers running deprecated firmware (*Table 4*). This simple calculation reveals that La Garriga and Sant Pere de Vilamajor stand out as the most healthy regions when it comes to basic security settings, scoring 31.80 and 35.23, respectively. On the contrary, Olot, Llerona, Bigues i Riells, and Barcelona would be ranked the most insecure. Interestingly, note that, in spite of their good score, La Garriga and Sant Pere de Vilamajor show large percentages of deprecated firmware (*Figure 18*) which, perhaps, showcases that the awareness about this basic security aspect needs to be emphasized the most.

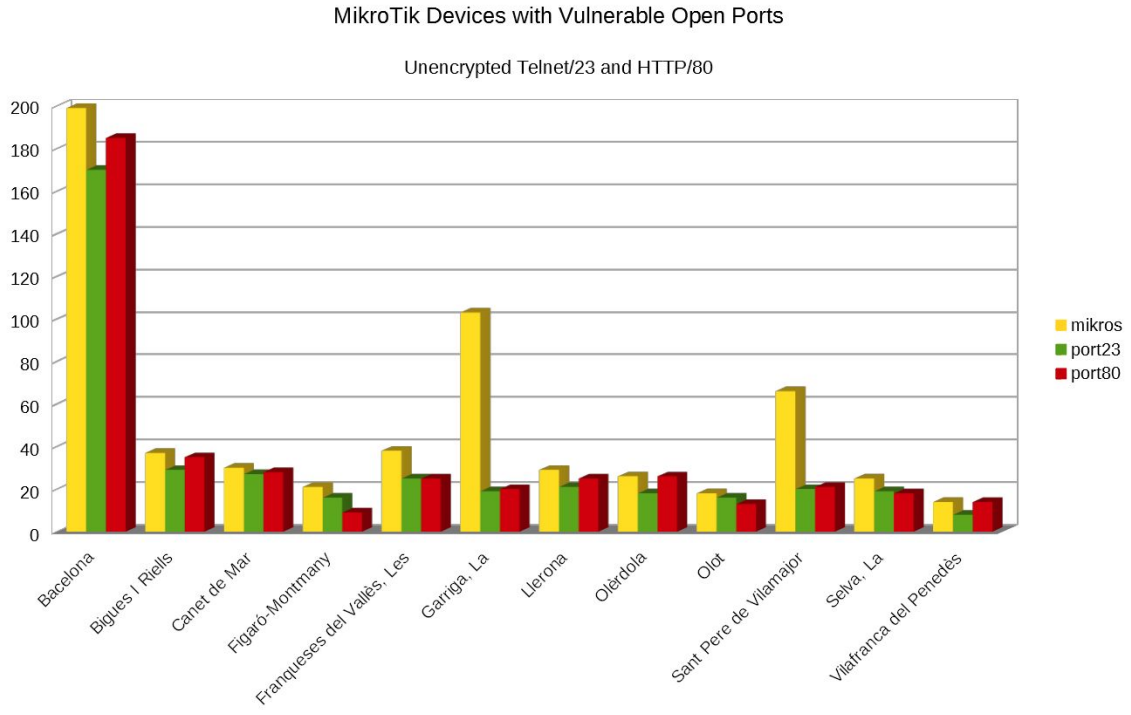


Figure 17. Number of MikroTIk devices with ports 23 and 80 open per region.

	23/Telnet	*80/HTTP*	*SSH*	*Outdated*	*SCORE*	Population 2016 (Google)
Bacelona	85.43	92.96	0.00	52.26	57.66	1609000
Bigues I Riells	78.38	94.59	0.00	59.46	58.11	8915
Canet de Mar	90.00	93.33	0.00	33.33	54.17	14284
Figaró-Montmany	76.19	42.86	4.76	47.62	42.86	1092

Franqueses del Vallès, Les	65.79	65.79	0.00	65.79	49.34	19417
Garriga, La	18.45	19.42	0.00	89.32	31.80	15912
Llerona	72.41	86.21	0.00	79.31	59.48	1055
Olèrdola	69.23	100.00	0.00	46.15	53.85	3529
Olot	88.89	72.22	5.56	83.33	62.50	34000
Sant Pere de Vilamajor	30.30	31.82	0.00	78.79	35.23	4257
Selva, La	76.00	72.00	4.00	60.00	53.00	3854
Vilafranca del Penedès	57.14	100.00	0.00	57.14	53.57	39365

Table 4. Percentages of open ports and outdated firmware, plus a score per region.

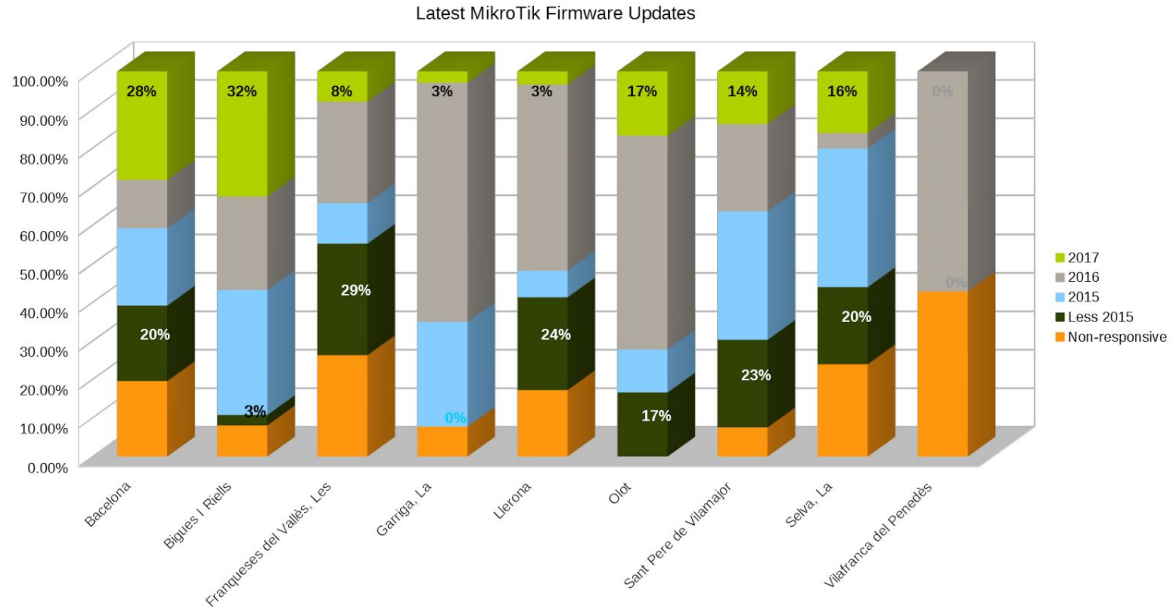


Figure 18. Distribution of firmware updates over the years in MikroTik routers.

Finally, even though it is beyond the scope of this work, one could wonder whether these data support the naive view that digital literacy and security awareness are larger in densely populated areas (where the Internet is widely available) as compared with small/rural populations. This is certainly not the case. The top regions in the ranking have populations ranging from just about 1000 inhabitants (Figaró-Montmany) to over 19000, in the case of Les Franqueses del Vallès, whereas more densely populated cities, such as Vilafranca del Penedès (over 39000 people) and Barcelona (over 1.6 million people) feature in the intermediate or low positions of the ranking. Likewise, other small villages such as Olèrdola or Bigues i Riells perform poorly in the ranking.

4.2.2. AirOS

To conclude this analysis, let us zoom into the remaining 3% of open devices running AirOS instead of RouterOS (*Figure 13*). As it can be seen in *Figure 19*, the results for open 23/Telnet and 80/HTTP ports are markedly different. Namely, the highly insecure 23/Telnet port was found to be open only in 16% of the scanned routers (recall from *Figure 14* that 63% of the MikroTik routers had the 23/Telnet port open). This may be due to this port being deactivated by default, which would be good news.

However, it turns out that the last couple of years AirOS devices have been suffering from different attacks, such as defacing, router farming, or worm attacks [82, 83, 84]. Subsequent firmware updates may help ensuring Telnet is not open by default and it may even prompt the user to close insecure or unused ports [37], even to update their firmware and restrict access through firewall filtering [84].

On the other hand, it is equally remarkable that port 80/HTTP is open in almost all scanned devices (*Figure 19*). Noticing that, similarly to MikroTik routers, there is an AirOS feature that allows configuration and management of Ubiquiti devices via HTTP or HTTPS [23]. Nonetheless, most of the attacks mentioned in the previous paragraph rely on HTTP/HTTPS and DNS alterations [82, 84].

For completeness, let us mention that none of the SSH connection attempts with default credentials was successful in the case of Ubiquity devices.

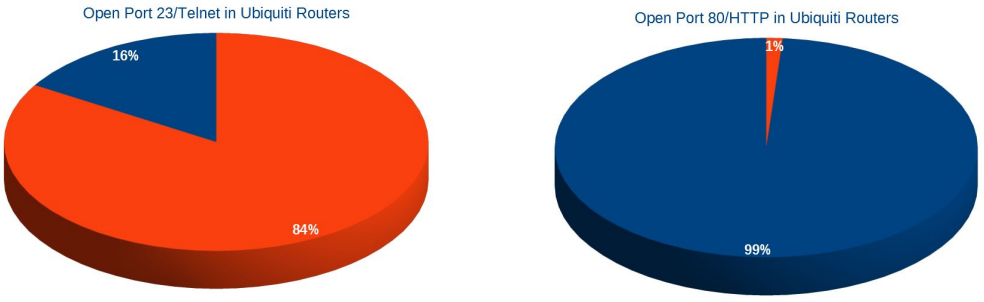


Figure 19. Percentage of routers with ports 80 and 23 opened.

Unfortunately, no conclusive results may be drawn from the analysis on firmware updates. Roughly, half of the accessible devices returned firmware versions that could not be classified according to their year of release. In contrast to what happens with MikroTik routers, the information available about firmware versions of Ubiquiti devices is sparse and largely incomplete (*Figures 20, 21 ,22*). What can be said according to the available information is that the number of Ubiquity routers running deprecated firmware would exceed that of updated devices, even in the best-case scenario in which all "non-responsive" devices (closed 161/SNMP port) happened to be up-to-date.

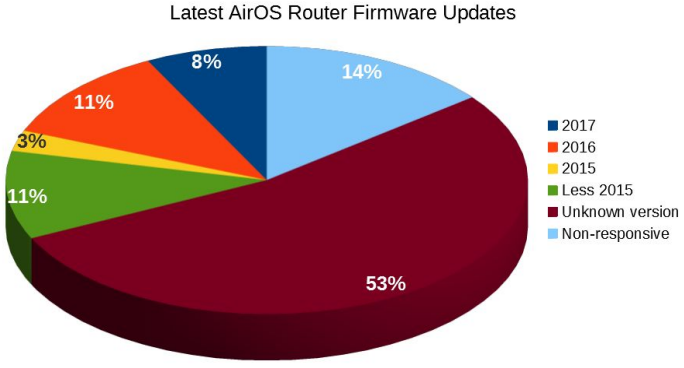


Figure 20. Distribution of firmware updates over the years in Ubiquiti routers.

One may also perform a geographic assessment analogous to that of the MikroTik devices. In this case, however, the number of accessible Ubiquiti devices was over 10 in only 6 out of the 14 scanned regions. These were Barcelona, Figaró-Montmany, Olèrdola, Olot, La Selva, and Vilafranca del Penedès (*Figure 21*). Due to the "inconclusiveness" of the firmware-version test, the resulting ranking may be largely misleading. For instance, Barcelona seems to score much better when it comes to the security settings of Ubiquity devices (*Table 5*), but this is probably due to the large number of devices whose firmware version could not be classified by date of release (37% in this case).

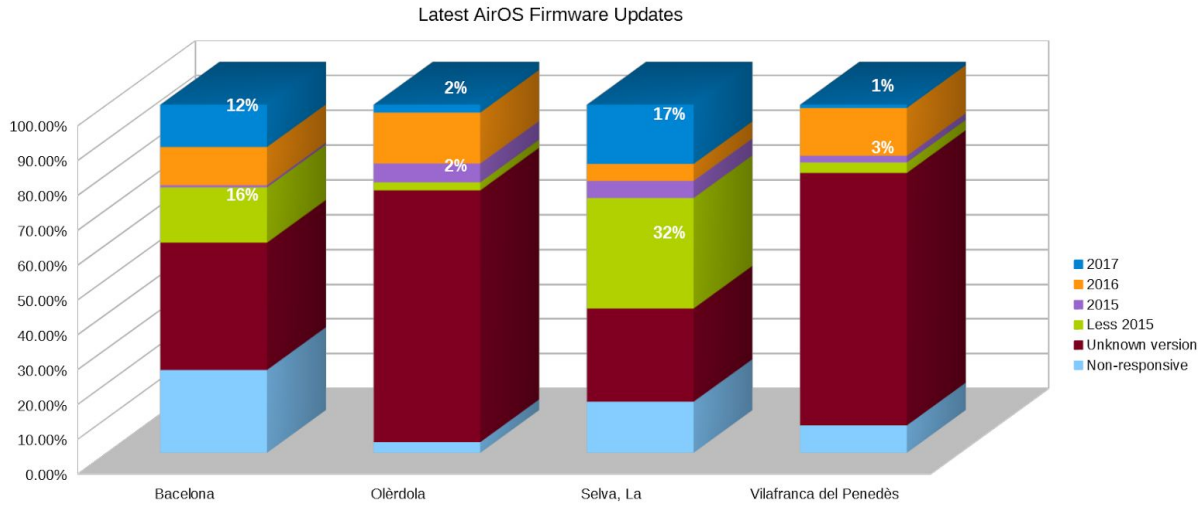


Figure 21. Distribution of firmware updates over the years in Ubiquiti routers per regions.

	23/Telnet	*80/HTTP*	*Outdated*	*SCORE*
Barcelona	7.93	99.39	27.44	33.69
Figaró-Montmany	54.55	100.00	9.09	40.91
Olèrdola	0.77	99.23	22.31	30.58
Olot	100.00	100.00	42.86	60.71
Selva, La	87.80	95.12	41.46	56.10
Vilafranca del Penedès	3.92	100.00	18.63	30.64

Table 5. Percentages of ports open and outdated firmware, plus scores of relevant data from regions.

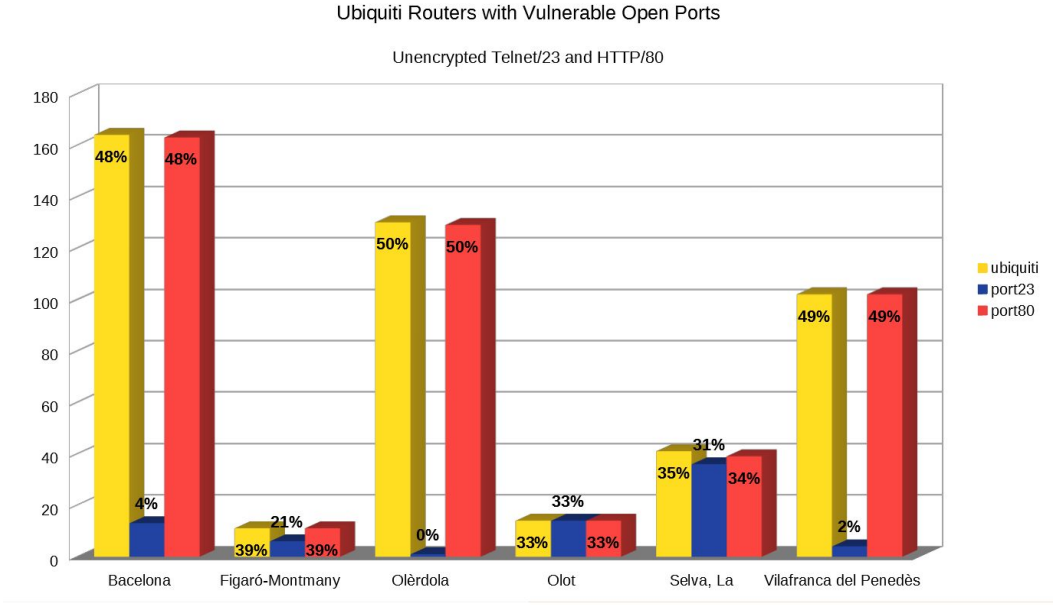


Figure 22. Number of Ubiquiti devices with ports 23 and 80 open per representative region.

5. Conclusions

“Luck is great, but most of life is hard work.”

—Iain Duncan Smith

This chapter wraps up all the contents and determines the main points, the benefits of the project, and its possible improvements.

5.1. Guifi.net security analysis

The main objective of the project has been assessing the current security status of Guifi.net by testing elementary security settings on key network devices: the routers. In particular, the focus was placed on searching for unused open ports, the usage of default passwords allowing for SSH connections, and the persistence of outdated firmware in network devices.

From the analysis of the previous chapter, the main conclusion that can be drawn is that not keeping up-to-date with firmware updates is currently the biggest security issue in Guifi.net, at least among the non-firewalled devices that could be studied. Neither the accessible MikroTik nor the Ubiquiti routers tested seemed to be dully updated, in general terms. On the other hand, the vast majority of the scanned devices had changed the default passwords on 22/SSH, thus making brute-force connection attempts mostly unsuccessful.

Nevertheless, this type of open doors, even on a single node, leaves plenty of options for an attacker to alter other services at will, and spread freely through the network. Hence, the importance of hardening network devices. What is more, it is equally important to promote security awareness and digital education for a society that, more than ever, relies on digital tools to thrive.

5.2. Project evaluation: innovative and applicable results

Even though this project is limited in scope, it tackles relevant security settings that can give a bona fide description of the general "health" status of Guifi.net. Even though the entirety of nodes from the network could not be scanned due to time limitations, the chosen sample is representative both from the geographical and demoscopic viewpoints. Hence, in spite of its simplicity, the aftermath of this project is a wake-up call about the necessity to increase the awareness of the importance of the use of updated firmware among the users of Guifi.net and it also calls for a revision of the existing DIY configuration tools offered to the users.

5.3. Future research

The present project was limited to the analysis of a couple of router makes. However, it could well be extended to other network elements and geographical regions, as the first step towards a more thorough security study of Guifi.net. Besides routers, web proxies also play a key role and may have an impact in the security of an open network like Guifi.net, which is connected to the Internet.

The scripts can be largely improved. For example, many of the algorithms can be transformed into functions for easy re-use, making the code clearer and more modular. Besides, there is an undesirable dependence on the layout of the working directories. Another negative/inefficient aspect of the work is that SNMP was performed on all routers, even those with port 161/SNMP closed.

During the AirOS tests, the scripts were significantly improved and made more robust (as compared with the original MikroTik data-analysis scripts). In addition, an alternative, yet unused, script for testing AirOS devices was written. This used SNMP to search for network devices, which turned out to be inconvenient since all routers with port 161/SNMP closed would be virtually invisible to the analysis.

Finally, while port 22/SSH was checked for connectivity using default username and password, both ports 443/HTTPS and 80/HTTP can act as access doors to management and configuration tools.

The use of default credentials in these is also an important security risk that could have been assessed. Likewise, ports 22/Telnet and 80/HTTP are not the only ones suffering from known vulnerabilities.

“Have you thought of an ending?"
"Yes, several, and all are dark and unpleasant."
"Oh, that won't do! [Stories] ought to have good endings.
How would this do: and they all
[found a way to harden their devices]?"
"It will do well, if it ever came to that."
— J.R.R. Tolkien, The Fellowship of the Ring

6. References

[1] **Women's Forum Global Meeting 2017** (2017). Date accessed 04/10/17. Retrieved from: http://www.womens-forum.com/meetings/global-meeting-2017

[2] **Open Technology Institute**. Date accessed: 04/10/17. Retrieved from: https://www.newamerica.org/oti/

[3] **Guifi.net/ca**. Date accessed: 04/10/17. Retrieved from: www.Guifi.net/ca

[4] Lepp, Haley. Georgetown University. Intersect, Vol 8, No 2 (2015). **An Investigation of Decentralized Networks Based Upon Wireless Mobile Technologies**. Date Accessed: 04/10/17. Retrieved from: http://web.stanford.edu/group/ojs3/cgi-bin/ojs/index.php/intersect/article/view/697/809

[5] The Guardian. **Why Network Neutrality Protests Matters** (11/07/17). Date accessed 04/10/17. Retrieved from: https://www.theguardian.com/technology/2017/jul/11/what-is-net-neutrality-threat-trump-administration

[6] Internet World Stasts. **Internet Growth Statistics**. Date accessed: 04/10/17. Retrieved from: http://www.internetworldstats.com/emarketing.htm

[7] European Parliament. **Bridging the Digital Divide in the European Union** (2015). Date accessed: 04/10/17. Retrieved from: http://www.europarl.europa.eu/RegData/etudes/BRIE/2015/573884/EPRS_BRI(2015)573884_EN.pdf

[8] European Parliament. **European Union Digital Divide Infographic** (2014). Date accessed: 04/10/17. Retrieved from: https://ec.europa.eu/digital-single-market/en/news/eu-digital-divide-infographic

[9] UK Government (2017). **National Cybersecurity Strategy**. Date accessed: 04/10/17. Retrieved from: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/567242/national_cyber_security_strategy_2016.pd

[10] Icelandic Human Rights Centre. **The Human Rights Protection of Vulnerable Groups**. Date accessed 04/10/17. Retrieved from: http://www.humanrights.is/en/human-rights-education-project/human-rights-concepts-ideas-and-fora/the-human-rights-protection-of-vulnerable-groups/

[11] Dalmau, Lluís. **Guifi.net** (2013). Emprenedoria Social. YouTube Video. Date accessed: 04/10/17. Retrieved from: https://www.youtube.com/watch?v=W65TggLGbJ4

[12] **Fundació Guifi.net**. Emprenedoria Social (2014). YouTube Video. Date accessed: 19/11/17. Retrieved from: https://www.youtube.com/watch?v=AavKbYNlses

[13] **Guifi.net: la red libre más grandre del mundo**. Mikrotik Talk (2016). YouTube Video. Date accessed: 19/11/17. Retrieved from: https://www.youtube.com/watch?v=1eGjYWNArGE https://mum.mikrotik.com//presentations/ES16/presentation_3911_1474457763.pdf

[14] Vega, Davide, et al. (2015). Universitat Politècnica de Catalunya. **A Technological Overview of the Guifi.net Community Network**. Date accessed: 04/10/17. Retrieved from: https://www.researchgate.net/publication/283984629_A_technological_overview_of_the_Guifinet_community_network

[15] Sarlé Puig, Albert (2012). Universitat Autònoma de Barcelona. Escola d'Enginyeria. **Millora en la Creació i Configuració Automatitzada de Dispositius a la Xarxa Lliure Guifi.net**. Date accessed: 03/10/17. Retrieved from: https://ddd.uab.cat/record/114915

[16] PMdesire. **SWOT Analysis for Risk Identification**. Date accessed: 05/10/17. Retrieved from: http://pmdesire.com/swot-analysis-for-risk-identification/

[17] Ministerio de Industria, Economía y Competitividad. Dirección General de Industria y de la Pequeña y Mediana Empresa. **Análisis DAFO**. Date accessed: 29/09/17. Retrieved from: http://dafo.ipyme.org/Paginas/Home.aspx

[18] **Gantt chart software** (2017). Date accessed: 26/09/17. Retrieved from: http://ganttproject.sourceforge.net/

[19] Roca, Ramón (2012). **Infraestructuras como procomún: Ramon Roca at TEDxMadrid.** Date accessed: 19/11/17. Retrieved from: https://www.youtube.com/watch?v=d_oTIoORR30

[20] **Xarxa de Comuns**. Fundació Guifi.net**.** Date accessed: 19/11/17. Retrieved from: https://fundacio.Guifi.net/Xarxa_de_Comuns.

[21] Guifi.net. **Proveïdors verificats a Guifi.net world.** Last accessed: 26/12/17. Retrieved from: https://Guifi.net/ca/node/3671/suppliers

[22] **Goufone.** Date accessed: 26/12/17. Retrieved from: http://goufone.com/

[23] AirOS8. Operating system for Ubiquiti AirMAX ac series products (12/01/2017). Ubiquiti Networks, Inc. Retrieved from: https://dl.ubnt.com/guides/airOS/airOS_UG_V80.pdf

[24] Baig, Rober et al (2016). Fundació Guifi and Universitat Politècnica de Catalunya. **Making Community Networks economically sustainable, the Guifi.net experience**. ACM Digital Library. Date accessed: Retrieved from: https://dl.acm.org/citation.cfm?id=2940163
[25] Marsan-Raventós, Clara (2012). Universitat Oberta de Catalunya. **The Right to Access Internet Services: the Importance of Net Neutrality. The Wireless Network Community Guifi.net in Catalonia.** Date accessed: 03/10/17. Retrieved from: https://www.academia.edu/2922701/The_right_to_access_Internet_services_the_importance_of_net_neutrality._The_wireless_network_community_Guifi.net_in_Catalonia
[26] **A technical view of theOpenSSL 'Heartbleed' vulnerability** (2015). IBM. Date accessed: 19/11/17. Retrieved from: https://www.ibm.com/developerworks/community/files/form/anonymous/api/library/38218957-7195-4fe9-812a-10b7869e4a87/document/ab12b05b-9f07-4146-8514-18e22bd5408c/media.
[27] TINETAULA: **Internet a la teva mà**. Date accessed: 26/12/17. Retrieved from: http://www.tinet.cat/tinetaula/curs-internet-la-teva-ma
[28] TINETAULA: **Participa a les xarxes socials**. Date accessed: 26/12/17. Retrieved from: http://www.tinet.cat/tinetaula/curs-participa-les-xarxes-socials
[29] Softcatalà. **Guia d'autodefensa digital**. Date accessed: 26/12/17. Retrieved from: https://autodefensa.softcatala.cat/).
[30] **TINET. Història** (2015). Date accessed: 26/12/17. Retrieved from: http://www.tinet.cat/tinethistoria http://www.tdx.cat/handle/10803/9156
[31] Hern, Alex. **'All wifi networks' are vulnerable to hacking, security expert discovers** (16/10/17). The Guardian. Date accessed: 26/12/17. Retrieved from: https://www.theguardian.com/technology/2017/oct/16/wpa2-wifi-security-vulnerable-hacking-us-government-warns
[32] Wikipedia. **Internet exchange point** (12/12/17). Date accessed: 26/12/17. Retrieved from: https://en.wikipedia.org/wiki/Internet_exchange_point
[33] **Delivering the Bits** *(2008). Tech.View.* London: The Economist Newspaper NA, Inc, Apr 18, ProQuest Central.
[34] Wikipedia. **Internet governance** (15/11/17). Date accessed: 26/12/17. Retrieved from: https://en.wikipedia.org/wiki/Internet_governance
[35] Frauenfelder, Mark. **Sir Tim-Berners Lee** (01/10/04). MIT Technology Review. Date accessed: 26/12/17. Retrieved from: https://www.technologyreview.com/s/403095/sir-tim-berners-lee/
[36] Vanhoef, Mathy. **Key Reinstallation Attacks** (05/17). Date accessed: 26/12/17. Retrieved from: https://www.krackattacks.com/
[37] Krebs, Brian. **The lingering mess from default insecurity** (12/11/15). krebsonsecurity.com. Date Accessed: 19/01/18. Retrieved from: https://krebsonsecurity.com/2015/11/the-lingering-mess-from-default-insecurity/
[38] Gibbs, Samuel. **WannaCry: hackers withdraw £108000 of bitcoin ransom** (03/08/17). Date accessed: 26/12/17. The Guardian. Retrieved from: https://www.theguardian.com/technology/2017/aug/03/wannacry-hackers-withdraw-108000-pounds-bitcoin-ransom
[39] Bohorquez Quevedo, Freddy. **Monitoreo de redes MikroTik con SNMP**. Date accessed: 26/12/17. Retrieved from: https://mum.mikrotik.com/presentations/BO14/freddy.pdf
[40] Hay Newman, Lili. **The biggest cybersecurity disasters of 2017 so far** (01/07/17). Wired. Date accessed: 26/12/17. Retrieved from: https://www.wired.com/story/2017-biggest-hacks-so-far/
[41] **Guifi.net. Europe map**. Date accessed: 26/12/17. Retrieved from: https://Guifi.net/node/17714/view/map
[42] MikroTIk Wiki. **Bruteforce login prevention**. Date Accessed: 19/01/18. Retrieved from: https://wiki.mikrotik.com/wiki/Bruteforce_login_prevention
[43] CATNIX. **Catalunya Neutral Internet Exchange**. Date accessed: 26/12/17. Retrieved from: http://www.catnix.net/en/
[44] Fyodor Lyon, Gordon. **NMAP: Scanning the Internet** (2016). Defcon 16. Youtube: nmapvideos. https://www.youtube.com/watch?v=R_vHhEzxYkY
[45] **Internet Defense League**. Date accessed: 26/12/17. Retrieved from: https://www.internetdefenseleague.org/
[46] TAILS. **Privacy for anyone anywhere**. Date accessed: 26/12/17. Retrieved from: https://tails.boum.org/
[47] **Internet Society**. Date accessed: 26/12/17. Retrieved from: https://www.internetsociety.org/
[48] GOV.UK. **Government backed scheme helps train two million people in digital skills** (14/11/17). Date accessed: 26/12/17. Retrieved from: https://www.gov.uk/government/news/government-backed-scheme-helps-train-two-million-people-in-digital-skills
[49] GOV.UK. **New online challenge will test teenagers' cyber security skills** (14/11/17). Date accessed: 26/12/17. Retrieved from: https://www.gov.uk/government/news/new-online-challenge-will-test-teenagers-cyber-security-skills
[50] Dalmau, Lluís. **What is Guifi.net?** (07/06/09). Date accessed: 26/12/17. Retrieved from: https://Guifi.net/en/what_is_Guifinet
[51] MikroTik. **Mikrotik RouterOS catalogue** (2010). Date accessed: 26/12/17. Retrieved from: https://www.mikrotik.com/download/pdf/what_is_routeros.pdf
[52] Stack Overflow. **Automatically enter SSH password with script** (2012). Date accessed: 26/12/17. Retrieved from: https://stackoverflow.com/questions/12202587/automatically-enter-ssh-password-with-script
[53] Guifi.net. **Zone and zone parent(s) network allocation(s)**. Date accessed: 26/12/17. Retrieved from: https://Guifi.net/ca/node/2413/view/ipv4
[54] Zivtech. **The benefits of open source software** (19/05/15). Date accessed: 26/12/17. Retrieved from: https://www.zivtech.com/blog/benefits-open-source-software
[55] Gibson Research Corporation. **Port 81**. Date Accessed: 19/01/2018. Retrieved from: https://www.grc.com/port_81.htm

[56] Broxkmeier, Joe. **Beginner's Guide to NMAP** (03/03/2010). Linux.com. Date Accessed: 19/01/2018 Retrieved from: https://www.linux.com/learn/beginners-guide-nmap
[57] Ubiquiti Community Blog. Search: "released". Retrieved from: https://community.ubnt.com/t5/forums/searchpage/tab/message?filter=location&q=has+benn+released&location=blog-board%3ABlog_airMAX&page=5&collapse_discussion=true
[58] Fyodor. Defcon 18: **Mastering the NMAP scripting engine** (12/12/2010). YouTube. Date Accessed: 19/01/2018. Retrieved from: https://www.youtube.com/watch?v=gVJHCGfm-dI
[59] MikroTik. Manual: **Securing your router**. Date accessed: 26/12/17. Retrieved from: https://wiki.mikrotik.com/wiki/Manual:Securing_Your_Router#Access_to_a_router
[60] **Easy map maker**. Date accessed: 26/12/17. Retrieved from: https://www.easymapmaker.com/
[61] MikroTik. **Upgrading RouterOS**. Date accessed: 26/12/17. Retrieved from: https://mikrotik.com/download
[62] MikroTik. **All current and historical releases**. Date accessed: 26/12/17. Retrieved from: https://mikrotik.com/download/archive
[63] MikroTik. **Index of routerOS 2.9.51** (2008). Date accessed: 26/12/17. Retrieved from: http://www.mikrotik.com.ua/download/routeros/routeros-all-2.9.51/
[64] OIDView. **MikroTik MIB**. Date accessed: 26/12/17. Retrieved from: http://www.oidview.com/mibs/14988/MIKROTIK-MIB.html
[65] MikroTik forum. **Get current firmware version using SNMP** (31/12/2012). Date accessed: 26/12/17. Retrieved from: https://forum.mikrotik.com/viewtopic.php?t=68596
[66] Solikom Dokuwiki. **AirOS de Ubiquiti. Nodo cliente** (27/11/2011). Date Accessed: 19/08/19. Retrieved from: https://roure.act.uji.es/wiki/public/guifinet/cursoinstaladoresguifi2011/airos/start
[67] **Ubiquiti AirMAX ac**. Retrieved from: https://www.ubnt.com/download/airmax-ac
[68] Ubiquiti Support. **What is the default username \ password for UAPs and controller?** (29/12/2017). Date Accessed: 17/01/2018. Retrieved from: https://help.ubnt.com/hc/en-us/articles/204909374-UniFi-What-is-the-default-username-password-for-UAPs-and-controller-
[69] **Linux scp command** (29/12/2017). Computer Hope. Date accessed: 12/01/17. Retrieved from: https://www.computerhope.com/unix/scp.htm
[70] Ubiquiti Community. **SNMP oid list** (08/12/012). Date Accessed: 19/01/2018. Retrieved from: https://community.ubnt.com/t5/airOS-Software-Configuration/Snmp-oid-list/td-p/310870
[71]Fyodor. Defcon 13: **NMAP hacking** (07/02/2014). YouTube. Date Accessed: 19/01/2018. Retrieved from: https://www.youtube.com/watch?v=V2BXeMTfrt0
[72] UnSolClic **Script for RouterOSv5.x.** Date accessed: 04/01/18. Retrieved from: https://Guifi.net/es/Guifi/device/4232/view/unsolclic
[73] **Nodo: MARCanParera** (02/04/2007). Date accessed: 04/01/18. Retrieved from: https://Guifi.net/es/Guifi/device/4232
[74] Wikipedia. **Internet Service Provider** (04/01/2018). Date accessed: 05/01/18. Retrieved from: https://en.wikipedia.org/wiki/Internet_service_provider
[75] Roger Baig, Ramón Roca, et al. **Guifi.net, a crowdsourced network infrastructure held in common** (2015). Date accessed: 05/01/18. Retrieved from: https://0-www.sciencedirect.com.cataleg.uoc.edu/science/article/pii/S1389128615002327
[76] Wiki Guifi.net. **Unsolclic** (2012). Date accessed: 05/01/18. Retrieved from: http://ca.wiki.Guifi.net/wiki/Unsolclic
[77] Wikipedia. **Guifi.net** (04/12/2017). Date accessed: 05/01/18. Retrieved from: https://es.wikipedia.org/wiki/Guifi.net
[78] Finch, L. **Guifi.net: Spain's wildly successful DIY network** (11/12/2013). Rising Voices. Date accessed: 05/01/18. Retrieved from: https://rising.globalvoices.org/blog/2013/12/11/Guifi-net-spains-wildly-successful-diy-wireless-network/
[79] APCNews. **Guifi.net, APC's new member in Catalonia: Creating free, open and neutral telecoms networks** (11/05/2017). Association for Progressive Communications (APC). Date accessed: 05/01/18. Retrieved from: https://www.apc.org/en/news/Guifinet-apcs-new-member-catalonia-creating-free-open-and-neutral-telecoms-networks
[80] Tabakalera. **Guifi.net mobile node design group** (2016). Date accessed: 05/01/18. Retrieved from: https://www.tabakalera.eu/en/guifinet-mobile-node-design-group
[81] Constantin, Lucian. **How to protect your home router from attacks** (03/01/2018). Motherboard. Date accessed: 11/01/18. Retrieved from: https://motherboard.vice.com/en_us/article/9kn3g7/how-to-protect-your-home-router-from-attacks
[82] Constantin, Lucian. **Unpatched vulnerability puts Ubiquiti networking products at risk** (16/03/2017). PC World. Date accessed: 11/01/18. Retrieved from: https://www.pcworld.com/article/3181833/security/unpatched-vulnerability-puts-ubiquiti-networking-products-at-risk.html
[83] Cimpanu, Catalin. **Tens of thousands of defaced MikroTik and Ubiquiti routers available online** (10/01/2018). Bleeping Computer. Date accessed: 11/01/18. Retrieved from: https://www.bleepingcomputer.com/news/security/tens-of-thousands-of-defaced-mikrotik-and-ubiquiti-routers-available-online/

[84] Symantec Security Response. **Thousands of Ubiquiti AirOS routers hit with worm attacks** (19/5/16). Symantec Security Blog. Date Accessed: 19/01/18. Retrieved from: https://www.symantec.com/connect/blogs/thousands-ubiquiti-airos-routers-hit-worm-attacks
[85] MikroTik documentation. Manual: TOC. Retrieved from: https://wiki.mikrotik.com/wiki/Manual:TOC
[86] Sherlock, Nicholas. **Login bypass in Ubiquiti airMAX/airOS before 8.0.2, 7.2.5, 6.0.2, 5.6.15 if airControl web-UI was used** (17/08/17). nichsherlock.com. Date Accessed: 19/01/18. Retrieved from: http://www.nicksherlock.com/2017/08/login-bypass-in-ubiquiti-airmax-airos-if-aircontrol-web-ui-was-used/

7. Bibliography

- Barrass, Robert (1978). **Scientists must write**. Editorial Chapman and Hall.
- Johnson, Chris (et al.) (2015). **Pro Bash Programming**. Editorial Apress.
- Odom, Wendell. **CCENT/CCNA. ICND 100-105** (05/2017). Cisco Press.
- **A Short Guide to Writing your Final Year Project Report or MSc Dissertation**. Retrieved from: https://www.cs.cf.ac.uk/PATS2/wiki/lib/exe/fetch.php?media=project-report.pdf Universitat Internacional de Catalunya. 2017.
- **Preparing for your Final Degree Project**. Cardiff University. School of Computer Science and Informatics. February 2011. Retrieved from: http://www.uic.es/en/library/services/training/preparing-final-project
- **Attribution ShareAlike 4.0 International**. Date accessed: 19/11/17. Retrieved from: https://creativecommons.org/licenses/by-sa/4.0/
- **Best practises for attribution**. Creative Commons. Date accessed: 19/11/17. Retrieved from: https://wiki.creativecommons.org/wiki/Best_practices_for_attribution
- **Choose an open source license**. Creative Commons Attributions Share Alike 4.0. Date accessed: 19/11/17. Retrieved from: https://choosealicense.com/licenses/cc-by-sa-4.0/
- **Proyecto de fin de carrera**. Wiki Guifi.net. Date accessed: 07/12/17. Retrieved from: http://es.wiki.guifi.net/wiki/Proyecto_de_Fin_de_Carrera#T.C3.ADtulo:_.22Guifi.net:_a_Security_Analysis_of_a_Heterogeneous_Community_Network..22

8 Appendices

8.1. Project management

This section covers the whole project management process. It's a follow-up from the Introduction, where the scope and objectives were presented. It begins with the planning, leading to the monitoring and control phases. The last section deals with the closing process.

8.1.1. Planning

Planning encompasses aspects that are needed in order to tackle specific tasks and possible problems. Thus, this section includes a scheduling of deliveries and milestones, a risk assessment, and the list of resources to be used. The project success depends upon the execution of these actions prior to start its development.

DELIVERABLES AND MILESTONES. SMART is a tool that helps setting objectives. In previous sections four of the five SMART goals have already been justified and explained. In addition, we introduce here the traceability.

- **Specificity**. The project is centred on analysing security aspects of Guifi.net, plus how users, as citizens, are affected by them.

- **Measurability**. Community networks exist to bring Internet to people in need. Simultaneously, networks exhibit a lot of vulnerabilities. Therefore, both network and users are at risk. While the network inherently poses vulnerabilities to users, these need the services the Internet offers to thrive both socially and economically.

- **Achievability**. The focus is placed on key software that deploys important network functions. This restraint makes the project scope both specific and achievable.

- **Relevance**. As a community network and the impact Guifi.net makes on people, the study of the security options it brings about is pertinent.

- **Traceability**. The project is capped by the remaining 15 weeks before its closing. As part of a university course, the hours assigned per week are roughly 25. Considering its 12 credits, it turns out to be a total of 300 hours to execute it.

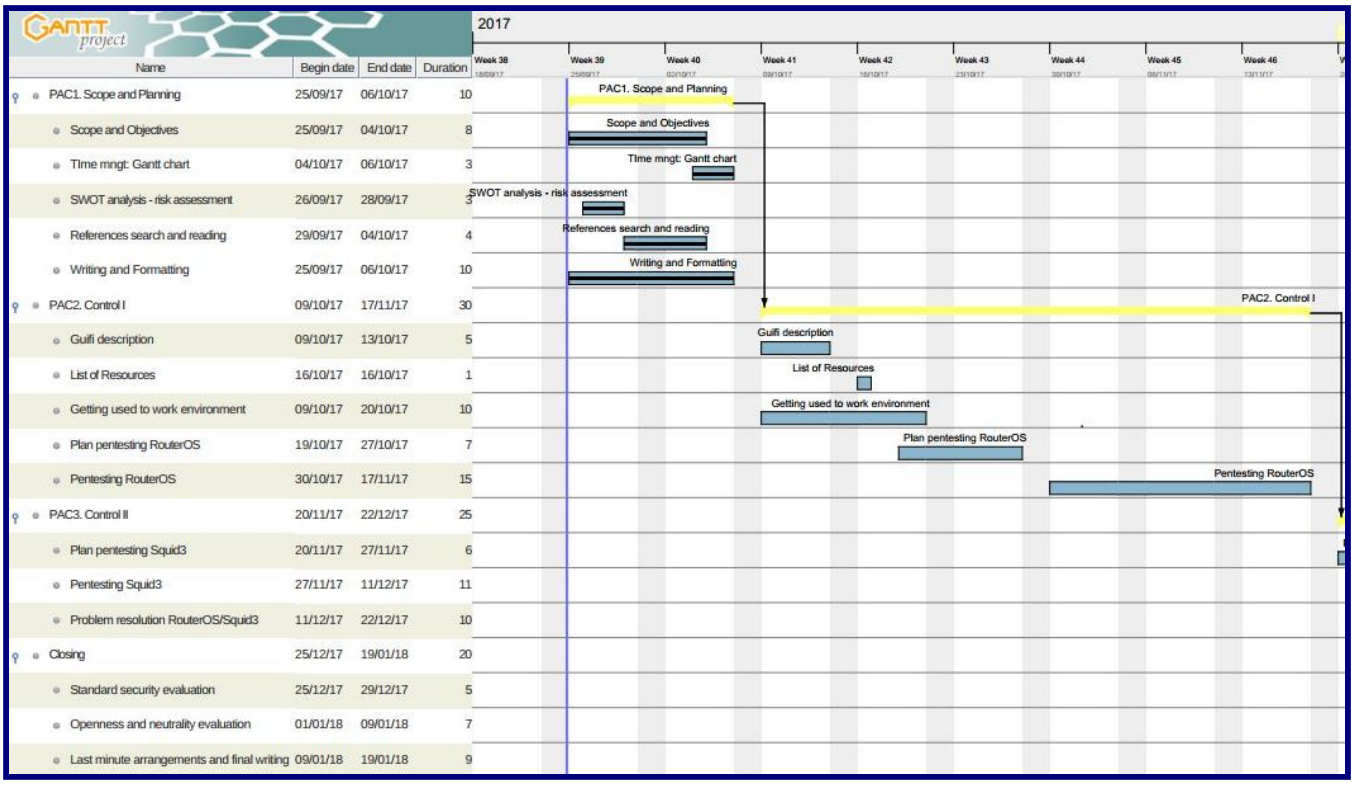
Intentionally leaving out this math the time spent on the planification, in the period 9/10/17-19/01/18 there are 20 hours of work per week (5 ½ hours per day), Monday to Friday. Despite this calculation, and following the SWOT assessment results, any additional time or effort needed will be considered in order to achieve the project goals.

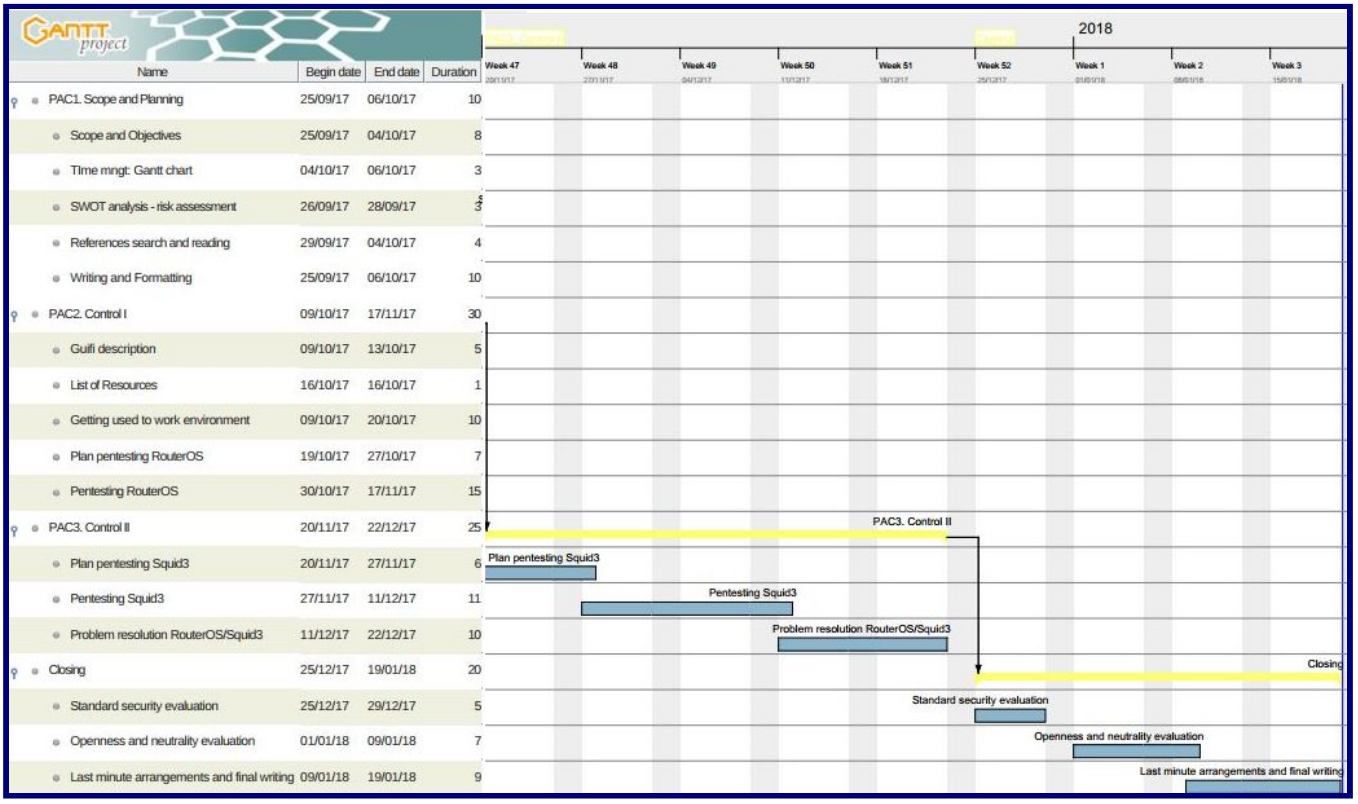
The **key milestones** are highlighted in the calendar shown below:

i. **6 October**: PAC1 assignment delivery day. Although the official date was scheduled on 2 October, there was an initial delay of four days.
ii. **9 October**: Beginning of the execution phase.
iii. **20 November**: 1st follow-up or control assignment (PAC2).
iv. **22 December**: 2nd follow-up or control assignment (PAC3).
v. **19 January**: Final project delivery date.

Image retrieved from this web.

Finally, a **Gantt chart** [18] is a project management tool that helps make and adjust a schedule. Failing to have a broad perspective on the activities and milestones needed to achieve the project success, will limit the response in case unforeseeable situations or problems arise.


GANTT project
2017
Name
Begin date
End date
Duration
PAC1. Scope and Planning 25/09/17 06/10/17 10
Scope and Objectives 25/09/17 04/10/17 8
TIme mngt: Gantt chart 04/10/17 06/10/17 3
SWOT analysis - risk assessment 26/09/17 28/09/17 3
References search and reading 29/09/17 04/10/17 4
Writing and Formatting 25/09/17 06/10/17 10
PAC2. Control I 09/10/17 17/11/17 30
Guifi description 09/10/17 13/10/17 5
List of Resources 16/10/17 16/10/17 1
Getting used to work environment 09/10/17 20/10/17 10
Plan pentesting RouterOS 19/10/17 27/10/17 7
Pentesting RouterOS 30/10/17 17/11/17 15
PAC3. Control II 20/11/17 22/12/17 25
Plan pentesting Squid3 20/11/17 27/11/17 6
Pentesting Squid3 27/11/17 11/12/17 11
Problem resolution RouterOS/Squid3 11/12/17 22/12/17 10
Closing 25/12/17 19/01/18 20
Standard security evaluation 25/12/17 29/12/17 5
Openness and neutrality evaluation 01/01/18 09/01/18 7
Last minute arrangements and final writing 09/01/18 19/01/18 9
Week 38
Week 39
Week 40
Week 41
Week 42
Week 43
Week 44
Week 45
Week 46
PAC1. Scope and Planning
Scope and Objectives
TIme mngt: Gantt chart
SWOT analysis - risk assessment
References search and reading
Writing and Formatting
PAC2. Control I
Guifi description
List of Resources
Getting used to work environment
Plan pentesting RouterOS
Pentesting RouterOS


GANTT project
2018
Name
Begin date
End date
Duration
PAC1. Scope and Planning 25/09/17 06/10/17 10
Scope and Objectives 25/09/17 04/10/17 8
TIme mngt: Gantt chart 04/10/17 06/10/17 3
SWOT analysis - risk assessment 26/09/17 28/09/17 3
References search and reading 29/09/17 04/10/17 4
Writing and Formatting 25/09/17 06/10/17 10
PAC2. Control I 09/10/17 17/11/17 30
Guifi description 09/10/17 13/10/17 5
List of Resources 16/10/17 16/10/17 1
Getting used to work environment 09/10/17 20/10/17 10
Plan pentesting RouterOS 19/10/17 27/10/17 7
Pentesting RouterOS 30/10/17 17/11/17 15
PAC3. Control II 20/11/17 22/12/17 25
Plan pentesting Squid3 20/11/17 27/11/17 6
Pentesting Squid3 27/11/17 11/12/17 11
Problem resolution RouterOS/Squid3 11/12/17 22/12/17 10
Closing 25/12/17 19/01/18 20
Standard security evaluation 25/12/17 29/12/17 5
Openness and neutrality evaluation 01/01/18 09/01/18 7
Last minute arrangements and final writing 09/01/18 19/01/18 9
Week 47
Week 48
Week 49
Week 50
Week 51
Week 52
Week 1
Week 2
Week 3
PAC3. Control II
Plan pentesting Squid3
Pentesting Squid3
Problem resolution RouterOS/Squid3
Closing
Standard security evaluation
Openness and neutrality evaluation
Last minute arrangements and final writing

RISK ASSESSMENT. SWOT analysis studies the Strength, Weaknesses, Opportunities and Threats a project idea have. That way, it is easier to check its viability from a risk perspective [16]. An interactive tool available at web PYMES [17] offered by the *Ministerio de Economía, Industria y Competitivad* from the Spanish Government has been used as a guide to perform this risk evaluation.

It is important to note that this analysis was done before reading and getting deeper information regarding the project topic (27 September 2017). By 4 October the weakness "*Indeterminació dels continguts del projecte*" does not pose a real risk, although attention should still be given to it, as different circumstances may arise that affect the contents initially planned.

Strengths	Weaknesses	Opportunities	Threats
14 weeks available to spend on project development	Lack of deep knowledge on security networks.	Real project management	Working laptop out of order
All needed technical resources available	Indetermination of real project contents.	Learn new tools	Loss of data
Commitment for good quality work	-	May find out Guifi.net vulnarability/ies	Unexpected problems or situations
Good network knowledge	-		Not Internet connection

As an individual work, where the same person performs all the stages of the project's development—planning, execution, management, and closing—the following aspects become especially important:

- Their personal circumstances.
- The dependency on technical elements to function as intended.
- And the skills needed to gain knowledge through the process, while closing the project with minor setbacks.

As a follow-up step, the previous results are evaluated to figure out new strategies that would apply, should any risky situation arise.

Survival Strategies	Adaptive Strategies	Defensive Strategies	Offensive strategies
Cloud working	Project management	Recover from unforeseeable circumstances	Excellent final report and project
Flexible time management	Continuous learning	-	-
Help from supervisor	-	-	-
Information from Internet resources	-	-	-

The risk assessment helps to highlight the areas that need more attention throughout the development of the project. Moreover, it also takes a further step into planning ahead the strategies to be adopted if necessary. Hence, this is a key planning step that takes prevention as one of its best assets.

8.1.2. Monitoring and control

Monitoring and control is the phase of a project development that follows and simultaneously assesses the execution phase. Basically, this helps the stakeholders to get a picture of the way things are going and adjust to the problematic situations that may arise.

This project may benefit from two assessments: one performed in November and another one in December. Both of them ensure the objectives are being met. Every assessment will include a detailed explanation of the execution to date plus comments about problems or difficulties encountered that need to be sorted out.

First assessment: November 2017

EXECUTION. This first execution part has multiple tasks: from documentation, to document design and organization, and the first technical activities. When it comes to the documentation, an effort has been made to get as many valuable references as possible to get a good and accurate picture of what Guifi.net is. On the work made on this document, the main target has been to make it appealing to the reader by organising the contents on a comprehensive and easy-to-follow way, and choosing the best layout possible. Next, to the process of learning the testing tools available followed the design of the precise steps to get the data needed to evaluate the security of RouterOS on Guifi.net.

In what follows, three lists with the aforementioned topics will display the tasks carried out throughout this first execution phase.

1. DOCUMENTATION

    a. Check for other thesis, final project documents, and other similar documents to get examples of possible content organisation and layout.
    b. Deepen my knowledge about networks: routable IPs, ISP tasks and tiers, how Guifi.net routes traffic, and the current neutrality and openness state of networks.
    c. Work on the understanding of Guifi.net: IPs used, VPN connections, SSH connections, subnets, and general topology.
    d. Look for a suitable copyright license for the project.
    e. Revision of proxies: utilities and use.
    f. Learn scanning techniques with NMAP.
    Get deeper knowledge regarding key scanning protocols like TCP or ICMP.
    g. Get used to a Unix environment and revise some useful commands.
    h. Get information about MikroTik routers: their functionalities and characteristics of their RouterOS —especially the current firmware versions. Compare it with Ubiquiti routers from different sources to make sure the assumption of MikroTik dominating the backbone is not wrong.
    i. Revision of SNMP concepts.
    j. Learn scripting for Unix: getting familiar with commands such as grep, awk, cat, and sed.

2. DOCUMENT DESIGN

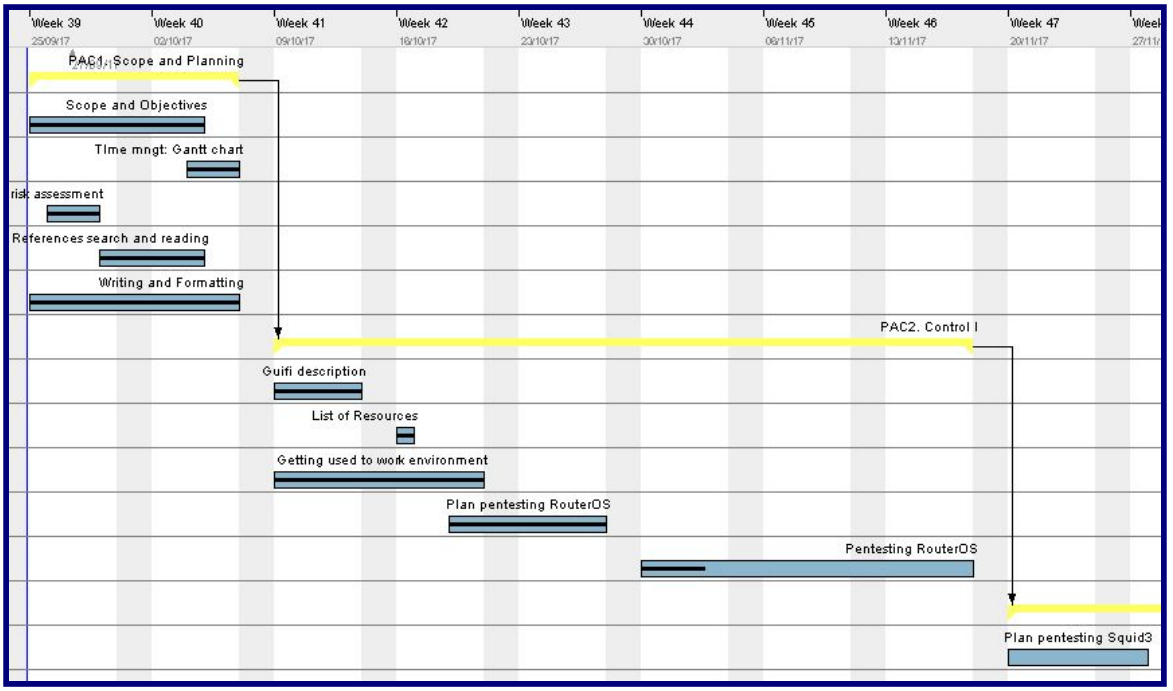
    a. Try to find a way to keep the contents of the first PAC on all the subsequent versions of the project document. The idea is that all the work done is reflected on a single document. Finally, it seems a good idea to keep the project scope and objectives visible, while moving the rest of it to the appendices under the headline “Project Management”. This is where sections like monitoring and control are located.

b. Structure the chapters and sections in a way that can be easily followed. Special care has been taken in maintaining the flow of introduction, development, and closing. To that end, the following decisions were taken with regard to the current document configuration:
    - a first introductory chapter that explains the project's goals and scope.
    - a second chapter that introduces what Guifi.net is from a variety of perspectives with the objective of setting the basis for a later analysis. Special emphasis is given to the concepts of neutrality and openness, as they play a key role on the final security evaluation.
    - A third chapter that includes the description of the project's technical aspects. There are several sections: approach, scope, and implementation.
    - the rest of chapters include the analysis of the results from chapter 3, an analysis on how this affects both neutrality and openness of networks like Guifi.net, and the list of references and bibliography.
    - appendices: as an addition to the main document.
c. The selection of colours, heading design, beginning of chapters, and font type and size (Arial, 11) for the body of the text, and line spacing have all been chosen to facilitate easy reading.
d. For the technical contents, the best approach is to explain the process and the results in chapter 3, but deferring the scripts and any other technical documentation to the appendices.
e. Adaptation of the introduction to the new layout of the project and improving its contents to make them clearer.
f. Writing of the Chapter 2.
g. Referencing the text and completing the references section.
h. Add links to the document for easy access to any section and between sections.

3. TECHNICAL DEVELOPMENT

a. Download of OpenVPN for Windows and installation.
b. Use Ubuntu VM to connect to Guifi.net via VPN, just in case it added more control or easy of use. However, it was proven that the VM sees the VPN from Windows; probably because the VM network configuration is NAT.
c. Use PuTTy for Windows as a tool for connecting to the UOC server through SSH.
d. First connections VPN and SSH and familiarity with the environment.
e. With Internet access: update and upgrade of the server plus installation of utilities such as NMAP, SNMP, or LYNX.
f. Download and practise with CHR on Virtualbox. This is potentially useful if it is needed to use the RouterOS terminal from 10.90.224.65 some time —via http.
g. Execution of different NMAP scans to hosts on the same subnet (10.90.224.64/27) to choose the most efficient set of flags and guarantee MikroTik devices can be identified with the scanning.
h. Start writing a script that parses a document that contains the results of the previous scanning.

PROBLEMS ENCOUNTERED AND WAYS TO IMPROVE. A good way of knowing whether the objectives are being met is by checking the Gantt chart (below). From the initial arrangements and time distribution, by 20 November, the project should have been finished the task of testing the RouterOS. However, that sections is currently not being fully accomplished.



This delay is almost fully explained by two facts:

1. The Guifi.net description was meant to be just that: a description. However, after the planning was finished and the actual contents of the project were considered, there was the need to write a full chapter that worked as a foundation to Guifi.net. With all the relevant information provided at the very start, it is possible to follow every step of the project, including the final discussions on security.
   This added a lot of workload, but it was still considered worth it.
2. A better time-management would help to keep up with the workload. The solution may lie on keeping a constant working flow. This is of especial importance when facing technical difficulties.

Another tool useful to evaluate the project from a different perspective is the risk assessment. That is, a quick look at the SWOT analysis, and a comparison with the first execution phase will allow for a thorough analysis of the project pros and cons so far.

ADVANTAGES	DISADVANTAGES
Determination for good quality work	Lack of deep cybersecurity knowledge
Learn new tools	Undetermined project contents
Practise managing a project	Lot of time available for the project

On the disadvantages, the time needed has been significantly higher than expected. Besides, unexpected technical difficulties requires even more allotted time. Despite those setbacks, there are also some positive points. For example, even though the difficulties found, there have always been determination for achieving the best possible quality work. In addition, a lot of new tools and concepts have been learnt in the process, together with the practice activities on project management.

To sum up, it is inevitable to find some problems and experience difficulties along the way. They can be taken as a way to measure to which point we can manage different situations and succeed at solving problems or overcoming difficulties.

Second assessment: December 2017

EXECUTION. This new execution phase has had at its core objectives the development of the RouterOS script and the improvement of the project report. As for these and other tasks performed during the second execution phase, they are all stated on the lists: Documentation, Document Design and Contents, and Technical Development.

1. DOCUMENTATION

a. Shell Scripting book to help with the basic programming skills.
b. Update on neutrality contents after the FCC repeal of net neutrality on US.
c. Look for the most used (% over total host IPs available) network regions in Guifi.net.

2. DOCUMENT DESIGN AND CONTENTS

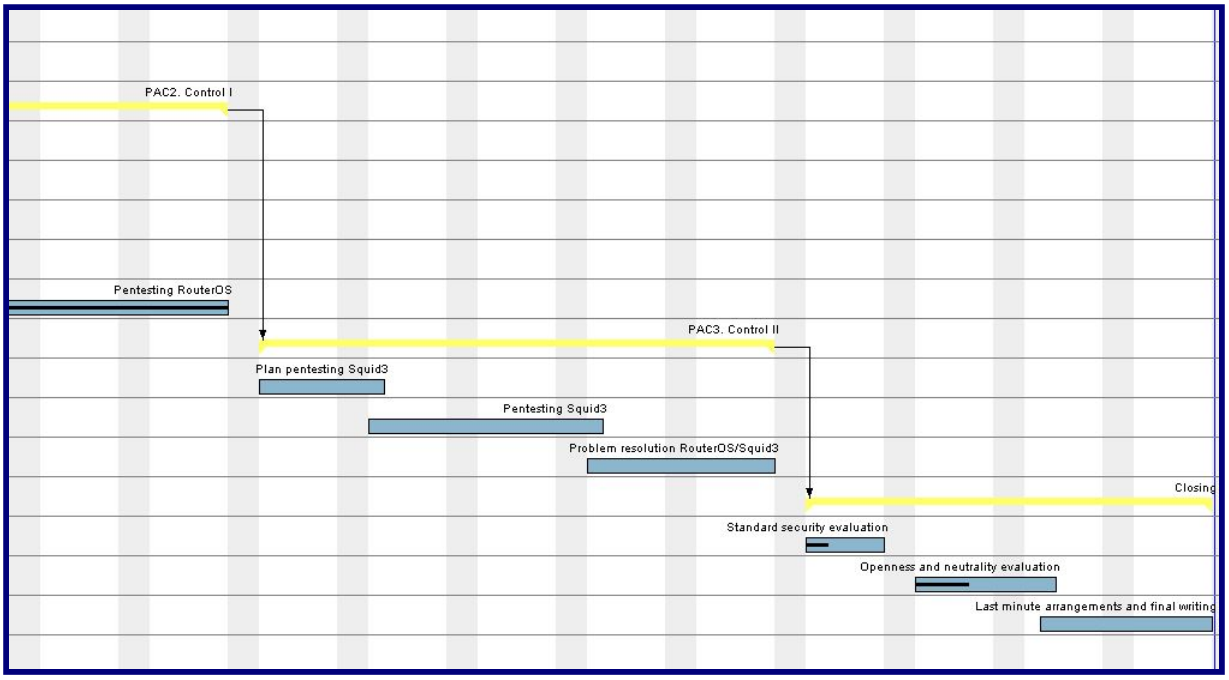
a. Modification of table of contents and references to a smaller font size and in columns.
b. Addition of an appendix section with all acronyms used.
c. Brainstorming of ideas to include in chapters 5 and 6.
d. Addition of figures to the appendix section “detailed scenario setup guide”.
e. Addition of subsections in chapter 2 to help the reader grasp the ideas of the text before reading.
f. Addition of references and content improvement in Chapter 3.
g. Change in “technology in use” with the use of a table for the sake of simplicity.
h. Abstract writing.
i. Explanation of the functionalities of the routerOS script with accompanied figures in Chapter 3.
j. Added more links on the report that were missing. Now it is possible to jump from “table of contents” to any heading and back to it.
k. Map creation with the geolocation of the towns to scan.

3. TECHNICAL DEVELOPMENT

a. Download of FileZilla for Windows and install to send/receive files to/from VM on Guifi.net.
b. RouterOS script writing, debugging, and testing. CHANGE: although in the beginning the idea was to write several scripts, I think writing just one is a more automated approach, especially when having to run the same script several times. Yet, it seems possible to call other scripts from the one executed…
c. Making script executable from any directory by using $HOME/bin and adding the route to $PATH.

PROBLEMS ENCOUNTERED AND WAYS TO IMPROVE. The time spent on programming the routerOS script has paid off with minor setbacks and problems. That means most of the time its development has been steady. Despite that, the project is behind the initial schedule (Gantt chart

below). Partly it is due to the already highlighted increased weight and importance of the first introductory part on Guifi.net that affected the content scope on the first deliverable.



Given that delay, it is time to plan ahead the work to do for the remaining time (cerca 3 weeks). In that period, it is advisable to shrink the dedication and scope of Squid, as it is preferable to have a neat and cohesive report than many run tests without further analysis. Besides, there are some activities towards the end of the project that require dedication that have not been taken into account on the Gantt chart: video presentation and final assessment. Apart from that, there are some activities that may take an undetermined amount of time, which are the final review, the last-minute arrangements, and any possible delayed tasks.

For that matter, it may help understand the risk of this new situation reviewing the SWOT analysis performed at the beginning of the project. It can be seen that time is no longer a strength. However the commitment for good quality work is still present, as well as a working routine has been developed.

8.1.3. Closing

LAST-MINUTE PROBLEMS.

Most of the problems encountered have already been mentioned in the main text of this memoir. The most time-consuming part of the project was not the scanning of the routers itself but the writing of the necessary data analysis scripts to extract the useful information from the humongous output files of NMAP.

In the last few days before the deadline for submission of this project, the scripts originally conceived for the analysis of the Ubiquiti devices had to be entirely re-written. As mentioned in the main text, this was due to the fact that the proposed analysis started from an SNMP scan, which made a large number of devices with port 161/SNMP closed go entirely unnoticed, thus not returning enough information for a meaningful statistical analysis. A new script, very similar to the one employed for the analysis of the MikroTik devices, had to be eventually prepared.

Another major last-minute drawback was the lack of detailed information about the year of release of the myriad of types of Ubiquiti devices. As already commented, this partly invalidated relevant parts of the security analysis for this type of devices.

As a direct consequence of that, both Squid and a review of Internet neutrality had to be left out. In turn, the idea of adding Ubiquiti devices to the project objectives came up throughout its development, as it proved appropriate giving the equal relevance in Guifi.net and somewhat identical testing procedures.

SUCCESS STORY?

As argued in Chapter 5, in spite of the unexpected problems and inherent limitations of this project, all its main goals were met. Beyond being a mere academic exercise for an end-of-degree project, this memoir could shed light on how to actually strengthen the security policies in Guifi.net (namely, placing the focus on keeping firmware updated). The inclusion of a detailed discussion of Guifi.net: its origins, development, and architecture, also provides an added value that a wide readership might encounter instructive.

8.2. Scripts

The scripts coded for this project are shown below as a reference or for further improvement or development. The first script is called **RouterOS tests** and scans set of IP addresses, searching for MikroTik devices basic security settings. An adaptation of this script is done to look for Ubiquiti devices. The script is called **AirOS tests**.

As a following step to successfully arrange the resulting data, the scripts **RouterOS data** and **AirOS data** are programmed for RouterOS and AirOS outputs, respectively.

8.2.1. RouterOS tests

```
#! /bin/bash

#: Title       : RouterOS tests
#: Date        : January 2018
#: Author      : "Patricia Ramos García" <patram@uoc.edu>
#: Version     : 1.10
#: Description : Analysis over a region range of IP addresses in order to determine which ones
#                belong to Mikrotik devices. It then makes a list with the open ports, tries
#                to establish a SSH connection with default credentials, & checks their
#                current firmware using SNMP.
#: Sections    : 1.  Launch of NMAP test over a selection of the top TCP ports.
#                2.  IP selection of Mikrotik devices.
#                3.  MikroTik devices open ports and IPs
#                4.  Login attempt over SSH/22
#                5.  Check of current firmware using SNMP.
#: Options     : None
#: Arguments   : File with IP addresses to scan
#                Name of region

################################################################################################

##-------------------------------------------------------------##
## Update to 1 if wish to perform: NMAP, SSH, or SNMP tests   ##
##-------------------------------------------------------------##
       nmap=0
       ssh=0
```

```
        snmp=0
##-------------------------------------------------------------##
##-------------------------------------------------------------##

#################################################################################################

# Variables
fdb="$HOME"/database_firmware.txt   # Database with firmware versions and years

#From arguments:
file=""                             # IPs to scan
path=""                             # Path to new directory

port=0                              # Port number read from file
ip=""                               # IP add read from file

#Functions

port_selection() #@ file_in file_out port_to_search
{
    if [[ $# -le 2 ]]
    then
        printf "\n Not enough input arguments\n"
        return 1
    else
        if [[ -s $1 ]]
        then
            case $3 in
            *[0-65535]*)
                if [[ -f $2 ]]
                then
                    rm $2
                fi

                # Generation of files containing IPs with port
                while IFS=":" read line_num line
                do
                    if [[ -z $( echo $line | cut -d '.' -s -f1 ) ]]
                    then
                        port=$line
                        if [[ "$port" -eq $3 ]]
                        then
                            # Check whether file exists to append or create
                            if [[ -f $2 ]]
                            then
                                printf ""$ip"\n" >>$2
                            else
                                printf ""$ip"\n" >$2
                            fi
                        else
                            continue
                        fi
                    else
                        ip=$line
                    fi
                done <$1

                printf "\n New file %s created with IPs from MikroTik " $2
                printf "devices that have port %d open\n" $3
            ;;
            *)
                printf "\n Port number not valid: "$3"\n"
                return 1
            ;;
            esac
```

```
        else
            printf "\n Searching for port %d open: Input file does not exist or is empty\n" $3
        fi
    fi
}


# User guide in case of error

display_usage ()
{
    printf "\n The usage of the script routerOS is: \n"
    printf "\n "$0" [addresses_file] [scan_region_name] \n"

    printf "\n [addresses_file] contains the list of addresses to analyse.\n"
    printf "\n [scan_region_name] is taken as the saving path for these script files.\n"
}

# First variable handling and dealing with errors
if [[ ( $# -lt 2 ) ]]
then
    printf "\n Missing required information as arguments.\n"
    display_usage
    exit 1
else
    file=$1
    path="$HOME"/"$2"
    mkdir "$path" 2> /dev/null
    if [[ $? -eq 0 ]]
    then
        printf "\n Directory created successfully!\n"
    else
        printf "\n Directory already exists.\n"
    fi
fi



##############################################################################################


### 1.  Launch of NMAP test over a selection of the top TCP ports.

# Variables
nmapout="$path"/nmap_output.txt   # scan result in XML format

# Check whether text file with addresses exists and is non-empty
if [[ ! -s "$file" ]]
then
    printf "\n The file provided either does not exist or is empty\n"
    exit 1
fi

if [[ $nmap -eq 1 ]]
then
    printf "\n Starting NMAP scan with ip_addresses.txt as input\n"

    # NMAP test on all the addresses provided and on a selection of most used ports
    nmap -n -Pn -sV -p21,22,23,53,80,81,82,443,2200 -T4 -iL "$file" -oX "$nmapout"

    # Checking everything went alright
    if [[ $? = 0 ]]
    then
        printf "\n NMAP scan finished successfully; find the results in nmap_output.txt\n"
    else
```

```
        printf "\n Ops... It seems that there are some problems with the scan..."
    fi
else
    printf "\n Skipping NMAP scan this time...\n"
fi

#############################################################################################

### 2.  IP selection of Mikrotik devices.

# Variables
ftempadd="$path"/temp_add.txt       # all scanned addresses
ftempline="$path"/temp_line.txt     # lines where scanned addresses are in nmap_output.txt
fip="$path"/fip.txt                 # combination of the above files
fmikro="$path"/mikro.txt            # lines and ips of MikroTik devices
flinemikro="$path"/lines_mikro.txt  # lines where MikroTik IPs are found


aux_line=0                          # holds the last read IP line
aux_ip=""                           # holds the last read IP address
difference=""                       # each scanned IP takes cerca 15 lines
num_lines=0                         # number lines on $fip
last_line_add=0                     # line where last address is on $ftempadd
last_add=""                         # last address on $tempadd

# Generation of file with lines and ips found in nmap_output.txt
grep -n " addr=" "$nmapout" | cut -d '"' -s -f2 > "$ftempadd"
grep -n " addr=" "$nmapout" | cut -d ':' -s -f1 > "$ftempline"
paste -d: "$ftempline" "$ftempadd" >"$fip"

printf "\n New file temp_add.txt with the scanned addresses.\n"
printf "\n New file temp_line.txt with ip line numbers from file nmap_output.txt.\n"
printf "\n New file fip.txt with all lines and scanned ips.\n"

# Generation of file with lines where mikrotik is found in nmap_output.txt
grep -n "mikrotik" "$nmapout" | cut -d ':' -s -f1 > "$flinemikro"
printf "\n New file lines_mikro.txt with lines where the pattern \"mikrotik\" is found.\n"

# If $fmikro exists, delete its content
if [[ -f "$fmikro" ]]
then
    cp /dev/null "$fmikro"
fi

# Finding lines and ips from MikroTik devices in nmap_output.txt
while read rline <&3 && read ip <&4
do
    while read lmikro
    do
        # Beware variable difference depends on NMAP scan
        let difference=$lmikro-$rline
        if [[ "$rline" -lt "$lmikro" && "$difference" -le 16 ]]
        then
            aux_ip="$ip"
            aux_line="$rline"
            break
        elif [[ "$aux_line" -lt "$lmikro" && "$rline" -gt "$lmikro" ]]
        then
            # Check whether file $fmikro exists to append or create
            if [[ -f "$fmikro" ]]
            then
                printf ""$aux_line":"$aux_ip"\n" >>"$fmikro"
            else
                printf ""$aux_line":"$aux_ip"\n" >"$fmikro"
            fi
```

```
            aux_ip="$ip"
            aux_line="$rline"

            # To avoid saving more than once the same ip
            break
        fi
    done <"$flinemikro"
done 3<"$ftempline" 4<"$ftempadd"


# Case last line is MikroTik

num_lines=$( echo $(wc -l $ftempline) | cut -d ' ' -s -f1 )
last_line_add=$( tail -n +$num_lines $ftempline )
last_add=$( tail -n +$num_lines $ftempadd )

while read mikroline
do
    if [[ "$mikroline" -gt "$last_line_add" ]]
    then
        # Check whether file $fmikro exists to append or create
        if [[ -f "$fmikro" ]]
        then
            printf ""$last_line_add":"$last_add"\n" >>"$fmikro"
            break
        else
            printf ""$last_line_add":"$last_add"\n" >"$fmikro"
            break
        fi
    fi
done <$flinemikro


# Only if $fmikro has been created, inform
if [[ -f "$fmikro" ]]
then
    printf "\n New file mikro.txt with lines and IPs of mikrotik devices.\n"
fi


#############################################################################################


### 3. MikroTik devices, open ports, and IPs

# Variables
ftemport="$path"/temp_ports.txt             # Open ports found on the NMAP scan
fportlines="$path"/port_lines.txt           # Lines where open ports where found
fports="$path"/ports.txt                    # Combination of the above files
flinemikro="$path"/line_ip_mikro.txt        # Lines where MikroTik IPs are on NMAP scan
fipmikro="$path"/ip_mikro.txt               # List of MikroTik IPs
fmikroports="$path"/mikro_ports.txt         # MikroTik IPs and ports, with lines from NMAP scan
fport22open="$path"/port_22open.txt         # MikroTik IPs with port 22 open
fport23open="$path"/port_23open.txt         # MikroTik IPs with port 23 open
fport80open="$path"/port_80open.txt         # MikroTik IPs with port 80 open

aux_ip_line=0                               # Line from last IP read
aux_ip=""                                   # Last IP read
num_lines_flinemikro=0                      # Number of MikroTik devices in NMAP scan
line_num_flinemikro=0                       # Line number last entry $flinemikro
num_lines_ftempline=0                       # All IP lines from NMAP scan
line_num_ftempline=0                        # Line number last entry $ftempline


# Generation of file stating lines of open ports found in nmap_output.txt
```

```
grep -n " portid=" "$nmapout" | grep "open" | cut -d '"' -s -f4 > "$ftemport"
grep -n " portid=" "$nmapout" | grep "open" | cut -d ':' -s -f1 > "$fportlines"
paste -d: "$fportlines" "$ftemport" >"$fports"

printf "\n New file temp_port.txt with the scanned open ports.\n"
printf "\n New file temp_line.txt with open port line numbers from file nmap_output.txt.\n"
printf "\n New file ports.txt with all lines and open ports.\n"

# Generation of file mikro_ports.txt containing MikroTik lines, IPs, and ports
if [[ -f "$fmikro" ]]
then
    cut "$fmikro" -d ':' -s -f1 >"$flinemikro"
    cut "$fmikro" -d ':' -s -f2 >"$fipmikro"

    # If $mikroports exists, delete its content
    if [[ -f "$fmikroports" ]]
    then
        rm "$fmikroports"
    fi

    # Finding MikroTik ips and their open ports

    # Number of lines and last entry of flinemikro and ftemplines
    num_lines_flinemikro=$( echo $(wc -l "$flinemikro") | cut -d ' ' -s -f1 )
    line_num_flinemikro=$(tail -n +"$num_lines_flinemikro" "$flinemikro")

    num_lines_ftempline=$( echo $(wc -l "$ftempline") | cut -d ' ' -s -f1 )
    line_num_ftempline=$(tail -n +"$num_lines_ftempline" "$ftempline")

    # Read line number and MikroTik IP
    while read mikro_line <&3 && read mikro_ip <&4
    do
        # Check whether file exists to append or create
        if [[ -f "$fmikroports" ]]
        then
            printf ""$mikro_line":"$mikro_ip"\n" >>"$fmikroports"
        else
            printf ""$mikro_line":"$mikro_ip"\n" >"$fmikroports"
        fi

        # Case MikroTik IP as last and only entry
       if [[ "$line_num_flinemikro" -eq "$line_num_ftempline" && "$num_lines_flinemikro" -eq 1
]]
       then
           while read line_port <&5 && read port <&6
           do
               if [[ "$line_port" -gt "$line_num_flinemikro" ]]
               then
                   printf ""$line_port":"$port"\n" >>"$fmikroports"
               fi
           done 5<"$fportlines" 6<"$ftemport"

        # Case MikroTik IP as last entry from many
        elif [[ "$line_num_flinemikro" -eq "$line_num_ftempline" && "$num_lines_flinemikro"
-gt 1 ]]
        then
            while read ip_line
            do
                if [[ "$ip_line" -gt "$mikro_line" ]]
                then
                    while read line_port <&5 && read port <&6
                    do
                        if [[ "$line_port" -gt "$mikro_line" && "$line_port" -lt "$ip_line" ]]
                        then
                            printf ""$line_port":"$port"\n" >>"$fmikroports"
```

```
                    elif [[ "$line_port" -lt "$mikro_line" ]]
                    then
                        continue
                    else
                        break
                    fi
                done 5<"$fportlines" 6<"$ftemport"
                break
            fi
        done <"$ftempline"

        # Last MikroTik IP ports
        while read line_port <&5 && read port <&6
        do
            if [[ "$line_port" -gt "$line_num_flinemikro" ]]
            then
                printf ""$line_port":"$port"\n" >>"$fmikroports"
            fi
        done 5<"$fportlines" 6<"$ftemport"

    # Case not MikroTik IP as last entry, independently of number IPs
    elif [[ "$line_num_flinemikro" -ne "$line_num_ftempline" ]]
    then
        while read ip_line
        do
            if [[ "$ip_line" -gt "$mikro_line" ]]
            then
                while read line_port <&5 && read port <&6
                do
                 if [[ "$line_port" -gt "$mikro_line" && "$line_port" -lt "$ip_line" ]]
                    then
                        printf ""$line_port":"$port"\n" >>"$fmikroports"

                    elif [[ "$line_port" -lt "$mikro_line" ]]
                    then
                        continue
                    else
                        break
                    fi
                done 5<"$fportlines" 6<"$ftemport"
                break
            fi
        done <"$ftempline"
    fi
done 3<"$flinemikro" 4<"$fipmikro"

#! NOTE ON $fmikroports: OPEN PORTS EXCLUSIVELY; FILTERED ONES NOT CONSIDERED !#
printf "\n New file mikro_ports.txt with MikroTik lines, ips, and ports.\n"
else
printf "\n Skipping open port analysis because there are no MikroTik devices.\n"
fi

# Search for MikroTik IPs with port 22/SSH open
port_selection "$path"/mikro_ports.txt "$path"/port_22open.txt 22

# Search for MikroTik IPs with port 23/Telnet open
port_selection "$path"/mikro_ports.txt "$path"/port_23open.txt 23

# Search for MikroTik IPs with port 80/HTTP open
port_selection "$path"/mikro_ports.txt "$path"/port_80open.txt 80

##############################################################################################


### 4.  Login attempt over SSH/22
```

```
# Variables
fssh22ip="$path"/ssh22_ips.txt       # MikroTik IPs with default credentials on port 22


# SSH scan on MikroTik devices with port 22 open
if [[ ( $ssh -eq 1 ) && ( -f "$fport22open") ]]
then
    printf "\n Starting SSH login attempt using found IPs with port 22 open\n"

    # If $mikro exists, delete its content
    if [[ -f "$fssh22ip" ]]
    then
        rm "$fssh22ip"
    fi

    while read ip_add
    do
        # Perform the SSH login attempt on the addresses provided by port_22open.txt
        sshpass -p "" ssh -o StrictHostKeyChecking=no admin@"$ip_add"

        # Checking everything went alright
        if [[ $? = 0 ]]
        then
            if [[ -f "$fssh22ip" ]]
            then
                printf ""$ip_add"\n" >>"$fssh22ip"
            else
                printf ""$ip_add"\n" >"$fssh22ip"
            fi
        fi
    done <"$fport22open"

    if [[ -f "$fssh22ip" ]]
    then
        printf "\n New file ssh22_ips.txt with listing all IPs with default credentials.\n"
    else
        printf "\n The SSH scan returned 0 MikroTik devices with default credentials.\n\n"
    fi
else
    printf "\n Skipping SSH login either because not active or zero MikroTik IPs\n\n"
fi



#############################################################################################


### 5.  Check of current firmware using SNMP.

# Variables

fsnmpout="$path"/snmp_out.txt       # SNMP scan results
firm="$path"/firmware.txt           # Data analysis over SNMP results

iso=1.3.6.1.4.1.14988.1.1.4.4.0     # OID MikroTik software version
firmware=""                         # Firmware read from snmp results file
ip=""                               # IP read from snmp results file
db_firm=""                          # Firmware read from database
db_year=""                          # Year firmware released read from database
firm1=0                             # Content first part firmware version
firm2=0                             # Content second part firmware version

year_2017=()                        # Devices last updated in 2017
year_2016=()                        # Devices last updated in 2016
```

```
year_2015=()                        # Devices last updated in 2015
more_3years=()                      # Devices not updated more 3 years
unreachable=()                      # Devices with unknown firmware state


# Scan over MikroTik IPs searching for their current firmware version
if [[ "$snmp" -eq 1 && -f "$fipmikro" ]]
then
    # If $snmpout exists, delete its content
    if [[ -f "$fsnmpout" ]]
    then
        rm "$fsnmpout"
    fi

    while read ip_add
    do
        # Perform the SNMP test over the MikroTik IPs
        firmware=$(snmpwalk -c public -v1 "$ip_add" "$iso" | cut -d '"' -s -f2 ) 2> /dev/null

        if [[ -f "$fsnmpout" ]]
        then
            printf ""$ip_add":"$firmware"\n" >>"$fsnmpout"
        else
            printf ""$ip_add":"$firmware"\n" >"$fsnmpout"
        fi
    done <"$fipmikro"

    #! NOTE ON $fsnmpout: IPs WITHOUT ASSIGNED CURRENT FIRMWARE WERE UNREACHABLE !#
    printf "\n New file snmp_out.txt which contains a list of MikroTik IPs "
    printf "and their current firmware.\n"

else
    printf "\n Skipping SNMP test this time. Either because the option has "
    printf "not been activated or because there are not MikroTik devices.\n"
fi


# Parse of the information provided by $fsnmpout and $fdb

#First check $firm is empty:
if [[ -f "$firm" ]]
then
    rm $firm
fi

if [[ -f "$fsnmpout" ]]
then
    while IFS=":" read ip firmware
    do
        firm1=$( echo $firmware | cut -d '.' -f1 )
        firm2=$( echo $firmware | cut -d '.' -f2 )

        if [[ -z "$firmware" ]]
        then
            unreachable+=("$ip")
        elif [[ "$firm1" -lt 6 || "$firm1" -eq 6 && "$firm2" -lt 25 ]]
        then
            more_3years+=("$ip")
        else
            while IFS=":" read db_firm db_year
            do
                if [[ "$firmware" = "$db_firm" ]]
                then
                    case $db_year in
                        2015) year_2015+=("$ip")
```

```
                            break
                        ;;
                        2016) year_2016+=("$ip")
                            break
                        ;;
                        2017) year_2017+=("$ip")
                            break
                        ;;
                    esac
                else
                    continue
                fi
            done <"$fdb"
        fi
    done <"$fsnmpout"
fi

# Generation of a file containing all MikroTik devices with year of last update
if [[ -f "$fsnmpout" ]]
then
    # If $firm exists, delete its content
    if [[ -f "$firm" ]]
    then
        rm "$firm"
    fi

    # Unreachable IPs
    printf "Unreachable:\n" >"$firm"

    if [[ ! -z "$unreachable" ]]
    then
        printf "%s\n" ${unreachable[*]} >>"$firm"
    fi

    # IPs updated 2017
    printf "2017:\n" >>"$firm"

    if [[ ! -z "$year_2017" ]]
    then
        printf "%s\n" ${year_2017[*]} >>"$firm"
    fi

    # IPs updated 2016
    printf "2016:\n" >>"$firm"

    if [[ ! -z "$year_2016" ]]
    then
        printf "%s\n" ${year_2016[*]} >>"$firm"
    fi

    # IPs updated 2015
    printf "2015:\n" >>"$firm"

    if [[ ! -z "$year_2015" ]]
    then
        printf "%s\n" ${year_2015[*]} >>"$firm"
    fi

    # IPs updated more than 3 years ago
    printf "<2015:\n" >>"$firm"

    if [[ ! -z "$more_3years" ]]
    then
        printf "%s\n" ${more_3years[*]} >>"$firm"
    fi
```

```
    printf "\n New file firmware.txt with a list of MikroTik IPs "
    printf "and year of last firmware update.\n\n"
fi

printf "\n END OF SCRIPT: RouterOS tests.\n\n"
```

8.2.2. RouterOS data

```
#! /bin/bash

#: Title       : RouterOS data
#: Date        : January 2018
#: Author      : "Patricia Ramos García" <patram@uoc.edu>
#: Version     : 1.0
#: Description : Parse of data from files resulting from "routerOS test" script
#: Options     : None
#: Arguments   : None

###################################################################################################
###################################################################################################

# Variables
pathdir=""                          # Path to directories in $HOME
pathbin=$HOME/bin                   # Path to relevant files in bin
fdirectory=$pathbin/directories.txt # List of regions directories
files=$pathbin/routerOS_files.txt   # List of files to parse
foutdata=$pathbin/routerOS_data.txt # Output data

file=""         # File to read within a specific directory
ssh=0           # Number IPs with default SSH credentials
port23=0        # Number IPs with port 23 open
port22=0        # Number IPs with port 22 open
port80=0        # Number IPs with port 80 open
fip=0           # Number IPs scanned
mikro=0         # Number IPs MikroTik
y2017=0         # Number IPs updated in 2017
y2016=0         # Number IPs updated 2016
y2015=0         # Number IPs updated 2015
other=0         # Number IPs updated before 2015

# Flags
seventeen=0
sixteen=0
fifteen=0
less=0

# Case foutdata exists, delete
if [[ -f $foutdata ]]
then
    rm $foutdata
fi

while read diread
do
    pathdir=$HOME/$diread
```

```
    # Case the directory does not exist
    if [[ ! -d $pathdir ]]
    then
        printf "\n The directory %s does not exist.\n" $pathdir
        exit 1
    fi

    # All variables to zero as they are shared with other directories
    ssh=0
    port23=0
    port22=0
    port80=0
    fip=0
    mikro=0
    y2017=0
    y2016=0
    y2015=0
    other=0

    # Read of each data file
    while read fileread
    do
        file=$pathdir/$fileread
        if [[ $fileread = "fip.txt" ]]
        then
               fip=$( wc -l $file | cut -d ' ' -s -f1)
        elif [[ ! -f $file && $fileread = "mikro.txt" ]]
        then
               printf "\n There are no MikroTik devices in the region: %s. \n" $diread
               break
        elif [[ $fileread = "mikro.txt" ]]
        then
               mikro=$( wc -l $file | cut -d ' ' -s -f1)
        elif [[ ! -f $file && $fileread = "ssh22_ips.txt" ]]
        then
               printf "\n No default SSH credentials on routers in: %s.\n" $diread
        elif [[ $fileread = "ssh22_ips.txt" ]]
        then
               ssh=$( wc -l $file | cut -d ' ' -s -f1)
        elif [[ $fileread = "port_22open.txt" ]]
        then
               port22=$( wc -l $file | cut -d ' ' -s -f1)
        elif [[ $fileread = "port_80open.txt" ]]
        then
               port80=$( wc -l $file | cut -d ' ' -s -f1)
        elif [[ $fileread = "port_23open.txt" ]]
        then
               port23=$( wc -l $file | cut -d ' ' -s -f1)
        elif [[ $fileread = "firmware.txt" ]]
        then
               less=0
               while read line
               do
                      if [[ $line = "2017:" ]]
                      then
                             seventeen=1
                      elif [[ $seventeen -eq 1 && $( echo $line | cut -d '.' -s -f1 ) -eq 10
]]
                      then
                             let y2017=$y2017+1
                      elif [[ $line = "2016:" ]]
                      then
                             sixteen=1
                             seventeen=0
```

```
                    elif [[ $sixteen -eq 1 && $( echo $line | cut -d '.' -s -f1 ) -eq 10 ]]
                    then
                        let y2016=$y2016+1
                    elif [[ $line = "2015:" ]]
                    then
                        fifteen=1
                        sixteen=0
                        seventeen=0
                    elif [[ $fifteen -eq 1 && $( echo $line | cut -d '.' -s -f1 ) -eq 10 ]]
                    then
                        let y2015=$y2015+1
                        elif [[ $line = "<2015:" ]]
                    then
                        less=1
                        fifteen=0
                        sixteen=0
                        seventeen=0
                    elif [[ $less -eq 1 && $( echo $line | cut -d '.' -s -f1 ) -eq 10 ]]
                    then
                        let other=$other+1

                    fi
            done <$file
        fi
    done <$files

    # Writing line with data from directory in routerOS_data.txt
    # Variable order: fip mikro port23 port80 port22 ssh y2017 y2016 y2015 other
    if [[ -f $foutdata ]]
    then
        printf "%d %d %d %d %d %d %d %d %d %d\n" $fip $mikro $port23 $port80 $port22 $ssh
$y2017 $y2016 $y2015 $other >>$foutdata
    else
        printf "%d %d %d %d %d %d %d %d %d %d\n" $fip $mikro $port23 $port80 $port22 $ssh
$y2017 $y2016 $y2015 $other >$foutdata
    fi
done <$fdirectory
```

8.2.3. AirOS tests

```
#! /bin/bash

#: Title       : AirOS tests
#: Date        : January 2018
#: Author      : "Patricia Ramos GarcÃƒÂa" <patram@uoc.edu>
#: Version     : 1.6
#: Description : Analysis over a region range of IP addresses in order to determine which
                 ones belong to Ubiquiti routers. It then makes a list with the open
                 ports, tries to establish a SSH connection with default credentials, &
                 checks their current firmware using SNMP.
#: Sections    : 1.  Launch of NMAP test over a selection of the top TCP ports.
#               2.  IP selection of Ubiquiti devices.
#               3.  Ubiquiti devices open ports and IPs
#               4.  Login attempt over SSH/22
#               5.  Check of current firmware using SNMP.
#: Options     : None
#: Arguments   : File with IP addresses to scan
#               Name of region
```

```
#################################################################################################
#################################################################################################

##-------------------------------------------------------------##
## Update to 1 if wish to perform: NMAP, SSH, or SNMP tests   ##
##-------------------------------------------------------------##
       nmap=0
       ssh=0
       snmp=0
##-------------------------------------------------------------##
##-------------------------------------------------------------##

#################################################################################################
#################################################################################################


#From arguments:
file=""                             # IPs to scan
path=""                             # Path to new directory

port=0                              # Port number read from file
ip=""                               # IP add read from file

#Functions

port_selection() #@ file_in file_out port_to_search
{
       if [[ $# -le 2 ]]
       then
           printf "\n Not enough input arguments\n"
           return 1
       else
    if [[ -s $1 ]]
    then
       case $3 in
        *[0-65535]*)
              if [[ -f $2 ]]
              then
                  rm $2
              fi
              # Generation of files containing IPs with port
              while IFS=":" read line_num line
              do
                  if [[ -z $( echo $line | cut -d '.' -s -f1 ) ]]
                  then
                      port=$line
                      if [[ "$port" -eq $3 ]]
                      then
                          # Check whether file exists to append or create
                          if [[ -f $2 ]]
                          then
                              printf ""$ip"\n" >>$2
                          else
                              printf ""$ip"\n" >$2
                          fi
                      else
                          continue
                      fi
                  else
                     ip=$line
                  fi
        done <$1

        printf "\n New file %s created with IPs from Ubiquiti " $2
                     printf "devices that have port %d open\n" $3
```

```
        ;;
        *)
              printf "\n Port number not valid: "$3"\n"
              return 1
        ;;
       esac
    else
        printf "\n Searching for open port %d: Input file does not exist or is empty\n"
$3
    fi
fi
}


# User guide in case of error

display_usage (){
    printf "\n The usage of the script AirOS is: \n"
    printf "\n "$0" [addresses_file] [scan_region_name] \n"

    printf "\n [addresses_file] contains the list of addresses to analyse.\n"
    printf "\n [scan_region_name] is taken as the saving path for these script files.\n"
    }

# First variable handling and dealing with errors
if [[ ( $# -lt 2 ) ]]
then
       printf "\n Missing required information as arguments.\n"
       display_usage
       exit 1
else
       file=$1
       path="$HOME"/bin/airos/"$2"
       mkdir "$path" 2> /dev/null

       if [[ $? -eq 0 ]]
       then
           printf "\n Directory created successfully!\n"
       else
           printf "\n Directory already exists.\n"
       fi
fi



##############################################################################################
##############################################################################################


### 1.  Launch of NMAP test over a selection of the top TCP ports.

# Variables
nmapout="$path"/nmap_output.txt   # scan result in XML format

# Check whether text file with addresses exists and is non-empty
if [[ ! -s "$file" ]]
then
       printf "\n The file provided either does not exist or is empty\n"
       exit 1
fi

if [[ nmap -eq 1 ]]
then
       printf "\n Starting NMAP scan with ip_addresses.txt as input\n"
```

```
        # NMAP test on all the addresses provided and on a selection of most used ports
        nmap -n -Pn -A -p21,22,23,53,80,81,82,443,2200 -T4 -iL "$file" -oX "$nmapout"

        # Checking everything went alright
        if [[ $? = 0 ]]
        then
            printf "\n NMAP scan finished successfully; find the results in
nmap_output.txt\n"
        else
            printf "\n Ops... It seems that there are some problems with the scan..."
        fi
else
        printf "\n Skipping NMAP scan this time...\n"
fi

#############################################################################################

### 2.  IP selection of Mikrotik devices.

# Variables
ftempadd="$path"/temp_add.txt       # all scanned addresses
ftempline="$path"/temp_line.txt     # lines where scanned addresses are in
nmap_output.txt
fip="$path"/fip.txt                 # combination of the above files
fmikro="$path"/mikro.txt            # lines and ips of MikroTik devices
flinemikro="$path"/lines_mikro.txt  # lines where "Ubiquiti" pattern is found

aux_line=0                          # holds the last read IP line
aux_ip=""                           # holds the last read IP address
num_lines=0                         # number lines on $fip
last_line_add=0                     # line where last address is on $ftempadd
last_add=""                         # last address on $tempadd

# Generation of file stating lines and ips found in nmap_output.txt
grep -n " addr=" "$nmapout" | cut -d '"' -s -f2 > "$ftempadd"
grep -n " addr=" "$nmapout" | cut -d ':' -s -f1 > "$ftempline"
paste -d: "$ftempline" "$ftempadd" >"$fip"

printf "\n New file temp_add.txt with the scanned addresses.\n"
printf "\n New file temp_line.txt with ip line numbers from file nmap_output.txt.\n"
printf "\n New file fip.txt with all lines and scanned ips.\n"

# Generation of file stating lines where mikrotik is found in nmap_output.txt
grep -n "=Ubiquiti" "$nmapout" | cut -d ':' -s -f1 > "$flinemikro"
printf "\n New file lines_mikro.txt with lines where the pattern \"ubiquiti\" is
found.\n"

# If $fmikro exists, delete its content
if [[ -f "$fmikro" ]]
then
    cp /dev/null "$fmikro"
fi

# Finding lines and ips from Ubiquiti devices in nmap_output.txt
while read lmikro
do
    while IFS=":" read rline ip
    do
        if [[ "$rline" -lt "$lmikro" ]]
        then
            aux_line=$rline
            aux_ip=$ip
            continue
        elif [[ "$rline" -gt "$lmikro" ]]
        then
```

```
            # Check whether file $fmikro exists to append or create
            if [[ -f "$fmikro" ]]
            then
                printf ""$aux_line":"$aux_ip"\n" >>"$fmikro"
            else
                printf ""$aux_line":"$aux_ip"\n" >"$fmikro"
            fi
            break
        fi
    done <"$fip"
done <"$flinemikro"

# Only if $fmikro has been created, inform
if [[ -f "$fmikro" ]]
then
    printf "\n New file mikro.txt with lines and IPs of Ubiquiti devices.\n"
fi


#################################################################################################

### 3. MikroTik devices, open ports, and IPs

# Variables
ftemport="$path"/temp_ports.txt             # Open ports found on the NMAP scan
fportlines="$path"/port_lines.txt           # Lines where open ports where found
fports="$path"/ports.txt                    # Combination of the above files
flinemikro="$path"/line_ip_mikro.txt        # Lines where MikroTik IPs are on NMAP scan
fipmikro="$path"/ip_mikro.txt               # List of MikroTik IPs
fmikroports="$path"/mikro_ports.txt         # MikroTik IPs and ports, with lines from
NMAP scan
fport22open="$path"/port_22open.txt         # MikroTik IPs with port 22 open
fport23open="$path"/port_23open.txt         # MikroTik IPs with port 23 open
fport80open="$path"/port_80open.txt         # MikroTik IPs with port 80 open

aux_ip_line=0                               # Line from last IP read
aux_ip=""                                   # Last IP read
num_lines_flinemikro=0                      # Number of MikroTik devices in NMAP scan
line_num_flinemikro=0                       # Line number last entry $flinemikro
num_lines_ftempline=0                       # All IP lines from NMAP scan
line_num_ftempline=0                        # Line number last entry $ftempline


# Generation of file with lines of open ports found in nmap_output.txt
grep -n " portid=" "$nmapout" | grep "open" | cut -d '"' -s -f4 > "$ftemport"
grep -n " portid=" "$nmapout" | grep "open" | cut -d ':' -s -f1 > "$fportlines"
paste -d: "$fportlines" "$ftemport" >"$fports"

printf "\n New file temp_port.txt with the scanned open ports.\n"
printf "\n New file temp_line.txt with open port line numbers from file
nmap_output.txt.\n"
printf "\n New file ports.txt with all lines and open ports.\n"

# Generation of file mikro_ports.txt containing MikroTik lines, IPs, and ports
if [[ -f "$fmikro" ]]
then
       cut "$fmikro" -d ':' -s -f1 >"$flinemikro"
       cut "$fmikro" -d ':' -s -f2 >"$fipmikro"

       # If $fmikroports exists, delete its content
       if [[ -f "$fmikroports" ]]
       then
           rm "$fmikroports"
       fi
```

```
        # Finding MikroTik ips and their open ports

        # Number of lines and last entry of flinemikro and ftemplines
        num_lines_flinemikro=$( echo $(wc -l "$flinemikro") | cut -d ' ' -s -f1 )
        line_num_flinemikro=$(tail -n +"$num_lines_flinemikro" "$flinemikro")

       # Used to conclude whether last AirOS IP entry is last IP from $nmapout
       # Useful when writing last IP ports into a file
        num_lines_ftempline=$( echo $(wc -l "$ftempline") | cut -d ' ' -s -f1 )
        line_num_ftempline=$(tail -n +"$num_lines_ftempline" "$ftempline")


        # Read line number and MikroTik IP
        while read mikro_line <&3 && read mikro_ip <&4
        do
            # Check whether file exists to append or create
            if [[ -f "$fmikroports" ]]
            then
                printf ""$mikro_line":"$mikro_ip"\n" >>"$fmikroports"
            else
                printf ""$mikro_line":"$mikro_ip"\n" >"$fmikroports"
            fi

           # Case MikroTik IP as last and only entry
            if [[ "$line_num_flinemikro" -eq "$line_num_ftempline" &&
"$num_lines_flinemikro" -eq 1 ]]
            then
                while read line_port <&5 && read port <&6
                do
                    if [[ "$line_port" -gt "$line_num_flinemikro" ]]
                    then
                        printf ""$line_port":"$port"\n" >>"$fmikroports"
                    fi
                done 5<"$fportlines" 6<"$ftemport"

             # Case MikroTik IP as last entry from many
            elif [[ "$line_num_flinemikro" -eq "$line_num_ftempline" &&
"$num_lines_flinemikro" -gt 1 ]]
            then
                while read ip_line
                do
                    if [[ "$ip_line" -gt "$mikro_line" ]]
                    then
                        while read line_port <&5 && read port <&6
                        do
                            if [[ "$line_port" -gt "$mikro_line" && "$line_port" -lt
"$ip_line" ]]
                            then
                                printf ""$line_port":"$port"\n" >>"$fmikroports"

                            elif [[ "$line_port" -lt "$mikro_line" ]]
                            then
                                continue
                            else
                                break
                            fi
                        done 5<"$fportlines" 6<"$ftemport"
                       break
                    fi
                done <"$ftempline"

        # Last MikroTik IP ports
               while read line_port <&5 && read port <&6
               do
               if [[ "$line_port" -gt "$line_num_flinemikro" ]]
```

```
                then
                        printf ""$line_port":"$port"\n" >>"$fmikroports"
                fi
                done 5<"$fportlines" 6<"$ftemport"

            # Case not MikroTik IP as last entry, independently of number IPs
            elif [[ "$line_num_flinemikro" -ne "$line_num_ftempline" ]]
            then
                while read ip_line
                do
                    if [[ "$ip_line" -gt "$mikro_line" ]]
                    then
                        while read line_port <&5 && read port <&6
                        do
                            if [[ "$line_port" -gt "$mikro_line" && "$line_port" -lt
"$ip_line" ]]
                            then
                                printf ""$line_port":"$port"\n" >>"$fmikroports"
                            elif [[ "$line_port" -lt "$mikro_line" ]]
                            then
                                continue
                            else
                                break
                            fi
                        done 5<"$fportlines" 6<"$ftemport"
                        break
                    fi
                done <"$ftempline"
            fi
        done 3<"$flinemikro" 4<"$fipmikro"

        #! NOTE ON $fmikroports: OPEN PORTS EXCLUSIVELY; FILTERED ONES NOT CONSIDERED !#
        printf "\n New file mikro_ports.txt with Ubiquiti lines, ips, and ports.\n"
else
        printf "\n Skipping open port analysis because there are no Ubiquiti devices.\n"
fi

# Search for Ubiquiti IPs with port 22/SSH open
port_selection "$path"/mikro_ports.txt "$path"/port_22open.txt 22

# Search for Ubiquiti IPs with port 23/Telnet open
port_selection "$path"/mikro_ports.txt "$path"/port_23open.txt 23

# Search for Ubiquiti IPs with port 80/HTTP open
port_selection "$path"/mikro_ports.txt "$path"/port_80open.txt 80

#############################################################################################

### 4.  Login attempt over SSH/22

# Variables
fssh22ip="$path"/ssh22_ips.txt              # Ubiquiti IPs with default credentials on
port 22


# SSH scan on Ubiquiti devices with port 22 open
if [[ ( ssh -eq 1 ) && ( -f "$fport22open") ]]
then
        printf "\n Starting SSH login attempt using found IPs with port 22 open\n"

        # If $mikro exists, delete its content
        if [[ -f "$fssh22ip" ]]
        then
            rm "$fssh22ip"
        fi
```

```
        while read ip_add
        do
            # Perform the first SSH login attempt on the addresses provided by
            # port_22open.txt
            sshpass -p "" ssh -o StrictHostKeyChecking=no guest@"$ip_add"

            # Checking everything went alright
            if [[ $? = 0 ]]
            then
                if [[ -f "$fssh22ip" ]]
                then
                    printf ""$ip_add"\n" >>"$fssh22ip"
                else
                    printf ""$ip_add"\n" >"$fssh22ip"
                fi
            fi

            # Second SSH connection attempt
            sshpass -p "ubnt" ssh -o StrictHostKeyChecking=no ubnt@"$ip_add"

            # Checking everything went alright
            if [[ $? = 0 ]]
            then
                if [[ -f "$fssh22ip" ]]
                then
                    while read ip
                    do
                        if [[ $ip=$ip_add ]]
                        then
                            printf "\n Address %s already saved in: %s.\n" $ip_add
$fssh22ip
                        else
                            printf ""$ip_add"\n" >>"$fssh22ip"
                        fi
                    done <$fssh22ip
                else
                    printf ""$ip_add"\n" >"$fssh22ip"
                fi
            fi

            # Third SSh connection attempt
            sshpass -p "ubnt" ssh -o StrictHostKeyChecking=no root@"$ip_add"

            # Checking everything went alright
            if [[ $? = 0 ]]
            then
                if [[ -f "$fssh22ip" ]]
                then
                    while read ip
                    do
                        if [[ $ip=$ip_add ]]
                        then
                            printf "\n Address %s already saved in: %s.\n" $ip_add
$fssh22ip
                        else
                            printf ""$ip_add"\n" >>"$fssh22ip"
                        fi
                    done <$fssh22ip
                else
                    printf ""$ip_add"\n" >"$fssh22ip"
                fi
            fi

        done <"$fport22open"
```

```
        if [[ -f "$fssh22ip" ]]
        then
            printf "\n New file ssh22_ips.txt with listing all IPs with default
credentials.\n"
        else
            printf "\n The SSH scan returned 0 Ubiquiti devices with default
credentials.\n\n"
        fi
else
    printf "\n Skipping SSH login either because not actived or zero Ubiquiti IPs\n\n"
fi



##############################################################################################


### 5.  Check of current firmware using SNMP.

# Variables

fdb="$HOME"/bin/airos/database_firmware_airos.txt # Firmware updates from last years
fsnmpout="$path"/snmp_out.txt                     # SNMP scan results
firm="$path"/firmware.txt                         # Data analysis over SNMP results

iso=.1.2.840.10036.3.1.2.1.4        # OID Ubiquiti software version
firmware=""                         # Firmware read from snmp results file
ip=""                               # IP read from snmp results file
db_firm=""                          # Firmware read from database
db_year=""                          # Year firmware released read from database
firm1=0                             # Content first part firmware version
firm2=0                             # Content second part firmware version

year_2017=()                        # AirOS last updated in 2017
year_2016=()                        # AirOS last updated in 2016
year_2015=()                        # AirOS last updated in 2015
more_3years=()                      # AirOS not updated more 3 years
unreachable=()                      # Devices with unknown firware state
unknown_version=()                 # AirOS with unknown firmware version
writen_ip=()                        # Flag to determine whether IP has been writen in
file

# Scan over MikroTik IPs searching for their current firmware version
if [[ "$snmp" -eq 1 && -f "$fipmikro" ]]
then
        # If $snmpout exists, delete its content
        if [[ -f "$fsnmpout" ]]
        then
            rm "$fsnmpout"
        fi

        while read ip_add
        do
            # Perform the SNMP test over the Ubiquiti IPs
            firmware=$(snmpwalk -c public -v1 "$ip_add" "$iso" | cut -d '"' -s -f2 ) 2>
/dev/null

            if [[ -f "$fsnmpout" ]]
            then
                printf ""$ip_add":"$firmware"\n" >>"$fsnmpout"
            else
                printf ""$ip_add":"$firmware"\n" >"$fsnmpout"
            fi
        done <"$fipmikro"
```

```
        #! NOTE ON $fsnmpout: IPs WITHOUT ASSIGNED CURRENT FIRMWARE WERE UNREACHABLE !#
        printf "\n New file snmp_out.txt which contains a list of Ubiquiti IPs "
        printf "and their current firmware.\n"

else
        printf "\n Skipping SNMP test this time. Either because the option has "
        printf "not been activated or because there are not Ubiquiti devices.\n"
fi


# Parse of the information provided by $fsnmpout and $fdb

if [[ -f "$fsnmpout" ]]
then
        while IFS=":" read ip firmware
        do
            writen_ip=0
            device=$( echo $firmware | cut -d '.' -s -f1 )

            if [[ -z $device ]]
            then
                unreachable+=("$ip")

            elif [[ $device="XC" || $device="XW" || $device="WA" || $device="XM" ]]
            then
                version_part1=$( echo $firmware | cut -d '.' -s -f3 )
                version_part2=$( echo $firmware | cut -d '.' -s -f4 )
                version=$version_part1.$version_part2
                version_part3=$( echo $firmware | cut -d '.' -s -f5 )

                # Version can be vx.x or vx.x.x; with last x<30 as upper bound
                if [[ $version_part3 -lt 30 ]]
                then
                    version=$version.$version_part3
                fi

                # Checked against $fdb to assign a last-update year
                while IFS=":" read db_firm db_year
                do
                    if [[ "$version" = "$db_firm" ]]
                    then
                        case $db_year in
                            2015) year_2015+=("$ip")
                                  writen_ip=1
                                 break
                            ;;
                            2016) year_2016+=("$ip")
                                  writen_ip=1
                                  break
                            ;;
                            2017) year_2017+=("$ip")
                                  writen_ip=1
                                  break
                            ;;
                            2014) more_3years+=("$ip")
                                  writen_ip=1
                                  break
                            ;;
                            2011) more_3years+=("$ip")
                                  writen_ip=1
                                  break
                            ;;
                        esac
                  fi
```

```
            done <"$fdb"

            if [[ writen_ip -eq 0 ]]
            then
                unknown_version+=("$ip")
                writen_ip=1
            fi
        fi
    done <"$fsnmpout"
fi

# Generation of a file containing all MikroTik devices with year of last update
if [[ -f "$fsnmpout" ]]
then
       # If $firm exists, delete its content
       if [[ -f "$firm" ]]
       then
           rm "$firm"
       fi

       # Unreachable IPs
       printf "Unreachable:\n" >"$firm"

       if [[ ! -z "$unreachable" ]]
       then
          printf "%s\n" ${unreachable[*]} >>"$firm"
       fi

       # Unknown versions
       printf "unknown_version:\n" >>"$firm"

       if [[ ! -z "$unknown_version" ]]
       then
           printf "%s\n" ${unknown_version[*]} >>"$firm"
       fi

       # IPs updated 2017
       printf "2017:\n" >>"$firm"

       if [[ ! -z "$year_2017" ]]
       then
           printf "%s\n" ${year_2017[*]} >>"$firm"
       fi

       # IPs updated 2016
       printf "2016:\n" >>"$firm"

       if [[ ! -z "$year_2016" ]]
       then
           printf "%s\n" ${year_2016[*]} >>"$firm"
       fi

       # IPs updated 2015
       printf "2015:\n" >>"$firm"

       if [[ ! -z "$year_2015" ]]
       then
           printf "%s\n" ${year_2015[*]} >>"$firm"
       fi

       # IPs updated more than 3 years ago
       printf "<2015:\n" >>"$firm"

       if [[ ! -z "$more_3years" ]]
       then
```

```
            printf "%s\n" ${more_3years[*]} >>"$firm"
        fi

        printf "\n New file firmware.txt with a list of MikroTik IPs "
        printf "and year of last firmware update.\n\n"
fi

#################################################################################################

printf "\n END OF SCRIPT: AirOS tests.\n\n"
```

8.2.4. AirOS data

```
#! /bin/bash

#: Title        : AirOS data
#: Date         : January 2018
#: Author       : "Patricia Ramos García" <patram@uoc.edu>
#: Version      : 1.0
#: Description : Parse of data from files resulting from "airOS test" script
#: Options      : None
#: Arguments    : None

#################################################################################################
#################################################################################################


# This script parses data from files resulting from "airOS test" script

# Variables
pathbin=$HOME/bin                          # Path to relevant files in bin
pathdir=$HOME/bin/airos                    # Path to AirOS scanned regions directories
fdirectory=$pathbin/directories_airos.txt  # List of regions directories

files=$pathbin/routerOS_files.txt    # List of files to parse
foutdata=$pathbin/airOS_data.txt           # Output data

file=""                                # File to read within a specific directory
ssh=0                                  # Number IPs with default SSH credentials
port23=0                               # Number IPs with port 23 open
port22=0                               # Number IPs with port 22 open
port80=0                               # Number IPs with port 80 open
fip=0                                  # Number IPs scanned
openip=0                               # Number non-filtered IPs scanned
mikro=0                                # Number IPs Ubiquiti
y2017=0                                # Number IPs updated in 2017
y2016=0                                # Number IPs updated 2016
y2015=0                                # Number IPs updated 2015
other=0                                # Number IPs updated before 2015
unknown_ver=0                          # Number AirOS which firmware is not in DB

# Flags
seventeen=0                            # Found list IPs last updasted in 2017
sixteen=0                              # Found list IPs last updated in 2016
fifteen=0                              # Found list IPs last updated in 2015
```

```
less=0                                  # Found list IPs updated before 2015
unknown=0                               # Found list IPs whose firmware is not catalogued

# Case foutdata exists, delete
if [[ -f $foutdata ]]
then
    rm $foutdata
fi

while read diread
do
    pathdir=$HOME/bin/airos/$diread

    # Case the directory does not exist
    if [[ ! -d $pathdir ]]
    then
        printf "\n The directory %s does not exist.\n" $pathdir
        exit 1
    fi

    # All variables set to zero as they are shared with other directories
    ssh=0
    port23=0
    port22=0
    port80=0
    fip=0
    openip=0
    mikro=0
    y2017=0
    y2016=0
    y2015=0
    other=0
    unknown_ver=0

    # Read of each data file
    while read fileread
    do
        file=$pathdir/$fileread

        if [[ $fileread = "fip.txt" ]]
        then
            fip=$( wc -l $file | cut -d ' ' -s -f1)
        elif [[ ! -f $file && $fileread = "mikro.txt" ]]
        then
            printf "\n There are no Ubiquiti devices in: %s. \n" $diread
            break
        elif [[ $fileread = "mikro.txt" ]]
        then
            mikro=$( wc -l $file | cut -d ' ' -s -f1)
        elif [[ ! -f $file && $fileread = "ssh22_ips.txt" ]]
        then
            printf "\n No default SSH credentials on routers in: %s.\n" $diread
        elif [[ $fileread = "ssh22_ips.txt" ]]
        then
            ssh=$( wc -l $file | cut -d ' ' -s -f1)
        elif [[ ! -f $file && $fileread = "port_22open.txt" ]]
        then
            printf "\n Port 22 not open on routers in: %s.\n" $diread
        elif [[ $fileread = "port_22open.txt" ]]
        then
             port22=$( wc -l $file | cut -d ' ' -s -f1)
        elif [[ ! -f $file && $fileread = "port_80open.txt" ]]
        then
             printf "\n Port 80 not open on routers in: %s.\n" $diread
        elif [[ $fileread = "port_80open.txt" ]]
```

```
        then
            port80=$( wc -l $file | cut -d ' ' -s -f1)
        elif [[ ! -f $file && $fileread = "port_23open.txt" ]]
        then
            printf "\n Port 23 not open on routers in: %s.\n" $diread
        elif [[ $fileread = "port_23open.txt" ]]
        then
            port23=$( wc -l $file | cut -d ' ' -s -f1)
        elif [[ ! -f $file && $fileread = "firmware.txt" ]]
        then
            printf "\n The file firmware.txt does not exist: %s.\n" $diread
        elif [[ $fileread = "firmware.txt" ]]
        then
            less=0
            while read line
            do
                if [[ $line = "unknown_version:" ]]
                then
                    unknown=1
                elif [[ $unknown -eq 1 && $( echo $line | cut -d '.' -s -f1 ) -eq 10 ]]
                then
                    let unknown_ver=$unknown_ver+1
                elif [[ $line = "2017:" ]]
                then
                    seventeen=1
                    unknown=0
                elif [[ $seventeen -eq 1 && $( echo $line | cut -d '.' -s -f1 ) -eq 10 ]]
                then
                    let y2017=$y2017+1
                elif [[ $line = "2016:" ]]
                then
                    sixteen=1
                    seventeen=0
                    unknown=0
                elif [[ $sixteen -eq 1 && $( echo $line | cut -d '.' -s -f1 ) -eq 10 ]]
                then
                    let y2016=$y2016+1
                elif [[ $line = "2015:" ]]
                then
                    fifteen=1
                    sixteen=0
                    seventeen=0
                    unknown=0
                elif [[ $fifteen -eq 1 && $( echo $line | cut -d '.' -s -f1 ) -eq 10 ]]
                then
                    let y2015=$y2015+1
                elif [[ $line = "<2015:" ]]
                then
                    less=1
                    fifteen=0
                    sixteen=0
                    seventeen=0
                    unknown=0
                elif [[ $less -eq 1 && $( echo $line | cut -d '.' -s -f1 ) -eq 10 ]]
                then
                    let other=$other+1
                fi
            done <$file
      fi
 done <$files

 # Writing line with data from directory in airOS_data.txt
 # Variable order: fip mikro port23 port80 port22 ssh y2017 y2016 y2015 other &
unknown_ver
    if [[ -f $foutdata ]]
```

```
    then
        printf "%d %d %d %d %d %d %d %d %d %d %d\n" $fip $mikro $port23 $port80 $port22
$ssh $y2017 $y2016 $y2015 $other $unknown_ver >>$foutdata
    else
        printf "%d %d %d %d %d %d %d %d %d %d %d\n" $fip $mikro $port23 $port80 $port22
$ssh $y2017 $y2016 $y2015 $other $unknown_ver >$foutdata
    fi
done <$fdirectory

#################################################################################################

printf "\n End of AirOS data script.\n\n"
```