Master Degree Project

SafeHome

A cross-platform mobile application
that increases homes network security

Giovanni Musacchio

Supervisor: Marcus Nohlberg
Examiner: Per Backlund

Master Degree Project (120 ECTS) in Informatics
with a specialization in Data Science/
Privacy, Information and Cyber Security
30 ECTS
Autumn term 2022

Abstract

The Internet, considered the most fast-growing technology ever, is starting to arrive in everyone's homes regularly. No longer just smartphones, computers, or tablets, now even refrigerators, washing machines, and other household appliances support Internet connection. The more internet connections increase, the more homes are exposed to attacks every day. Companies work hard to protect devices from attacks by periodically releasing security updates that, in most cases, are never executed. The longer devices remain out of date, the more vulnerable they remain, therefore, they continuously risk network attacks.

This study presents SafeHome, a cross-platform mobile application that aims to shorten the gap between the release of an update and the actual device's update. With SafeHome, also people who are not familiar with technology will be able to increase the security of their homes with a few simple steps: pressing a button to scan the home network and just waiting until an update is released for one of the scanned devices.

The final result is still a prototype that follows the rules of a research study which highlights the possibility to reach the goal when technology will make another step forward.

Table of Contents

List of Figures

Listings

Chapter 1

Introduction

The Internet is clearly one of the most complex infrastructure ever created by humans. Due to its fast spreading all over the world, it has been recognized as one of our most fast-growing technologies. In 2000, the Internet had its first 413 million users, then, 5 years later, it crossed the one-billion barrier, reaching, in 2016, over 3.4 billion active users. Thanks to the popularity of social networks and the opportunity for people to connect to the Internet with a cheap smartphone wherever they are, this number increases and it will get closer to the number of people in the world year by year. For this reason, the Internet, nowadays, is populated by a wide belt of users with different capabilities, types of usages, and awareness. Following, a picture of the distribution of users in 2000 and the in 2020 (Roser, Ritchie, and Ortiz-Ospina 2015)

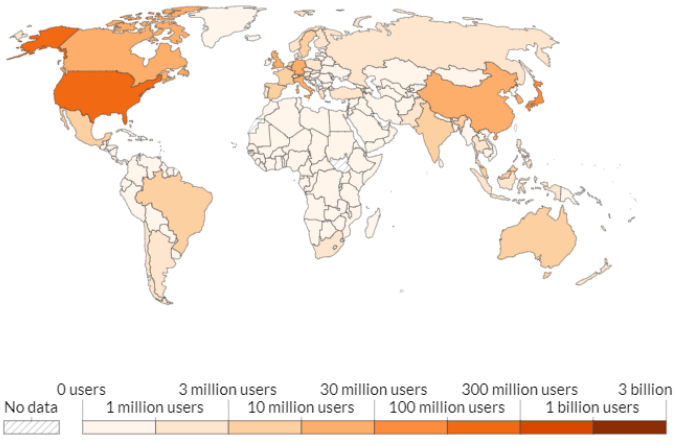


Figure 1.1: Users distribution in 2000

In 1.1 we can see the spread of the Internet in 2000. People affected by this new technology were located in the most developed areas of the world, such as America, central Europe, and China. The spread in those years was slow due to technological limitations that affected the poorest areas like Africa. Over the years, things changed and the need to connect the whole world to the Internet increased. There were a lot of improvements for Africa, especially in terms of intercontinental connectivity and terrestrial fiber networks. Submarine cable investment has amounted to around 3.8 billion dollars and terrestrial networks have seen over 8 billion dollars of investment (Ayeni 2019). Major changes can be seen in 1.2, where, 20 years later, numbers in poorer areas started to be significant and numbers in more developed areas increased even more.

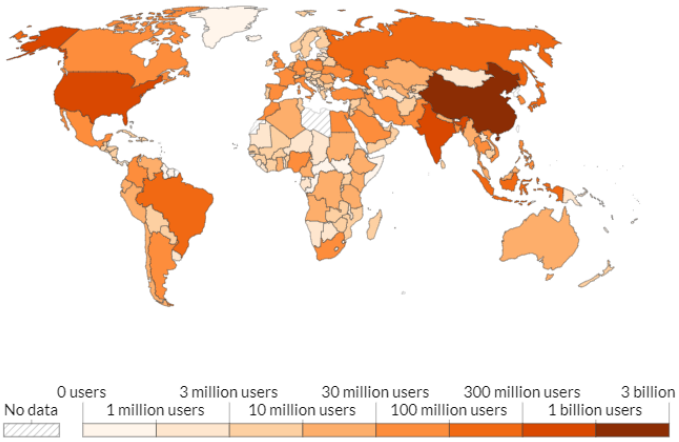


Figure 1.2: Users distribution in 2020

In the next chapter, users will be classified based on their usage of Internet, and through this classification, a specific target audience will be taken in consideration. It is not possible to assume that every user is aware of what he is doing over a network, of how he is using the features and of what are the main problems that can occur when being connected. The users are not aware of Information Technology security issues, they take it very lightly (Tariq, Brynielsson, and Artman 2014), and this results in a lack of awareness and dangerous situations for non-technical users. There are several cyber threats, most of them are dangerous and should be taken seriously (Shruti 2022), such

as zero-day vulnerabilities. A vulnerability is a piece of code executed in a malicious way from an attacker, and it takes the name of "zero-day" when it is discovered and published, in order to let companies work on it to fix the problem and release a patch through a software update. It is clear that the more time elapses between the release of an update and the actual update, the more are the possibilities to be the victim of a cyber attack. The awareness concerning the importance of software updates is poorly documented, all the information of an update are hidden behind the "details" button, which is not enough to efficiently alert all the types of users.

1.1 Problem description

The introduction has given an overview of the context, now we need to define the real problem by illustrating the main research question that led to the purpose of this work. The number of connected users to the internet is extremely huge, and, while being connected to the internet, a very low percentage of them understand at least few risks that can occur. Unfortunately or fortunately, every type of software currently available on the market has an enormous number of lines of code. The more lines of code are in a software, the more are the possibilities to step into weak code. Usually, weak code is fine, it is easy to think and write even though it generates some bugs, but, developers are human and cannot guarantee a software free from them. Nonetheless, we should make a distinction between small bugs and severe bugs, which we will deepen in the next chapter. The problem shown in this thesis concerns how weak code can be dangerous for users and how they don't protect themselves using the provided solutions through software updates by not doing them.

1.1.1 Research questions

A main research question must be found and stated to have the focus on what is the problem that we want to solve. Before stepping into a main question, the problem faces cyber threats caused by the weakness of the written code of a software, so we would like to know what are the main causes of cyber

threats for users. Furthermore, we want to know how is possible to alert users and help them to improve their security against cyber threats. The solution provided in this thesis concerns the urgency to update devices to make changes to the code effective and we want to achieve this goal through a smartphone application. This leads to the formulation of the main research question, which is the following:

Research Question

How can we get average users to be more aware of updating their devices?

The research question provided can be supported by other sub-questions in order to be more specific in the given context. The main question gives a challenge, but **RQ1**: *How can we highlight possible risks of not updating devices?* and **RQ2**: *What is the most efficient way to inform users about possible risks?* In the next chapter, a detailed description of what an "average user" is will be given.

1.2 Delimitations

In this thesis work, we will talk about the concept of "Smart Home". It is necessary to note from the outset that we are going to refer to "Smart Home" as *some* connected devices to the home network. Some example could be the obvious smartphones and personal computers, also tablets, smartwatches and smart TVs. Are excluded from this project all other IoT devices such as fridges, washing machines and other household appliances. We are referring to are only IP address-based devices, so we exclude any other device based on other protocols like zigbee or z-wave. Other than that, it is important to understand who is the target audience for this project. In the next chapter, the reader will find a section that explains which is the interested category of users and why it has been chosen instead of another one. During this explanation, a list of percentage will be presented, calculated in (Brandtzaeg, Heim, and Karahasanovic 2011) and referred to a dataset of 12,666 people in a range of 16-74 years old. Also, these samples are taken only in Norway, Sweden, Austria, Spain and United Kingdom.

1.3 Thesis content

The thesis is structured as follows:

- **Chapter 1 - Introduction**

  An introduction to the thesis. Here it can be found briefly the context and the topic of the thesis, as well as the basic description of the problem we found.

- **Chapter 2 - Background**

  The background will help the reader to fully comprehend all the notions which will be described in detail and will be found during the reading.

- **Chapter 3 - Method**

  In this chapter, there will be a discussion regarding the method used to solve the problem. Method is not referred to the solution in a practical sense but to the adopted strategy that helped to solve the problem.

- **Chapter 4 - Design**

  The design's chapter is where the reader will find information on how the solution is designed before its implementation. This is the first part of the method and it includes the preliminary brainstorming.

- **Chapter 5 - Implementation**

  This is where the actual implementation is explained. There will be also a study on how the code is structured and what are the reasons why something has been chosen rather than something else.

- **Chapter 6 - Results**

  Design and implementation led to a result that must be illustrated and explained. This is the chapter that we dedicate to the results and, if something was not possible, we include it in the results chapter under the limitations section.

- **Chapter 7 - Discussion**

  After illustrating the results, there is the need to discuss them. Here,

other than discussion, the reader will find some considerations from an ethical perspective.

- **Chapter 8 - Conclusions**

  In the final chapter where we summarize conclusions and illustrate some possible future works in terms of research and implementation.

Chapter 2

Background

This chapter presents a background based on the method that will be explained in the next half part of the thesis. Firstly, we will define our target audience. As has been said, currently in the world there is almost half of the population connected to the internet, so we need to find the real target we are referring to. Then, we will take a look at vulnerabilities, what they are, and what damages they can cause if countermeasures are not taken quickly. Then, we will discover how these vulnerabilities are fixed through the concept of "patch". Last, we will deepen briefly the concept of API, which will be useful to understand the implementation part of the project.

2.1 Target audience

In the first chapter, during the introduction, we talked a lot about the users. Internet was popularized in the second half of '90, and, even though it is the most fast-growing technology, it took a lot of years to succeed on the market. During this time, people started taking a position related to this new technology. Some of them were interested, some of them were completely indifferent, and another part was even against it. Due to all those separations, a lot of people did not keep up with the progress and that is the cause of why we are now forced to make a distinction between users. According to the International Journal of Human-Computer Studies, there are 5 categories of users on the Internet (Brandtzaeg, Heim, and Karahasanovic 2011), so the

following list will help us better understand which type of user will be the final interested user in this project:

- **Non-Users** This category is made for literally non-users, people who don't use Internet and do not take advantage of all its features.
- **Sporadic Users** This category is made for people who use Internet in a basic way. They usually use it for emails or standard tasks.
- **Entertainment Users** In this category, we add people who use Internet as a source of entertainment, such as watching films, playing video games, listening to music, and so on.
- **Instrumental Users** Instrumental users are people who use Internet in a functional way. They make use of Internet to manage a lot of things in their life such as bank accounts, social networks, and also they use it to gather general information when needed. We can find also people whose job requires basic usage of this technology.
- **Advanced Users** In this category, we add the remaining part, who use Internet every day as a job or personal source of earnings.

This is quite a large definition of user. It is better to restrict this list and try to find out just two main categories. We are not going to consider the first category, since they do not make use of Internet at all. What we want to do is to merge the next three categories, leaving alone the category of advanced users. Therefore, there are two categories left that we can name "Advanced Users", as suggested by the Journal, and "Average Users". In the next paragraphs, the definition of both categories will be explained to better understand the type of users who are inside them.

2.1.1 Advanced users

Advanced users are not so common. It is only a small subset of users compared to the averages. Typically, advanced users have been living and using technology, internet in particular, for a long time, since its growth or even its birth.

By this definition, it might seem that advanced are only people born or lived in the period of Internet's birth. The definition of advanced users is slightly more complex. Nowadays, a young man can be also defined as an advanced user for several reasons, for example, thanks to his studies, or it might be for his passion for technology, but there could be way more reasons. Furthermore, some people around a couple of years less than the age of majority could be considered advanced, since they are living in the era in which the Internet has its best presence in the world and have the ability to understand specific concepts in that context. Trying to summarize, we will consider an advanced user as a user able to comprehend technicalities, understand where, when, and how he is doing something, and, above all, knows about at least the most important threats that can occur in a technological environment.

2.1.2 Average users

The biggest part of users is composed of people whom we define as average users. Thanks to all the means people have nowadays, a huge number of them are connected to the Internet. As said in the introduction, there are probably more than 4 billion active users now all over the world, and, since this one is the biggest part, more than 2 billion users are cataloged as "average". Deepening into this word, we tend to exclude from this category, all the characteristics of advanced users described before. Since this is the largest part of users, a proper way to catalog them is based on their use of technology. There are people linked to old habits, no matter their age, so they could use devices only if necessary like keeping in touch with their family with a low-cost smartphone. Other people make better use of devices caused by strong and frequent use of social networks, a number that is rapidly increasing. A single person can do something even more than just navigate on a social network, like doing research or using general online services, and still be defined as an average user because he does not know technicalities, it is not possible to expect someone to know what a vulnerability is if he has not studied something before. In the end, a summary could be that average users are not aware of the risks they are subjected to every day while connected to the network, which makes them

vulnerable. Excluded from this category are all those people who, despite being average users, have matured over time, and maybe for business reasons, a certain practicality with technology, taking courses, documenting themselves, and learning from friends and colleagues more and more from technology.

2.2 Security vulnerabilities

Nowadays, the entire world is full of devices connected to the internet. This gives large space to attackers that want to steal information or just make a device unavailable. When a person can attack a device, it means that there is a *vulnerability*. The process of developing software is complex, and even with a lot of testing mistakes could happen. More precisely, a vulnerability is a piece of code used in a way it's not intended. In other words, a vulnerability is a weakness in the application which can be an implementation bug or a design flaw that allows an attacker to cause harm to the user of the application and get extra privileges. An attacker could take advantage of this piece of code to do what he wants even though that code has been written for different purposes. The event of an attacker exploiting a vulnerability takes the name of *threat*. Vulnerabilities just discovered are called zero-day, which alludes to the fact that developers have zero days to address a vulnerability the moment it is discovered. Online can be found the official database which contains the list of all the discovered vulnerabilities along with a score, the Common Vulnerability Scoring System (CVSS), which works as a grade and indicates the potential risk of that particular vulnerability. Still today, vulnerabilities are one of the most dangerous causes of attacks on the internet and the biggest flaws in system security (Rapid7 2021). A process can be used to describe what happens as soon as a vulnerability is identified, as shown in 2.1.

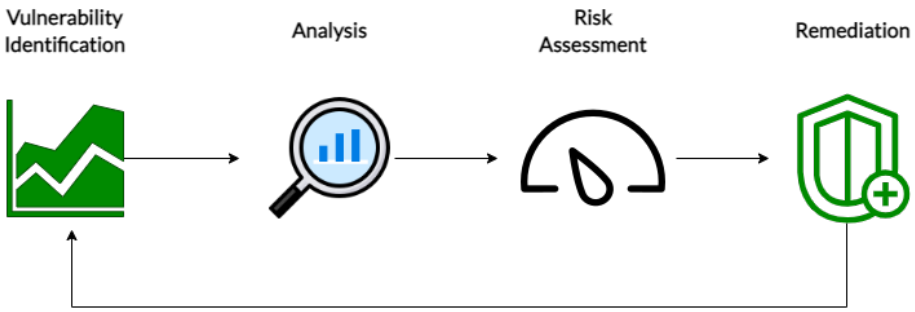


Figure 2.1: Vulnerability assessment process

This process is called *vulnerability assessment* and, along with *penetration testing*, it is the fastest way to identify vulnerabilities and remedy to them.

2.2.1 Vulnerability assessment

Vulnerability Assessment and Penetration Testing is a step-by-step process. Vulnerability assessment is the process of scanning the system or software or a network to find out the weakness and loopholes in that. These loopholes can provide a backdoor for an attacker to harm the victim. A system may have access control vulnerabilities, Boundary condition vulnerabilities, Input validation vulnerabilities, Authentication Vulnerabilities, Configuration Weakness Vulnerabilities, Exception Handling Vulnerabilities, etc (Goel and Mehtre 2015). Penetration testing is the next step after vulnerability assessment. Penetration testing is to try to exploit the system in an authorized manner to find out the possible exploits in the system. In penetration testing, the tester has the authority to enter systems to find out possible exploits. Vulnerability assessment is a long and complex process. For this reason, security analysts have various tools to use to analyze applications and check for vulnerabilities within them. Among these we can find (Rosencrance 2021):

- **Network-based scans** This scan is used to discover vulnerabilities inside a network. The power of these tools is given by the fact that they can analyze also wireless networks, ensuring a wide coverage area of security.
- **Host-based scans** Tools similar to the network-based ones, but these don't scan the networks rather they look for vulnerabilities in servers,

machines, and workstations in general. In particular, they scan also services and open ports.

- **Application scans** The scan of an application deals with security issues that can occur on a website. These types of scanning are very important due to the huge number of cyber attacks that affect internet.
- **Database scans** The same as the previous, but this time the scan is made over a database. Another important job due to attacks made against databases to steal data and personal information.

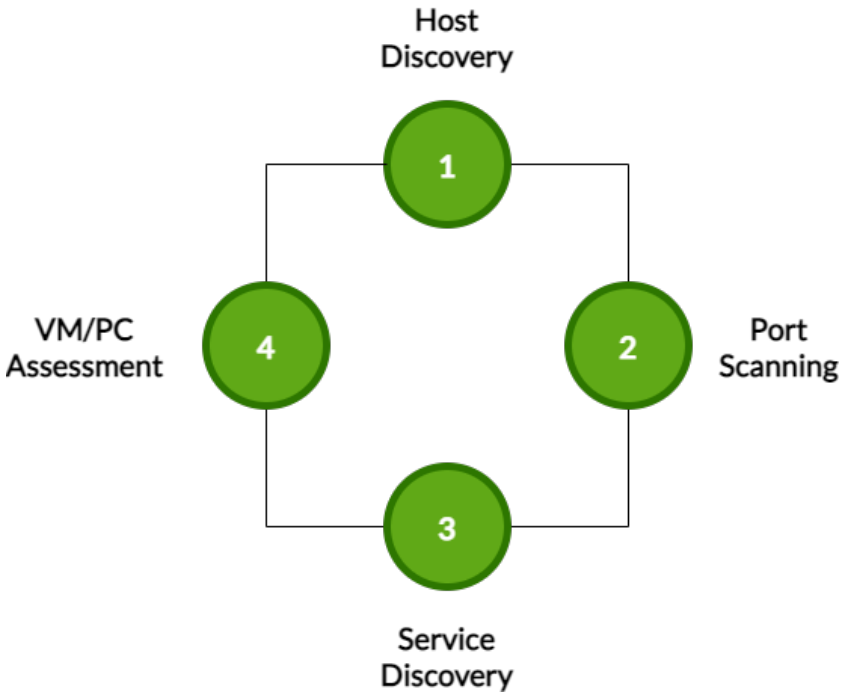


Figure 2.2: Scanning process during vulnerability assessment

2.2.2 Priority and scoring system

Common Vulnerability Scoring System (CVSS) is a free and open industry standard for assessing the severity of computer system security vulnerabilities. It is under the custodianship of the Forum of Incident Response and Security Teams (FIRST). It attempts to establish a measure of how much concern a vulnerability warrant, compared to other vulnerabilities, so efforts trying to solve them can be prioritized (Kaluarachchi, Tsokos, and Rajasooriya 2016).

CVSS Score	Qualitative Rating
0.0	None
0.1 - 3.9	Low
4.0 - 6.9	Medium
7.0 - 8.9	High
9.0 - 10.0	Critical

2.3 Security patches

A security patch is roughly defined as the solution to a vulnerability. As soon as a zero-day vulnerability is discovered, the developers of the piece of code under consideration must work to first understand why that snippet results in a vulnerability, and second update and release a new piece of code that mitigates the vulnerability. These releases are called patches, and they are usually released with other bug fixes and small changes that improve the system. For the changes to take effect, users must update their devices to install correctly the patches (Ojuwoni 2022).

2.4 Application Programming Interfaces

Application Programming Interfaces, always abbreviated in "API", are ways to let two or more computers communicate with each other. A computer can use a service exposed by another computer through APIs without knowing how

that service is implemented. APIs will be mentioned in the chapter related to the implementation since this project uses APIs to communicate with services exposed by the vendors of many devices.

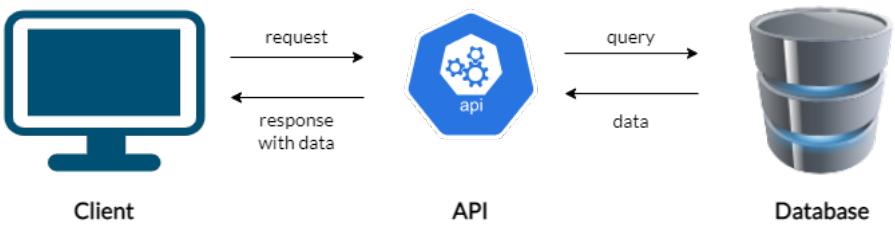


Figure 2.3: How an API call works

2.5 Related works

Several studies have been conducted in the security update field. In particular, the area that this work deals with is the relationship between users and updates, which is a difficult relationship and involves various reasons why people could decide to not update devices.

A first study is "A study of users' experiences and beliefs about software update messages" (Fagan, Khan, and Buck 2015). This study deals with the thoughts of people when seeing a message of a new update and uses two frameworks to analyze it, Communication-Human Interface Processing (C-HIP) and the Affect-Reason-Involvement (ARI). The C-HIP model takes into account the source, which is the entity that is trying to communicate something (in this case the company that alarms the user about the update), the communication channel (the message) and the receiver (the final user). According to this model, there are two important features that the message must have to be successful: it must be attention-grabbing, which means that it must quickly get the user's attention, and it must be understandable, so every user must understand the content of the text. With this definition, it is clear that the majority of users are not affected by these messages because they don't under-

stand the context and their purpose. The other used model is ARI. This model uses emotional attempts to persuade users, for example, companies would need emotional messages to convince users to update a device, such as the actual problem that the patch fixes, or even some information regarding the possibility to lose data or to have serious problems with the system.

Another study is "Update now or later?" (Rajivan, Aharonov-Majar, and Gonzalez 2020). This work aims to understand why people decide to delay the action of updating software by studying their past experiences. They investigate two main aspects: firstly they consider the fact that users have never faced an attack, in their minds it is unlikely to receive one at that moment so they decide to delay the update. In the second place, the study takes into account costs. For a user, it can be time-consuming to update a device while doing another primary work, so this results in a delay. It must be considered that a delay can lead to a "never" update, a user can decide to delay many times until a certain point in which it will be useless to update.

Chapter 3

Method

This chapter focuses on methodology. Two methodologies can be distinguished in this case: a research methodology and a development methodology. The first one explains how data are collected in relation to the research question. The second one is the methodology used for the actual implementation of the project.

3.1 Research methodology

Referring to the research question, there is a huge part of users that are not aware of all the problems that may occur by not updating devices. The way that has been used to gather data about this problem, other than related works, is the research methodology. The research method is chosen based on the research question, the data needed, and the resources available. There are several methodologies, the chosen one that best fits this work is qualitative research. The introduction states that people are not familiar with updating their devices, and this has even a major impact on people who are not familiar with technology, who are called here "average users". The methodology that allowed to end up to this statement is a type of qualitative research: **observation**. There are studies conducted to demonstrate that participant observation is a valid method to collect data in qualitative research, in fact, according to (Creswell 2014), qualitative observation is a method used by researchers to take notes on the behavior and activities of individuals. The validity of this method

can be strongly supported by interviews, surveys, questionnaires, and so on (Kawulich 2005). The reason why qualitative research was chosen rather than quantitative research is that there is no data in numerical form such as numbers, percentages, and so on. Instead, qualitative research focuses on people's experiences. As said before, it is possible to support observation with another method, in this case **interviews**. Interviews are a useful research strategy for gathering information from a group of people. (Creswell 2014) says that qualitative interviews are a method where the researcher conducts face-to-face interviews with participants to get feedback and opinions from them. From (Alshenqeeti 2014), research has shown four types of interviews: the first one is structured where the researcher finds a set of questions to ask the participants. Typically, these questions have just "yes" or "no" as answers, therefore it is a limited type of interview. The second type of interview is unstructured and provides more freedom to both the interviewer and interviewees. For this reason, answers, in this case, can be longer and more complicated. The third type is a mix of the first two. With this method, there could be the use of a checklist that would help cover all relevant areas of the interview. The last type is "focus group interviewing", interviews where participants are selected because they are a sampling of a specific population. The chosen type of interview will be described in the results chapter under the evaluation paragraph.

3.1.1 Evaluate the solution

To enhance the use of observation, interviews were used. The intention is to provide the interviewee with a series of questions aimed at both gathering data related to the research question and data related to the final result of the implementation. Hence, the following table contains the chosen questions with their identifiers. Questions from Q1 to Q3 belong to the first set, indeed they provide information regarding whether a user sees and feels the problem of updating devices or not. Questions Q4 and Q5 are asked after the presentation of the prototype. They aim to get information on the experience that users feel after using for a while the application. In this way, after an appropriate phase of data analysis, it is possible to evaluate the solution and derive conclusions.

ID Question	Question
Q1	How often do you update your devices?
Q2	Are you aware of what may occur if a device is not being updated for a long time?
Q3	Do you think an app could help?
Q4	Do you find it easy to navigate inside the application?
Q5	Would you update a device after receiving a notification?

As soon as the questions are outlined, the interviewees need to be selected. There are two factors to be considered: the number of interviewees and the type of interviewees based on their closeness to technology. The number is important because it has to be sufficient to validate the interview. On the other hand, the choice must consider the interviewees' background in relation to the research question. In this case, the research question takes into account average users, but, in the background chapter, the description of average users explains that it does not matter the actual background of a user such as his job, rather it is more important to consider his familiarity with technology. For this reason, to evaluate properly the solution, different users with different backgrounds and familiarity were selected. Regarding the number, it seems that four or five participants help to discover at least 80% of problems related to usability tests (Turner, Lewis, and Nielsen 2006), which is a sufficient percentage. In this case, five people were selected to be interviewed. In the next table, it is possible to see the set of characteristics that describes an interviewee:

Person	**Age**	**Sex**	**Background**	**FwT**

There are five parameters. The first one is *person*, which is a sort of identifier for an interviewee. In the evaluation, this field will be filled with random names for privacy reasons, and it will be used to refer to that person in the answers section. The next two parameters, age, and sex, are self-explanatory. The *background* parameter refers to the interviewee's job and it gives information about the possible technological background. The last parameter FwT is the

acronym for "familiarity with technology". This is a number between 1 and 10 asked to the interviewee before starting. The higher the FwT, the more the person is familiar with technology, therefore possibly closer to certain technical concepts like vulnerability, patch, and others. As an example, a person that gives himself a 4 is considered unfamiliar with technical aspects of technology. A lower Fwt could also indicate that the person does not even know what the interview is about, even though he or she owns a simple device, so in that case, there will be the necessity to explain and give the interviewee context.

An interview session with one person worked as follows: firstly a brief context was given to the interviewee, then questions started and it took around 25 minutes for an interview to be completed. All the answers given were collected on a paper, then they were transcribed in the results chapter where it can be found the table with users' data and the questions that have been found and considered the best for addressing the problem.

3.1.2 Data analysis

According to (Asamba 2023), the purpose of data analysis is to obtain useful and usable information that will identify differences by comparing variables. Gathering information through interviews allows the researcher to collect a significant amount of data in the research field. Through analysis, researchers can have a better understanding of a participant's experience and perspective and use this information to derive conclusions on the conducted research. There are several qualitative data analysis methods, for example, thematic analysis, content analysis, and grounded theory. This last one mentioned was not appropriate since it is based on the generation of theories and hypotheses from data gathered. Grounded theory generates new theory from data and opposes testing the existing theory (D. Mohajan and H. Mohajan 2023). Thematic analysis is based on the identification of patterns in data and their subdivision into themes or topics. In this case, content analysis was selected as the best method for this project. Content analysis is a method that involves the analy-

sis of the actual content. Each person gave different answers and each answer provided a different content. The best way in this case is to take each answer and analyze its content to derive conclusions.

3.2 Development methodology

A development process, or methodology, is the process of dividing the development of software into sub-processes which can be parallel or sequential. It is also known as a software development life cycle (SDLC) and it is a structure imposed on the development of a software product (Sarker et al. 2015). There are a few different models that may be used for these kinds of processes. Each of these models describes an approach to a different kind of activity that takes place during the process. The development models consist of a variety of processes or techniques that are being considered for use in the development of the project. This decision is being made with respect to the objectives of the project. Many different models of development life cycles have been created throughout the years to meet the requirements of a variety of various kinds of goals. The models detail the several steps that make up the process as well as the specific sequence in which those steps are carried out. The choice of model has a considerable impact on the outcomes of the development that is carried out. The methodologies that we want to highlight here are:

1. **Waterfall Model**
2. **Agile Development**
3. **DevOps Development**

These three approaches will be discussed in the following sections, with an emphasis not just on their strong aspects but also on their limitations, in order to provide a context for the final decision of which approach will be applied.

3.2.1 Waterfall Model

Waterfall is the most popular method of software development. It belongs to the category of methodologies that makes its sub-processes sequential instead of parallel. Once a portion of this process has been completed, it is impossible to make any changing. This model gives huge importance to the documentation of the software in question, in which developers will find a highly detailed description of each functionality.

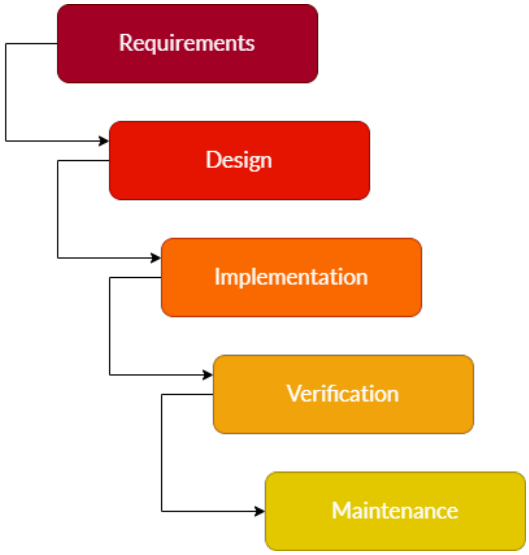


Figure 3.1: Illustration of the Waterfall Model

Teams implementing the Waterfall development methodology can provide an accurate release date prediction by precisely estimating the amount of time necessary to finish each step of the development process based on research conducted during the pre-development phase.
Due to its rigid structure, the Waterfall model seems to be the best solution for well-defined projects since the beginning with strong documentation. On the other hand, it is not the best solution when modifications may occur during the development (Synopsys 2017).

3.2.2 Agile Development

Agile software development refers to a development methodology that uses iterative development, frequent consultation with the customer, small and frequent releases, and rigorously tested code (Barlow et al. 2011). Agile development was created primarily to solve the limitations of plan-based techniques of software development, such as the popular waterfall method (Royce 1987). The main weakness of plan-based techniques is the lack of responsiveness to change along the development of software, and this is essentially the main reason why agile development was created. The efficiency of agile is proved by its impact on IT companies, in fact, in the United States, 71% of the companies have adopted this methodology. Plus, the success rate of projects using agile is 64% against the 49% of the Waterfall model (Flynn 2022).

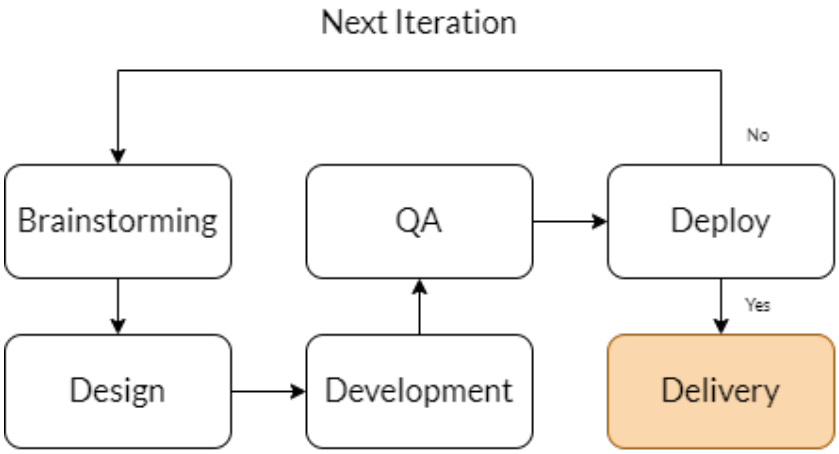


Figure 3.2: Agile Development Cycle

As shown in Figure 3.2, Agile Development is divided into sub-processes arranged as a cycle. Each step represents an increment in the development phase. Deepen into each of them, there is:

- **Brainstorm** The first phase is dedicated to brainstorming. All the requirements are gathered here to understand in the best way all the features that must be implemented.
- **Design** To avoid several code changes, especially when it comes to the

User Interface, a design phase is needed to prototype what is going to be developed. For this reason, here the interfaces are designed first with wireframes and then with mock-ups.

- **Development** The actual implementation of the features is being done in this phase.
- **QA** QA is the acronym for Quality Assurance. Essentially, this phase is testing the features to ensure that everything is working properly.
- **Deploy** Whenever the output of the quality assurance phase is positive, the features are submitted to the client for the final check. If this is also positive, the features are marked as completed.
- **Delivery or Next Iteration** The last phase can be split in two, depending on the output of the deployment. If the previous phase was the deployment of the last feature that had to be implemented, the delivery branch is executed, the project is delivered to the client and everything is marked as completed. If the previous was not the last, a new iteration is needed to step into the next feature to be implemented.

Thanks to incremental steps and individual features development, Agile Model allows a close relationship with the client, in order to better understand requirements and to deliver the best product at the end.

3.2.3 DevOps Development

Another recent development model is Development & Operations, most of the times abbreviated in *DevOps*. This term was initially used in 2008 when Patrick Debois, at the Agile 2008 Conference, mentioned the need for an agile infrastructure and interaction between the development and operations teams. This model, therefore, aims at producing fast delivery to customers by bringing the development and operation team to work together (Mishra and Otaiwi 2020). More precisely, DevOps is a set of methods in which developers and operations communicate and collaborate to deliver software and services rapidly,

reliably, and with higher quality. DevOps is sharing tasks and responsibilities within a team empowered with full accountability for its service and its underlying technology stack.

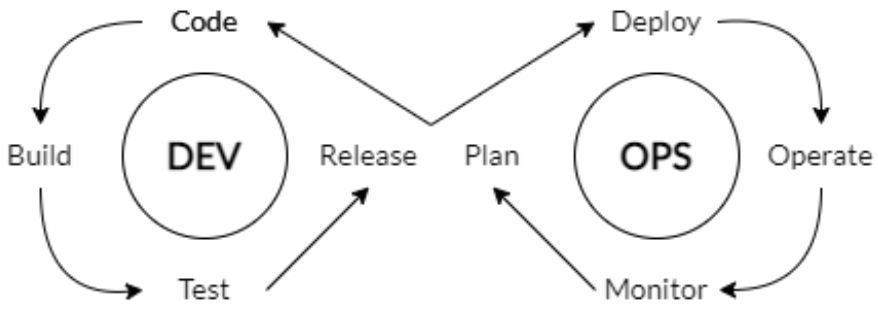


Figure 3.3: DevOps Development Cycle

3.3 Chosen methodology

The proper software development process is a crucial instrument for handling complicated jobs in an effective and coordinated manner. There are several development strategies that are suited to unique qualities and needs. It is crucial for the success of a project to follow an appropriate software development approach that increases team productivity. Since this thesis work is conducted by one student, the analysis of the pros and cons of the three methodologies illustrated are the following:

	Budget	**Deadlines**	**Requirements**	**Clients**
Waterfall	-	+	-	-
Agile	+	-	+	+
DevOps	-	+	+	+

Table 3.1: Pros and cons of the presented methodologies

As shown in table 3.1, the presented methodologies differ in four categories: budget, deadlines, requirements, and clients. The first yardstick refers to the budget and measures which of the models is more expensive than the others. Deadlines specify how strict a model should be in terms of deliveries. Requirements point out how properly the requirements have been understood, and this goes along with the last yardstick, clients, which relates to the quality of the communication, if any, with customers. So, inspecting the table, it can be clearly obvious that the Waterfall model is the weakest, pointing to its strength in deadlines, which is not useful for this project. Between Agile and DevOps models, Agile Development has been chosen for the following reasons:

1. DevOps is a good option when working closely with the QA team. Since the presented project includes only one developer, the choice of this method is useless.

2. Agile Development has the same number of pros as DevOps. The difference between the two models is the third pro which in the case of Agile is on the budget instead of deadlines. Having a pro on budget means that development will be as cheap as possible, hence, as fast as possible.

3. Agile is the best method when it comes to developing features one by one. A single iteration of an Agile cycle corresponds to the completion of a single feature which, at the end of the iteration, must be finished and fully working.

Therefore, Agile development has been officially chosen. From a developing side, all the considerations that have been made regarding the choice of a strategy have been confirmed by the efficient development done afterward. As said, the roadmap of the project, which will be shown later in the next chapter, is composed of a series of features that need to be implemented serially. In this sphere, the Agile strategy allowed to build each feature from the design to the testing without losing focus on it. Sometimes if developers try to implement more than one feature at the same time, they will lose focus on what they were doing and this will result in a bad implementation of all the features involved. Here, instead, the focus is on just one feature through all the Agile steps and

this is why this method was a perfect match for this kind of project and just one developer.

Chapter 4

Design

In the previous chapter, in particular at 3.1.2, Agile Development has been illustrated and then selected as the optimal development methodology for this project based on several factors that were thoroughly detailed. This chapter will now examine the brainstorming and design phase of Agile and describe in detail how each phase has been completed according to its definition.

4.1 Planning

Planning is a crucial step in the development of any project, including the creation of a mobile application. The project's scope and objectives are established during the planning phase, and a development roadmap is also developed.
Some specific tasks that may be involved in the planning phase of a mobile app development project include:

1. Defining the target audience: identify whom the app is intended for and what their needs and goals are. This has been largely discussed in the second chapter.

2. Identifying the features and functionality: what features and functionalities the app will include, and how these will address the needs and goals of the target audience.

3. Creating a roadmap: a detailed plan that outlines the steps that are

needed to develop the app, including the timeline and any dependencies or milestones.

The planning phase is an important foundation for the rest of the development process, as it helps ensure that your app is well-defined, focused, and aligned with your goals and the needs of your users.

4.1.1 Roadmap

Roadmap refers to the plan that will guide the team members and show them how to reach the desired goals. It mainly includes a step-by-step plan for achieving the milestone (Anshul 2022). In most cases, the roadmap resembles a Gantt chart; however, if not, it looks like a step-by-step plan depicted on an arrow that goes from the beginning of the project through to its conclusion. Roadmap is different from the development methodology phases that have been illustrated before, in fact, the roadmap does not refer to the generic development, rather, it refers to the phases in terms of functionalities. For this project, the roadmap is not considered a Gantt chart but a step-by-step plan. The plan includes four main features with some sub-features:

- **Connection Check**

  Since the application performs network operations, it is necessary to check the connection and see if it is present at that particular time or not. A more meticulous check could be to make the application unusable if the connection is not available, but this is a more meticulous check and is not strictly required by our application as it is a prototype. For this reason, the check is only performed at the time of scanning.

- **Network Scanning**

  This feature will be responsible for scanning the network. With this operation, all the devices in the network will be found. To make things more efficient, this job will be assigned to a separate thread, in order to lighten the application and foster efficient usage on older devices too. More details will be found in the next sections. This feature includes two sub-features: device listing and manual insertion.

  - **Device Listing** The network scanning feature will retrieve a list of devices. To properly show this list, a fresh user interface is required. This phase is critical because it is not only about making the interface, it is about linking the user interface with the backend and storing the information on the device. This step is useful because, without it, the user would have to scan the network each time the program is launched to locate the devices. Thus, the information will be preserved even when the app is closed.
  - **Manual Insertion** When working with networks, several problems may occur. Suppose that a user is not able to scan the network, therefore can not use the app in the way it should be used. This sub-feature faces this problem: it allows the user to manually insert a device in the list that did not show up during the scan. This will be done by inserting manually the IP address of the device and then, after a background check described in the next sections, the device will be added to the list.

- **Updates Check**

  The core feature of the application. This feature will allow the application to tell the user if a security update is available for a particular device listed with the previous feature. The main job here is to detect the update using a dataset and to send the information to the notification system. For this reason, this feature works together with the notification system to provide the user with the correct functioning of the app.

  - **Dataset Building** The creation of the dataset is another crucial step. Typically, the initial step is to identify the source of the data, resulting in the "columns" of the dataset. This section addresses the question "What data will be contained inside the dataset?" After that, data are gathered through online research and entered into the dataset row by row.

- **Notification System** Every respectable application has a fully functioning notification system. In this case, the important is to avoid that

the user must enter the app to check if an update is available. Instead, with the notification system, the user will be notified without having to check every time. Notifications created will be the key to effective communication with the user about possible risks.

At the end, this is how the roadmap should look like:

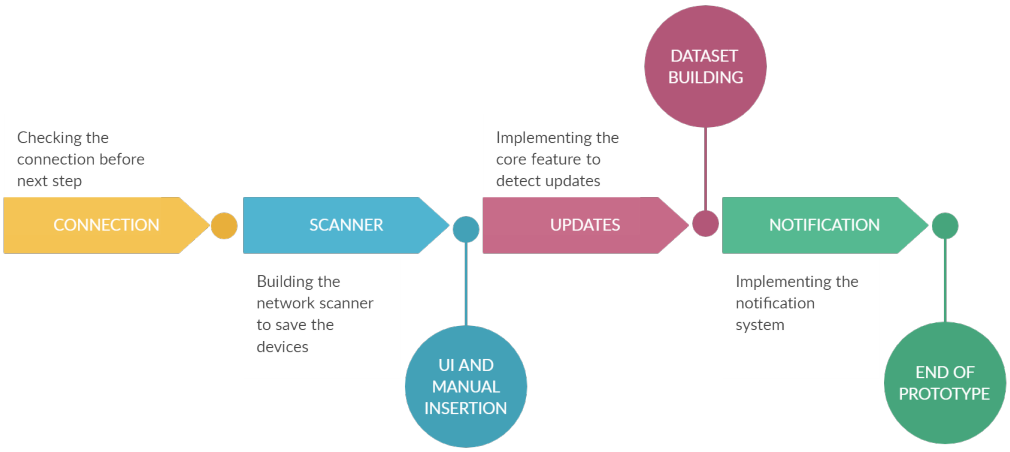


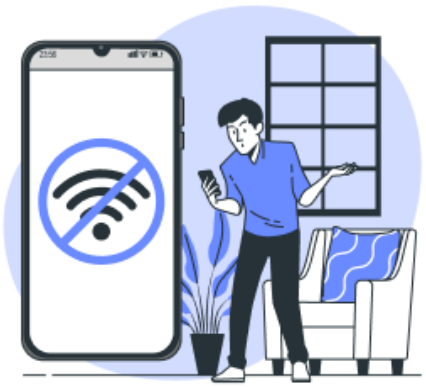
Figure 4.1: Roadmap of the project

4.2 Design

The design phase in software, but in this case app, development is the process of creating the visual and interactive design for it. During this phase, wireframes, mockups, and user flows are created to define the user interface and user experience. Since this is only a prototype, for this project wireframes have been hand-drawn, while mockups have been designed digitally with the help of Figma, a cloud-based design and prototyping tool that allows designers to create and collaborate on user interfaces (Figma 2016). User flows have been omitted since this is only a prototype and what the user can do is limited.

4.2.1 Mockups

In this small section, mockups related to the features listed before will be presented and explained. The design chosen is a minimalist design that aims to be easy to use and easy to understand. Mockups have been created using Figma, a software used for the design and creation of interfaces. The main idea was to create something that the user could use without thinking, the design should guide the user from the beginning to the end of its purpose. For this reason, the number of elements in the interface is reduced to give the user the least number of choices possible. The main color is light blue and all the mockups are designed by me. The first mockup is the *no connection* screen.



Oops!

Something went wrong. Check your device connection and come back!

Reload

Figure 4.2: No connection screen

This first mockup is related to the first feature of the app. As said, the app needs a connection to work. The connection check will ensure that the device is properly connected to a network. If not, the screen 4.2 will appear asking the user to reload as soon as the device has reestablished connection again. Once the connection is reestablished, the reload button will redirect the user to the homepage, therefore the user will be able to choose again the operation or to restore what he was doing before losing the connection.

Next feature is the scan of the devices. During the design of the app, it has been decided to allow the user to also insert manually a device, so that if the automatic scan is not accurate or it does not find all the devices, the user can insert it manually.

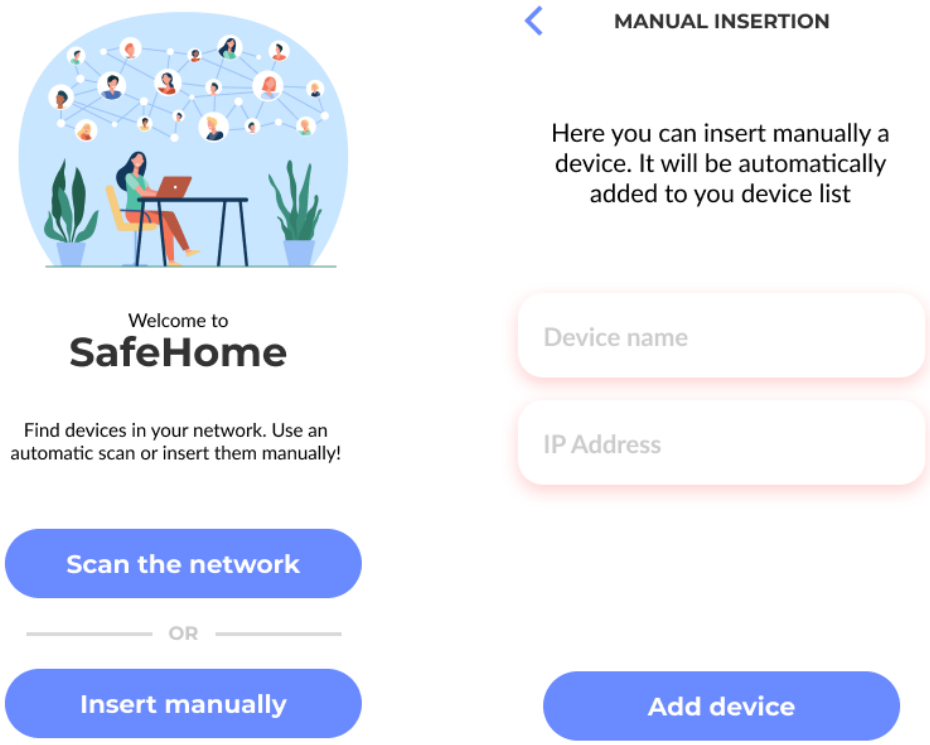


Figure 4.3: Homepage and a scan with no results

Deepening into this couple of screens, we can find on the left side a sort of homepage where the user has two options: one for the scan itself, and one to insert manually a device. The second option can still be found in 4.3 on the right side. The interface provides the user with two text boxes, one for the

device name and one for the IP address. Once clicked the *add device* button, the app will perform some operations to check the existence of this device and then will add it to the list.

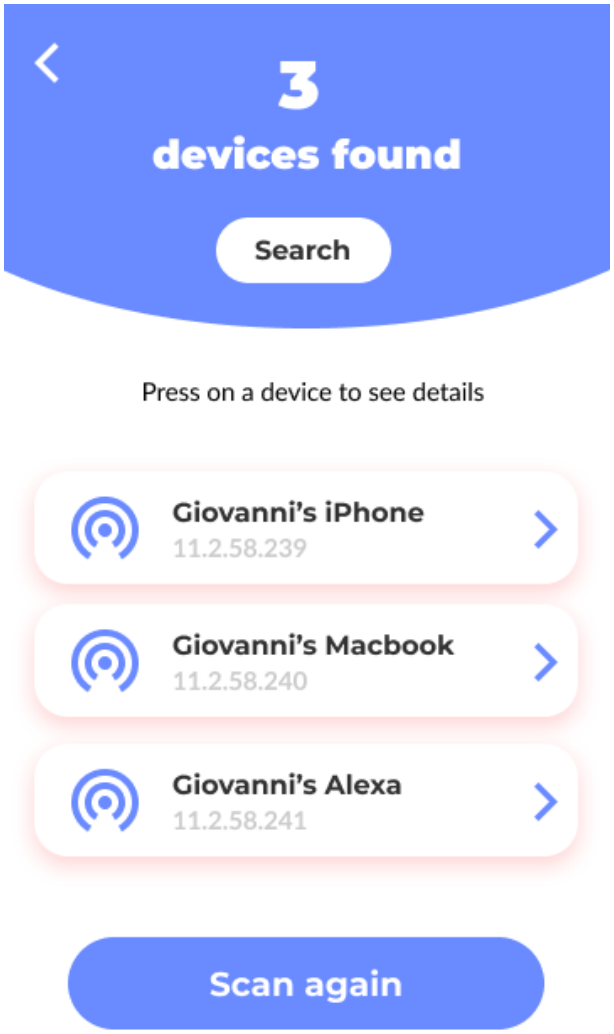


Figure 4.4: Devices' list screen

In Figure 4.4 it is possible to find the design of the result page of a working scan. The design can be divided into three sections: the first and upper section is an introduction to the page. It shows the number of found devices and the possibility to filter the results with a search button. The second section is dedicated to the actual list of results. A device is designed as a "card", where the user will find the name, its IP address, and an arrow icon which will lead

the user to click on the card to show more details about it. The third and last section is dedicated to a button that allows the user to scan again the network, in case of new or missing devices.

4.3 Platform discussion

A respectable mobile application nowadays is made for both iOS and Android. There are two main reasons why both operating systems have been chosen:

- **Reach** Even though both Android and iOS have a significant market share in the mobile operating system industry, it is possible to reach a larger audience by creating apps for both. For companies, this can be crucial since a larger user base often leads to more clients and incomes.
- **User preferences** Many people have strong preferences for one platform or the other, and by developing apps for both platforms, it is possible to satisfy the preferences of all the users of an application.

Overall, developing apps for both Android and iOS can help increase the reach, user base, and engagement of your app, which can be particularly important in today's competitive mobile app market.

Chapter 5

Implementation

This section presents the actual implementation of the project. Some code will be shown and explained, along with the design choices that led to that code. Not all the code will be shown since the building of the interfaces is long, and complex and results in a lot of lines. For convenience, therefore, only the code related to the features and the interface linked to those features will be explained. This section will have a technical language and will omit basic programming concepts.

5.1 Technologies involved

When it comes to the development of mobile applications, there is always a debate between those who think that developing in native languages is a better option, despite the double implementation and knowledge involved in the development. On the other hand, this is the time in which cross-platform development is reaching its peak. For this project, a cross-platform language has been chosen, and all the reasons for this choice can be found at the end of this chapter in the "Platform discussion" section. The language in question is Flutter. From the official website, Flutter is an open-source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase (Google 2014). Flutter is based on Dart, a programming language still made by Google. It has been integrated into Visual Studio Code as the main editor and, to deploy the application, a Realme 7 Pro has been

used as a real testing device. The testing on a real device was due to the impossibility of the emulator to get internet information. Along with Flutter, some packages have been used to help and ease the coding session. Following is the list of packages and a brief description of them and how they have been used in the project:

- **Connectivity Plus**

  A Flutter plugin that provides a simple interface for checking the network connectivity status of a device. It allows developers to determine whether the device has an active internet connection and what type of connection is being used (such as WiFi or cellular). It is important to check also the type of connection since the cellular connection is not valid in this project.

- **Network Info Plus**

  A Flutter plugin that provides a simple interface for accessing information about the device's network connection. It allows developers to get information about the IP addresses, subnet masks, and gateway addresses of the device's network interfaces. This plugin is crucial for the scanner feature, everything starts from getting the IP address made possible by this.

- **Dart Ping**

  A Flutter plugin that provides a simple interface for sending Internet Control Message Protocol (ICMP) "ping" packets to a specific IP address and measuring the time it takes to receive a response. It allows developers to determine whether a device is online and measure the network latency between two devices. This plugin is used to see which of the IP addresses is alive in the network.

5.2 Code inspection

Writing the code for a mobile application is a crucial step in developing a good, user-friendly app. The code is what allows the app to work, and once it

has been written, it must be tested and optimized to guarantee that it works properly across all platforms and devices for which it was designed. In this section, there will be an in-depth study of the code that has been written for Safe Home. The study will mainly focus on the "back-end" part, but a brief description of how the user interface works will still be done. Before starting, it is important to understand what is a design pattern, because the code inspection will make a distinction between three components extracted from the chosen pattern.

A design pattern is a technique that provides a general, reusable solution for regular software design problems. A design pattern can be seen as the actual development strategy after choosing the development methodology, Agile in this case. The chosen design pattern is Model-View-Controller, abbreviated in MVC. MVC is a design pattern introduced as a way to structure interactive applications in a modular way. This pattern decomposes functionality into three major components (F. Grove. and Ozkan. 2011):

- **Model** The model is a representation of a real-world object. More specifically, it can be seen as a class, in an object-oriented programming environment
- **View** The view is the component that provides information to the users through user interfaces. The view corresponds to the complete UI of the application
- **Controller** Controller is responsible for taking user actions and letting the application behave in the planned way. It is basically the back-end part of the application

By these definitions, it is possible to make a match with the components of Safe Home.

5.2.1 Models and appearance

As said before, the model is a representation of a real-world object. In this prototype of Safe Home, it was identified one main model to describe, which

was the Host, representing a single device from a network perspective. This class was designed to describe a host in the network. As a host, some basic information is needed such as the IP address and the MAC address. In addition, since the host matches with a device, a name has been added to the class to identify the device in a more user-friendly way, so that the user would not see only the IP address but also the device's name, for example, "John's iPhone". The model of the host can be found in the appendix.

The view is what users are going to see in the application. To build interfaces Flutter, just like React, uses **widgets**. From the official website, Widgets describe what their view should look like given their current configuration and state. When a widget's state changes, the widget rebuilds its description. In Flutter, everything is a widget, also the application itself. The coding part of the user interface is quite long due to the style of Flutter and, for this reason, following there will be only a part of the code explained, in particular the Results page. As soon as the scanning is completed, the user will be sent to another screen where he will see the results of the scan with all the devices found. The same page will be built when the scan will not find any devices. In the next pages, there will be just some snippets of code illustrating the most important part of the project. First, there will be the construction of the device list, then in the next paragraph, the reader will find the check connection function and the scanning function.

```
ListView.builder(
    itemCount: widget.activeHosts.length,
    itemBuilder: (context, int index) {
      return Container(
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(20),
        ),
        child: ListTile(
          leading: Icon(
            Icons.wifi_tethering,
            color: shBlue,
            size: 50,
          ),
          tileColor: Colors.white,
          title: Text(
            widget.activeHosts[index].name,
            style: GoogleFonts.montserrat(
                fontSize: 16, fontWeight: FontWeight.bold),
          ),
          subtitle: Text(
            widget.activeHosts[index].ip,
            style: GoogleFonts.lato(
                fontSize: 15, fontWeight: FontWeight.bold),
          ),
          trailing: Icon(
            Icons.arrow_forward_ios,
            color: shBlue,
          ),
        ),
      );
   },
),
```

Listing 1: ListView of scan's results

This snippet of code is only a small part of the user interface here. This is probably the most complex screen of the application. The part shown here explains how the devices' list is generated once the scanning is completed. Flutter uses a ListView with its builder.

ListView.builder creates a scrollable, linear array of widgets that are created on demand. In this case, the widget that is going to be created iteratively is a *ListTile*, which is a simple card with information in it. The number of items created is established by one of the parameters, which is *itemCount*. All the information to generate the list is stored in **activeHosts**, which is a list of Host passed in the constructor of the interface by the scanning process. The length of the list will be used as the number of cards to generate and the name of the hosts will be used to display the name and the IP address of the devices on the cards. Plus, the ListTile widget allows insertion of icons at the beginning and at the end of the widget to decorate it, in this case, an icon for the wifi and an icon for the arrow have been used.

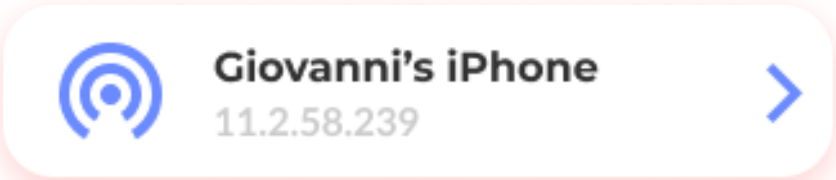


Figure 5.1: Device card in the results' list

5.2.2 The logic of SafeHome

The logic of an application represents the controller discussed before. The logic explains how the application thinks and works. It can be compared to the back-end in web development since it is something it is not seen by the user but only used. The logic aims to provide the user with a functional application without knowing how it works and how it is implemented.

In SafeHome, the logic is hidden in several steps, the first one is the scanning feature. The user is allowed to use the feature but is not allowed to know how it works. The same thing happens for the connection check, the updates check, and the functioning of the notification system. The connection check is a very good example of logic. As will be shown, the connection check is an important preliminary phase for the usage of the application. Without checking the connection, the application would switch to the scanning phase which will never be executed, causing the malfunctioning of SafeHome. Checking the connection must be considered as a wall that can be crossed only if the condition is satisfied, the availability of internet through wifi connection. If there is no availability, the application must drive the flow of actions and block the usage until the connection is available again.

Scanning the network also is another example. When a new scan is requested, the application will delete previously found devices and will start over again the scanning process. This is not a background process, at least not in this prototype, so the system needs to block any other possible activity to the user by showing a loading screen with no other option. The only thing the user can do is going back to the homepage, but this would stop the scan because it will be rebuilt the widget of the home. Afterward, there will be a diagram that will show the flow of the application starting from the scanning request. As said, checking the connection is the preliminary phase, so before showing the flow, a small snippet of code to illustrate how the connection is checked is shown.

```
void checkConnection(BuildContext context) {
    subscription = Connectivity()
        .onConnectivityChanged
        .listen((ConnectivityResult result) async {
      isConnected = await InternetConnectionChecker().hasConnection;

      if (!isConnected) {
        Navigator.of(context).pushNamed("/no-connection");
      }
   });
}
```

Listing 2: Connection Check feature

Another package has been used to ease the implementation, which is *internet_connection_checker*. The function in the first moment declares a variable *subscription* and assigns to it a listener coming from the package Connectivity Plus illustrated before. This listener works in such a way that its value changes as soon as the connection status of the device changes. When the status changes, the code inside the inner brackets is executed and a boolean variable *isConnected* awaits a value from *hasConnection* of the Internet Connection Checker package. Then, the function makes a simple control by checking the value of isConnected. If the value is true, the application keeps working with no interruptions. Otherwise, if the value is false, the function sends the user to the No Connection screen. In that situation, there will be a button to allow the user to reload the page so that if there is a connection again, he can be sent again to the homepage.

Once the connection is checked, the application can proceed to scan the network. The general flow is illustrated as follows:

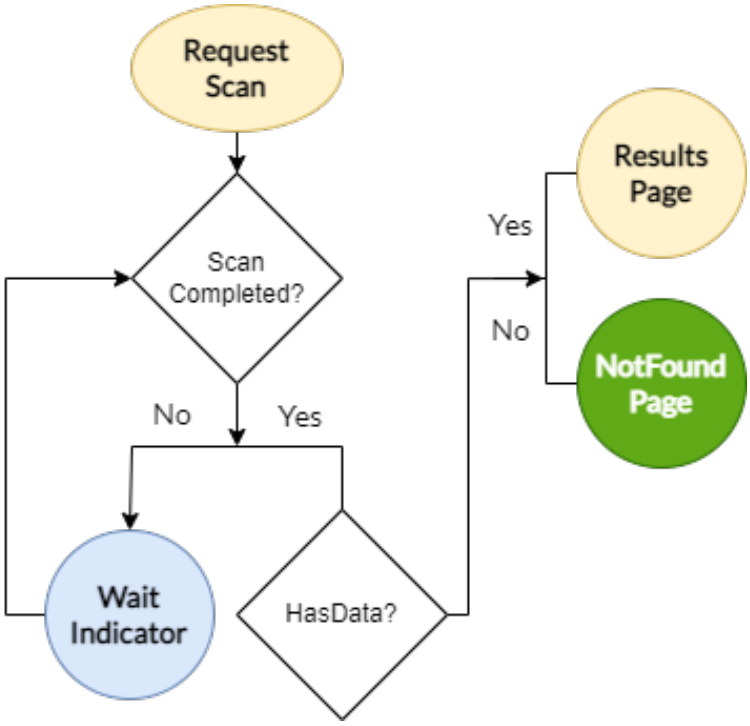


Figure 5.2: Genaral flow of the application

In this diagram, it is possible to follow the flow of the application starting from the moment in which the user presses the button to scan the network. It is implicit that the connection check has been successfully passed. Until the scan is completed, a widget with a progress indicator is returned. As soon as the scan is completed, it is important to check if the list of hosts is populated or not. If it has hosts inside, the results page is returned, else, the not found page is returned as an error. To complete the explanation, following there will be the snippet of code for the network scanning since it is important for this work.

```
Future<List<Host>> scanNetwork() async {
    List<Host> foundHosts = [];
    await (NetworkInfo().getWifiIP()).then( (ip) async {
        final String subnet = ip!.substring(0, ip.lastIndexOf('.'));
        for (var i = 2; i < 255; i++) {
          String ip = '$subnet.$i';
          await Ping(ip, count: 5).stream.toList().then(
            (result) async {
              if (result.last.summary!.received > 0) {
                foundHosts.add(Host(ip, ip));
              } else { print("${ip} is NOT a host!"); }
            },
          );
        }
      },
    );
    return foundHosts;
}
```

Listing 3: Scan the network feature

The scanNetwork() function needs a little attention to be understood. As can be seen from the signature, it takes no parameters and retrieves a Future List of Hosts, which means that the list will be affected by asynchronous data. The first step is initializing the list of hosts as an empty list. Then, using the package Network Info Plus illustrated before, the function will await for the instance to get the IP address of the wifi. As soon as it gets it, the subnet variable is created by taking the substring of the IP address and truncating it from the last dot until the end. Now, the core loop of the application begins. This for loop will iterate from 2 to 254 included, and, for each iteration, it will declare a variable *ip* and assign to it the value of the subnet plus the current loop counter. Then, with this variable, it will send a **Ping** to this IP address and it will wait for a response. If the address responds, it will create the Host

object with this address and add it to the foundHosts list. At the end of the loop, it will retrieve the list with all the alive IPs, and this is the moment where the connection in the previous FutureBuilder will stop and the snapshot will retrieve the data.

Chapter 6

Results

This chapter highlights what it was possible to achieve with the development of SafeHome. We will summarize the main contributions of this work and how it can contribute to this field of study. On the other hand, we will see some limitations as well, we will discuss and reuse them in the next chapter to address future works.

SafeHome is a mobile application prototype for Android and iOS written with the Flutter framework. This study shows how and why it was designed and implemented with a particular focus on the actual necessity of such an application. In the text, along with the description of the prototype, there is a description of the complex process of software development, which is subject to implementation bugs and design flaws since it is made by human people. These bugs and flaws can directly lead to vulnerabilities that can affect every kind of software. To solve vulnerabilities and avoid their exploitation, developers release security patches that fix bugs and interrupt channels that attackers have to steal data and compromise entire systems. SafeHome is not an application that practically solves this problem but helps average internet users to be more aware of software updates, through which security patches are installed on devices and solve vulnerabilities. To do so, a prototype of SafeHome was developed using the Agile development technique which allows a strong separation between functionalities and their correct implementation and testing. The following section will evaluate what has been described through some

interviews taken with people of different backgrounds and familiarity with technology. It is important to highlight again that SafeHome, at this stage, is only a prototype, and does not cover the whole solution. For this reason, the research in this thesis presents some limitations regarding technologies available nowadays and also limitations related to the time available to develop the project. To better understand this concept, after describing the results from interviews, we will list all the limitations that affected the development of the actual application.

6.1 Evaluation

After the development of the prototype, it is important to validate the results. This can be done by entrusting the prototype to different people and collecting all their feedback to ensure that the work has reached its purpose. The validation was conducted by different people with different backgrounds, ages, and familiarity with technology. Following is the table which shows the interviewed people:

Person	**Age**	**Sex**	**Background**	**FwT**
Marie	53	F	Housewife	5
Jake	82	M	Ex-Teacher	2
David	52	M	IT Employee	9
Sophie	19	M	Student	8
Amber	33	F	Lawyer	6

Each person answered 5 different questions. These questions helped to understand how the examined person faces the problem of updating devices and if he or she can understand all the risks. The last questions, instead, will be the actual feedback about the application in terms of user interface and user interaction.

ID Question	Question
Q1	How often do you update your devices?
Q2	Are you aware of what may occur if a device is not being updated for a long time?
Q3	Do you think an app could help?
Q4	Do you find it easy to navigate inside the application?
Q5	Would you update a device after receiving a notification?

The interview session was conducted as follows: the examined person answered the first three questions, then an Android smartphone was provided to test the application under instructions. After the usage, the person answered the last two questions. Following are the answers related to each interviews person:

- **Marie**

  1. **"How often do you update your devices?"**
     "I actually never update my devices. When it happens, it's because my husband does it but I never know why."
  2. **"Are you aware of what may occur if a device is not being updated for a long time?"**
     "No, don't know anything about it."
  3. **"Do you think an app could help?"**
     "As long as I don't know its importance, it's quite difficult for me to say if it could help."
  4. **"Do you find it easy to navigate inside the application?"**
     "I really like the design, the colors and the minimalism. Very easy to use because it is straight to the point."
  5. **"Would you update a device after receiving a notification?"**
     "If the notification says that it is urgent, I would do that."

- **Jake**

  1. **"How often do you update your devices?"**
     "I don't know what an updated is. I just use my phone and laptop to do basic things."

2. **"Are you aware of what may occur if a device is not being updated for a long time?"**
   "No, should I?"
3. **"Do you think an app could help?"**
   "For what I know, today there is an app for almost everything, so I guess so."
4. **"Do you find it easy to navigate inside the application?"**
   "The presence of a limited number of objects helped to navigate the application in an easy way, but unfortunately I still don't understand the actual purpose."
5. **"Would you update a device after receiving a notification?"**
   "I tend to forget things due to my age, in most cases, I would probably miss the notification."

- **David**

1. **"How often do you update your devices?"**
   "Almost every time a huge update comes, especially when I know that a lot of improvements have been made by the companies. I don't usually update when there are only small fixes."
2. **"Are you aware of what may occur if a device is not being updated for a long time?"**
   "Yes, working in the IT field makes me aware of it."
3. **"Do you think an app could help?"**
   "It depends on the features, what the app can offer."
4. **"Do you find it easy to navigate inside the application?"**
   "The app is straight to the point, intuitive, and within everyone's reach. I really like the possibility to keep a list of all the devices and, for this reason, I would like to see also their names in a future implementation."
5. **"Would you update a device after receiving a notification?"**
   "In my case, the problem is not the reminder. Sometimes when

I have to update a device I tend to skip it only because I need that device in that particular moment. It could be a great idea to implement other types of notifications, for example a notification before going to sleep and the usage of the device is not needed anymore."

- **Andrew**

  1. **"How often do you update your devices?"**
     "When a newer version of iOS of my iPhone is released"
  2. **"Are you aware of what may occur if a device is not being updated for a long time?"**
     "I know that there is a reason why small updates are released but I don't know exactly why. Probably something about security."
  3. **"Do you think an app could help?"**
     "If it's important, yes."
  4. **"Do you find it easy to navigate inside the application?"**
     "I use to navigate in a lot of apps everyday and when I find apps like this with such a minimal design I'm always happy."
  5. **"Would you update a device after receiving a notification?"**
     "A notification system will for sure help me to not forget. An advice could be put inside the app a section where the user could read and understand why it's important to use it."

- **Sophie**

  1. **"How often do you update your devices?"**
     "Actually never, it happened probably two or three times since I've been using devices in general."
  2. **"Are you aware of what may occur if a device is not being updated for a long time?"**
     "No, I don't know anything about it, but I'm curious now."
  3. **"Do you think an app could help?"**

"It depends on the importance. Is it really important to update a device? Then an application could help."

4. **"Do you find it easy to navigate inside the application?"**
   "Yes, I really like the design. Also, the choice of colors is appropriate in my opinion."
5. **"Would you update a device after receiving a notification?"**
   "After understanding the risk, I will definitely update when a notification comes."

6.1.1 Interviews discussion

All the answers given through the interviews can be used to summarize the general opinion about this work. As said in the previous chapters, vulnerabilities are a particular topic that is not close to everyone, instead, it is considered a specific topic for people who study in this field. Since the purpose of this work is to warn people about what might happen if someone does not update devices, it is crucial to educate people about vulnerabilities. From the answers, it can be seen that only the person working in the IT field knows about vulnerabilities, meanwhile, a person who still has a proper grade of familiarity with technology does not know a lot about it. Nevertheless, people with a higher grade of familiarity with technology are more inclined to respond to a notification and upgrade their devices.

6.2 Limitations

The solution to the problem provided during the previous chapters is comprehensive, intuitive, and easy to follow with the features sequence of the roadmap. For each feature, an in-depth study has been made, starting from the connection check to the notification system. Since the roadmap provides features sequentially, it is easy to understand that all the features are propedeutic for the next ones. So, the first problem that led to the limitations is that, if a feature is not properly completed, the next one could not be implemented correctly. In this prototype, two out of four features have been

completely implemented.

The first feature that had implementation problems is the building of the dataset and its usage. The dataset was designed to have two sources of data, the brand name and the URL to call the remote method. After a lot of research, sadly only a few brands made available APIs to get the latest version of the operating systems. Even if some of them did, a possible implementation of the calls would have been long and tricky for a simple reason: the return value is never a single number such as the operating system's version, but it is usually a huge JSON file that differs from brand to brand. At this point, the only solution would be to try to parse the JSON file to get only the value of the version, but since the file always differs because each company has its own JSON standard, we should have had a parser for each call, which is unbelievable for a prototype of an application.

Another limitation is related to the scan of the network. This is only for informative purpose since the scan in the application is working in a more than acceptable way. Sometimes the scan does not provide the whole list of devices. This is because, as explained in the previous chapter, the scan function uses the ping method to see if a host is alive or not. The ping method has some security limitations regarding ports that bring some devices to not respond to the ping, therefor it seems like the host is not alive even if it was. This is a common problem, tested with other scanners as well as with a simple ping on the command line. Still, the method is acceptable since the percentage of the found device is way higher than the others. Following, there is a diagram that recalls the roadmap shown previously by showing the four main features of the application. Highlighted are the features implemented at the end of this study, the non-highlighted features are meant to be implemented after future works and the resolution of the listed limitations.

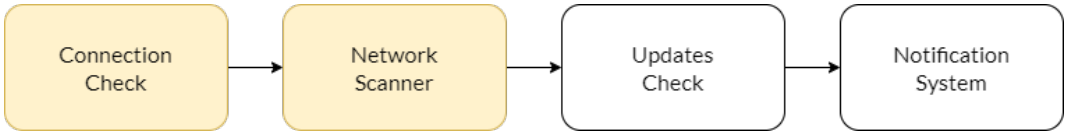


Figure 6.1: Updated roadmap with steps done

Despite these limitations, the thesis is in itself a whole research on this field of study and provides a fully functional solution to the problem if correctly implemented in the application. An in-depth analysis will be made in the next chapter.

Chapter 7

Discussion

This chapter discusses the obtained results with some personal considerations. Also, the chapter analyzes this work from an ethical point of view, discussing all the possible problems that can occur in terms of privacy.

SafeHome is a prototype of an application that aims to increase devices' protection by letting users apply patches as soon as possible. Patches must be executed by the users but they are not aware of their importance nor of the consequences that can occur. We have discussed before why users are not willing to update their devices, this can be due to a matter of time, they do not want to spend time doing an update, or to a case of unawareness, they take this topic lightly and in their minds, nothing would happen by delaying the update.

SafeHome partially solves this with its ability to persuade users. From a technical point of view, SafeHome aims to convince users to update devices at the same time they receive the notification. In particular, SafeHome uses a different way of communication instead of standard system messages, which also related works demonstrated that they are not properly effective. The ability of SafeHome to reformulate update messages helps to reach a larger portion of users since they will be more understandable and attention-grabbing. The need for a message reformulation comes from interviews conducted before, the majority of users are not properly aware of the risks, so they need something that aims to describe in simple words what the update is about, trying to

reach all the possible users, no matter their technological background. However, SafeHome is still in the development phase, and the provided application is a prototype that does not fully reach the final purpose of this research study.

Despite the limitations that the prototype currently has, after further development that will be listed in the conclusion chapter, SafeHome has the potential to become an essential tool for improving devices' security by increasing their protection from cyber threats by applying patches in the shorter time possible.

7.1 Considerations

In this section, we are going to discuss how results have an impact on our current society, considering both positive and negative aspects. The main improvement that SafeHome brings to society is the general increase of security in homes. For this reason, attackers have fewer opportunities to harm devices, unless they use zero-day vulnerabilities. While this is a positive aspect for users, attackers may start finding new ways to attack devices by focusing only on zero-day vulnerabilities. Of course, an attack that exploits a zero-day will be, in the end, successful since SafeHome improves security by persuading users to apply patches, which means that vulnerabilities are already solved at this stage.

The possibility to find new zero-day vulnerabilities to exploit is not the only negative aspect. When SafeHome can help a user to apply a new patch, from a user perspective this can be soothing, causing the user to think that he is completely safe. This is, of course, false. Security is a large topic and comes from different user behaviors and, for this reason, users are responsible for their security every time they are connected to the internet. The reason why they are responsible is that every application, software, or website cannot be 100% secure, therefore, they cannot guarantee that a user is fully safe when using their services.

7.2 Privacy discussion

When talking about security, it is very important to make an ethical study and see what are the limitations when working with networks. The study conducted here has a crucial point where ethical considerations can be done: the scan of the networks. Nowadays there is particular attention to privacy, users want to feel safe and want their data to be protected. This is something that in software development must be taken into consideration, especially when we talk about security. There are several considerations to take into account when scanning networks, we are going to examine them one by one.

First of all, thinking about privacy, scanning of the networks can easily reveal sensitive information. GDPR, the acronym for General Data Protection Regulation, talks about IP address, which is the information we gather in this study, and says: "These identifiers refer to information that is related to an individual's tools, applications, or devices, like their computer or smartphone. The above is by no means an exhaustive list. Any information that could identify a specific device, like its digital fingerprint, are identifiers" (*Personal Data for GDPR* 2019). The list appointed in the citation includes cookies, RFID tags, and IP addresses.

This brings the discussion towards another point: the ownership of the network. Since the GDPR considers IP addresses as personal data, from an ethical perspective, scanning networks that we do not own and storing the data could have ethical implications. Nevertheless, SafeHome does NOT save any of that information in any database but only in the cache of the device. The ethical consideration is still valid since future developments could start using a database to store data.

Chapter 8

Conclusions

This is the final chapter of this thesis. This place is where we will summarize what has been done in order to move the discussion to what it could be done in the future in further research and implementations.

From the beginning of this thesis, the central attention was given to the research question found: *How can we get average users to be more aware of updating their devices?*. What it turns out from this question and the previous research is that users are not aware of the risks. For this reason, other two sub-questions were provided to decorate the main questions:

1. **RQ1:** *How can we highlight possible risks of not updating devices?*
   The first step to let users be more aware of updating devices is to let them know what are all the possible risks that may occur when a device is not properly updated to the latest version.

2. **RQ2:** *What is the most efficient way to inform users about possible risks?*
   The second step regards the practical way used to alarm users about the possible risks.

Currently, the only way that users have to understand something about the risk is to read carefully the system messages that come when a new update is available. The problem with these messages is that most times they are not understandable if the user has not a good technological background. This

results in indifference from users and then in carelessness about the update. This thesis provides an answer to the first question by sending messages in a new form, using a vocabulary understandable by the majority of users, and going straight to the point to highlight what the update is about and what are the risks if it is not executed.

Moreover, the answer to the second question has also been provided. This question aims to identify an effective way through which these messages can be delivered to get the best effectiveness. How can custom messages be delivered to the user? Through a custom mobile application that can detect the presence of a new update of a particular device and can send a message to alarm the user.

Referring now to the main research question, it is possible to provide a way to get average users more aware of updating devices and this thesis shows a possible way to achieve this purpose, even though the provided solution is only a prototype and it is not completed due to technical limitations.

8.1 Future works

The state of the prototype of this project allows the next research to extend the project, especially in terms of implementation. The previous chapter highlighted the limitations faced during the development, and it is from here, future works should start working sooner or later. We have two main areas where future works can improve the project, and these areas match the two main features of the application. More precisely, the application can be improved following two different paths:

1. The first path is the most exhaustive. The goal here is to prioritize the completion of the app while maintaining a raw state of the entire system. What you want to do then is to complete the app by implementing the remaining features. This may be a good strategy, probably the best, however, we must consider the possibility of having problems with earlier

features as they are implemented superficially (devices not found).

2. The other way, however, aims to refine all the features already implemented before implementing the new ones. We are talking about the exact opposite of the previous path, but in this case, a complete system would be achieved in more time.

There is a lot to refine in the project. Even though what has been implemented works quite well, there is always something that can be improved. The scanning feature, as said in the limitations, is not able to detect all the devices due to security reasons and the ping method. What can be done in the future is trying to change strategy, leaving behind the ping method, and trying to implement another solution that it is maybe able to detect everything. To improve this feature, since we are considering only IP address-based devices, it could be useful to expand this search to devices based on other protocols such as zigbee or z-wave

When improvements will be done, maybe there will be the possibility to have some APIs from the companies that match a common standard. Working with different standards to get the same type of value is not the best, it implies that it will need a different parser for each of them, and this takes a lot of time and effort. Also, the hope is that one day all the companies that at this moment don't provide APIs, will in the next future, in order to expand the dataset as much as possible. When the application will work completely, the idea is to expand it to the whole concept of Smart Home, so that it could include also other kinds of devices like fridges, washing machines, and other household appliances.

Bibliography

Alshenqeeti, Hamza (2014). "Interviewing as a data collection method: A critical review". In: *English linguistics research* 3.1, pp. 39–45.

Anshul (Dec. 2022). *What is a roadmap? (definition and examples)*. URL: https://chisellabs.com/glossary/what-is-roadmap/.

Asamba, Micah (Feb. 2023). "Data Analysis". In.

Ayeni, Toba (June 2019). "The Barriers To Open Internet In Africa". In: pp. 43–55.

Barlow, Jordan et al. (Aug. 2011). "Overview and Guidance on Agile Development in Large Organizations". In: *Communications of the Association for Information Systems* 29, pp. 25–44. DOI: 10.2139/ssrn.1909431.

Brandtzaeg, Petter, Jan Heim, and Amela Karahasanovic (Mar. 2011). "Understanding the new digital divide - A typology of Internet users in Europe". In: *International Journal of Human-Computer Studies* 69, pp. 123–138. DOI: 10.1016/j.ijhcs.2010.11.004.

Creswell, J.W. (2014). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. SAGE Publications. ISBN: 9781452226095. URL: https://books.google.it/books?id=PViMtOnJ1LcC.

F. Grove., Ralph and Eray Ozkan. (2011). "THE MVC-WEB DESIGN PATTERN". In: *Proceedings of the 7th International Conference on Web Information Systems and Technologies - WEBIST,* INSTICC. SciTePress, pp. 127–130. ISBN: 978-989-8425-51-5. DOI: 10.5220/0003296901270130.

Fagan, Michael, Mohammad Khan, and Ross Buck (Oct. 2015). "A study of users' experiences and beliefs about software update messages". In: *Computers in Human Behavior* 51, pp. 504–519. DOI: 10.1016/j.chb.2015.04.075.

Figma (Oct. 7, 2016). *Figma: A cloud-base design and prototyping tool*. URL: https://www.figma.com/.

Flynn, Jack (Nov. 2022). *16 amazing agile statistics [2022]: What companies use agile methodology*. URL: https://www.zippia.com/advice/agile-statistics/.

Goel, Jai Narayan and B.M. Mehtre (2015). "Vulnerability Assessment & Penetration Testing as a Cyber Defence Technology". In: *Procedia Computer Science* 57. 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015), pp. 710–715. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2015.07.458. URL: https://www.sciencedirect.com/science/article/pii/S1877050915019870.

Google (Nov. 23, 2014). *Flutter is an open source framework by Google for building multi-platform applications*. URL: https://flutter.dev/.

Kaluarachchi, Pubudu, Chris Tsokos, and Sasith Rajasooriya (Jan. 2016). "Cybersecurity: A Statistical Predictive Model for the Expected Path Length". In: *Journal of Information Security* 07, pp. 112–128. DOI: 10.4236/jis.2016.73008.

Kawulich, Barbara (May 2005). "Participant Observation as a Data Collection Method". In: *Forum: Qualitative Social Research* 6.

Mishra, Alok and Ziadoon Otaiwi (Nov. 2020). "DevOps and software quality: A systematic mapping". In: *Computer Science Review* 38, p. 100308. DOI: 10.1016/j.cosrev.2020.100308.

Mohajan, Devajit and Haradhan Mohajan (Feb. 2023). "Straussian Grounded Theory: An Evolved Variant in Qualitative Research". In: *Studies in Social Science Humanities* 2, pp. 33–40. DOI: 10.56397/SSSH.2023.02.06.

Ojuwoni, Adedapo (Jan. 2022). "The Role of Security Updates and Patches in Addressing Cyber Security Threats and Vulnerabilities: A Case Study of Recent Cyber Security Attacks". In.

Personal Data for GDPR (Feb. 2019). URL: https://gdpr.eu/eu-gdpr-personal-data/.

Rajivan, Prashanth, Efrat Aharonov-Majar, and Cleotilde Gonzalez (Jan. 2020). "Update now or later? Effects of experience, cost, and risk preference on

update decisions". In: *Journal of Cybersecurity* 6. DOI: 10.1093/cybsec/tyaa002.

Rapid7 (2021). *Vulnerabilities, Exploits, and Threats.* URL: https://www.rapid7.com/fundamentals/vulnerabilities-exploits-threats/.

Rosencrance, Linda (2021). *What is a vulnerability assessment (vulnerability analysis)? definition from searchsecurity.* URL: https://www.techtarget.com/searchsecurity/definition/vulnerability-assessment-vulnerability-analysis.

Roser, Max, Hannah Ritchie, and Esteban Ortiz-Ospina (July 2015). *Internet.* URL: https://ourworldindata.org/internet.

Royce, W. W. (1987). "Managing the Development of Large Software Systems: Concepts and Techniques". In: *Proceedings of the 9th International Conference on Software Engineering.* ICSE '87. Monterey, California, USA: IEEE Computer Society Press, pp. 328–338. ISBN: 0897912160.

Sarker, Iqbal et al. (Nov. 2015). "A Survey of Software Development Process Models in Software Engineering". In: *International Journal of Software Engineering and its Applications* 9, pp. 55–70. DOI: 10.14257/ijseia.2015.9.11.05.

Shruti, M (Oct. 2022). *10 types of cyber attacks you should be aware in 2023.* URL: https://www.simplilearn.com/tutorials/cyber-security-tutorial/types-of-cyber-attacks.

Synopsys (Mar. 2017). *Top 4 software development methodologies.* URL: https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies/.

Tariq, Muhammad Adnan, Joel Brynielsson, and Henrik Artman (Aug. 2014). "The Security Awareness Paradox: A Case Study". In: DOI: 10.1109/ASONAM.2014.6921663.

Turner, Carl, James Lewis, and Jakob Nielsen (Jan. 2006). "Determining Usability Test Sample Size". In: vol. 3.