# POLITECNICO DI TORINO

## Faculty of Computer Engineering
Master's of Science Degree in Data Science

Master's Thesis

# Security communications through IoT devices: a tested prototype


**Relator:**
Prof. Riccardo Sisto
**Co-Relator:**
Prof. Lorenzo De Carli

**Candidate:**
Antonio Mignano

October 2019

# Abstract Eng

Nowadays, IoT technologies are becoming ubiquitous inside domestic environments, but even though these devices help users during their daily life, people are often unaware of the implication of their presence. Companies tend to create simpler interface that hide the complexity to the end user and make people unaware and uninvolved in security processes. Among the reasons for companies to follow this trend there are **(i)** the behavior of people that tend to avoid security decision [6, 18] and **(ii)** the convenience, economically speaking [16].

Studies have shown that neutral to negative attitudes dominate the end user experience with security technologies: frustration, pragmatism and futility. People are often incapable of distinguishing problems, tending to think of them as unique and hence try to find a unified solution. This kind of mental model leads to some implications: some solutions may be rejected for being considered just partial and some may be mistakenly interpreted as complete when, in reality, they are not. Among the practices used by people to avoid security decisions we can find delegation. Security decisions can be delegated to technologies, individuals, organizations or institutions. Other studies [1, 3, 5] demonstrated how information presentation can influence the way in which people perceive security and get informed, as long as how habituation can be reduced through better UI design.

The **problem** becomes clear now: people need to be better involved into security processes and informed about automatic security decisions. Non-tech experts feel uncomfortable around security matters due to too many knowledge requirements, bad user interface and improper information presentation.

It is important to first understand what are the **limitations of current approaches** before trying to give a solution. Many studies took into account participants living in smart homes and using smart devices by their own choice meaning that these people are probably passionate about technology, early adopters and in general tech-savvy, hence more aware of security issues than people who do not have these characteristics.
Under the educational point of view, there is a clear lack of adequateness about threats that users face and they are not sufficiently explained, instead they focus on practical and actionable advice that can be ignored if not perceived as necessary.

The **goal** of this thesis was to create a new way of interacting with users regarding security matters, trying to better involve people without making them feel incapable of understanding and taking the proper actions and countermeasures when needed.

We achieved our project goal by designing and implementing a message-broker-style middleware that links a *custom application layer* to different devices that will work as interface from and to the user. The Message Broker was designed in such a way to be compatible with different applications (what we just referred to as custom application layer) that may want to use the Message Broker Network to communicate with the user.

The **final result** of the development phase was a multi-platform system capable of involving the user in security processes, both with the user intervention required and not. The software produced are an Angular/NodeJS web-app runnable on any environment (personal computers, servers, etc...) and different applications for the devices used to test the system: Emulated Smart TV, Android Smartphone and SONOS Speaker.

There were some **challenges** to overcome during the designing and development phases that worth to be mentioned: **(i)** a discovery protocol was developed and integrated into the Smart TV and the Smartphone app in order to allow the devices to be automatically discoverable without the user intervention. Some problems arose with the Android app because of the background processes' management of the OS. **(ii)** The Smart TV was emulated by an Angular/NodeJS web-app in order to speed up the development phase. **(iii)** We were unable to use the official HTTP API of the SONOS Speaker and delegated the communication protocol to a third-party library.

To **evaluate** the usability and utility of our new system we decided to conduct a user study. Summing it up: most of the participants were younger than 25 years old, which is significant because younger people are in general more habituated to technology. The majority of them declared to have very low understanding about network security. Almost all of the participants stated that they were comfortable with the interface of the system even though UI improvements are required. More than 60% of the participants understood the action requested by the system. Some concerns may arise since 30% of participants said they did not feel in control of the situation. In general, 60% of the participants will be willing to use a system like the one prototyped inside their own house, 30% is neutral and just 10% closer to the "no side" of the decision. Feedback received through open questions is more explanatory: some of the participants would have preferred to be notified in a different way, while those who were fine with the current method of notification would have wanted the notifications to be more "invasive". The suggestions on possible improvements mainly focused on the graphical aspect.

The main limitation of this work was due to time constrain. The project was big and ambitious and some parts were unavoidably left for **future work**. A better error handling has to be considered for future development. In order to reduce the codebase and not increase its complexity some edge case were not handled properly and may create confusion if the system is used by user without a deep knowledge of the technologies used. Although, there is consistent logging of all the actions the system performs, therefore it is easy to backtrack problems to their source. For what concerns the Message Broker, a database connection to a MongoDB is requested to obtain data persistence; the HTTP API require some sort of authentication and authorization to properly work on cloud environments; the decision system currently adopt a naive decision method based on which device is available and have enough capability to handle the request and hence improvable. For what concerns the devices: the smartphone app could use some

background service management improvement seen that, as for now, services may be randomly killed by the OS and the UI part of the push notification should be reviewed; as said the Smart TV was emulated, in future work it may be useful to implement the framework on a real Smart TV; the official SONOS HTTP API should be used in order to be fully aware of all the communications happening within the network.

To **conclude**, we managed to make what we planned. With promising feedback obtained by the user study we conducted, we hope the project will have a follow up. We are happy to say that this project was successfully completed and can be used for future researches and/or projects. The overall experience was gratifying and helped the student learn how to perform research and organize and plan a big project.

# Abstract Ita

Oggigiorno, le tecnologie IoT stanno diventando onnipresenti all'interno degli ambienti domestici, e anche se questi dispositivi aiutano gli utenti durante la vita quotidiana, le persone sono spesso ignare delle implicazioni dovute alla loro presenza. Le compagnie tendono a creare interfacce semplici che nascondono la complessità agli utenti finali tenendoli allo scuro e non coinvolte nei processi legati alla sicurezza. Tra le ragioni per cui le compagnie seguono questa tendenza ci sono: **(i)** il comportamento delle persone che tendono a evitare di prendere decisioni legate alla sicurezza [6, 18] e **(ii)** la convenienza in termini economici [16].

Studi hanno dimostrato che attitudini neutrali e negative dominano le esperienze relativi alla sicurezza (rispetto alle tecnologie) dell'utente: frustrazione, pragamatismo e futilità. Le persone sono spesso incapace di distinguere i problemi, tendendo a pensare a questi come un unico problema e di conseguenza cercando una unica soluzione. Questo tipo di modello mental porta a delle implicazioni: alcune soluzioni potrebbero essere rifiutate essendo considerate solo parziali e altre incorrettamente interpretate come complete quando in realtà non lo sono. Tra le pratiche utilizzate dalle persone per evitare di prendere decisioni legate alla sicurezza possiamo trovare la delegazione. Le decisioni riguardante la sicurezza possono essere delegate alle tecnologie, ad altri individui, organizzazioni o istituzioni. Altri studi [1, 3, 5] hanno dimostrato come la forma in cui le informazioni vengono presentate può influenzare il modo in cui le persone le percepiscono e ottengono informazioni, allo stesso modo come l' "abitudine" possa essere ridotta attravero interfacce ben progettate.

Il **problema** sembra chiaro adesso: le persone hanno bisogno di essere coinvolte di più nei processi relativi alla sicurezza e informate riguardo contromisure automatiche. I non esperti di tecnologia si sentono a disagio riguardo argomenti relativi alla sicurezza per via della troppa conoscenza richiesta, interfacce mal progettate and informazioni presentate non correttamente.

È importante, prima di provare a dare una soluzione, conoscere le **limitazioni degli approcci correnti**. Molti studi prendono in considerazione partecipanti che vivono all'interno di "smart home" per loro scelta, implicando che queste persone sono probabilmente appassionate di tecnologie, adottatori precoci e in generale molto più propensi alle innovazioni tecnologiche, dunque, molto più consapevoli di argomenti riguardanti la sicurezza rispetto a chi non ha queste caratteristiche.

Sotto il punto di vista dell'educazione c'è una evidente mancanza di adeguatezza riguardo i problemi che gli utenti affrontano e come vengono spiegati, invece, ci si concentra di più sul dare consigli pratici e subito utilizzabili che possono essere ignorati se non percepiti come nec-

essari.

L'**obiettivo** di questa tesi era di creare un nuovo metodo per interagire con gli utenti e provare a includerli di più nei processi legati alla sicurezza, evitando di farli sentire incapaci di comprendere e prendere i giusti provvedimenti e contromisure quando necessari.

Siamo riusciti a raggiungere il nostro obiettivo progettando e implementando un "message-broker-style middleware" che connette un *livello applicativo custom* a diversi dispositivi che funzioneranno da interfaccia da e verso l'utente. Il Message Broker è stato progettato in modo da essere compatibilecon differenti applicazioni (ciò a cui ci siamo appena riferiti come livello applicativo custom) che potrebbero voler utilizzare la rete del Message Broker per comunicare con l'utente.

Il **risultato finalte** della fase di sviluppo è stato un sistema multipiattaforma capace di coinvolgere l'utente nei processi di sicurezza, sia in caso di richiesta d'intervento da parte dell'utente stesso che non. I software prodotti sono una applicazione web sviluppata con Angular/NodeJS capace di funzionare su ambienti diversi (computer personali, servers, ecc...) e diverse applicazioni per i dispositivi utilizzati per testare il sistema: Smart TV emulata, Smartphone Android e altoparlante SONOS.

Ci sono state delle **sfide** da superare, che vale la pena menzionare, durante la fase di progettazione e sviluppo: **(i)** un "discovery protocol" è stato sviluppato e integrato nella Smart TV e nell'app per lo Smartphone per far si che i dispositivi potessero essere scoperti automaticamente all'interno della rete senza l'intervento dell'utente. Alcuni probemi sono nati con l'applicazione Android per via del metodo di gestione dei processi in background del SO. **(ii)** La Smart TV è stata emulata con un'applicazione web Angular/NodeJS per velocizzare lo sviluppo. **(iii)** Non è stato possibile utilizzare le API HTTP ufficiali dell'altoparlante SONOS e dunque abbiamo delegato il protocollo di comunicazione a una libreria di terze parti.

Per **valutare** l'usabilità e l'utilità del sistema da noi progettato abbiamo deciso di condurre un esperimento. Riassumendo: la maggior parte dei partecipanti avevano meno di 25 anni, che è significativo poiché i più giovani son in genere più abituati ad avere a che fare con la tecnologia. La maggior parte di loro ha dichiarato di avere poco conoscenza riguardante sicurezza delle reti. Quasi tutti i partecipanti hanno dichiarato di essere a proprio agio con l'interfaccia anche se dei miglioramenti sono richiesti. Più del 60% dei partecipanti ha compreso le azioni richieste dal sistema. Potrebbe preoccupare il fatto che il 30% dei partecipanti ha affermato di non sentirsi in controllo della situazione. In generale, il 60% dei partecipanti è disposto a utilizzare un sistema come quello proposto nelle loro case, il 30% è rimasto neutrale e solo l' 10% più orientato sul no. Il feedback ricevuto tramite le domande aperte è stato più esplicativo: alcuni dei partecipanti avrebbero preferito essere notificati in modo diverso, mentre quelli a cui andava bene il corrente metodo avrebbero voluto che le notifiche fosse più "invasive". I suggerimenti sui possibili miglioramenti sono concentrati principalmente sull'aspetto del sistema.

La limimtazione principale di questo lavoro è dovuta al tempo in cui è stato sviluppato. Il progetto era grande e ambizioso e alcune parti sono state inevitabilmente lasciate per **sviluppi futuri**. Una migliore gestione degli errori deve essere considerata. Per ridurre la dimensione del codice e non incrementare la sua complessità alcuni casi particolari non sono stati correttamente gestiti e potrebbero creare confusione nel caso il sistema dovesse essere usato da un utente senza una profonda conoscenza delle tecnologie usate. A ogni modo, c'è un logging consistente delle

azioni che il sistema esegue ed è dunque semplice risalire alla radice del problema. Per quanto riguarda il Message Broker, l'implementazione di un database Mongo è richiesta per ottenere la persistenza dei dati; le API HTTP richiedono qualche forma di autorizzazione e autenticazione per funzionare correttamente in ambienti cloud; il sistema decisionale adotta un sistema molto semplice basato sui dispositivi correntemente disponibili e con abbastanza capacità da gestire le richieste e può dunque essere migliorato. Per quanto riguarda i dispositivi: l'applicazione per lo smartphone ha bisogno di miglioramenti con la gestione dei processi in background che potrebbero essere terminati in modo casuale dal SO; come già detto, la Smart TV era emulata, potrebbe essere interessante implementare il framework su una vera Smart TV; le API HTTP ufficiali dello Speaker SONOS dovrebbero essere utilizzate al posto della libreria di terze parti per essere così consapevoli di tutte le comunicazioni che avvengono sulla rete.

Per **concludere**, siamo riusciti a creare quanto ci eravamo prefissati. Con dei feedback promettenti ottenuti dagli esperimenti con gli utenti che abbiamo condotto, speriamo che il progetto possa avere un seguito. Siamo davvero felici di dire che il progetto è stato completato con successo e può essere utilizzato per ricerche e/o progetti futuri. Complessivamente l'esperienza è stata gratificante ed ha dato allo studente la possibilità di imparare come svolgere una ricerca e organizzare e pianificare un grande progetto.

# Contents

*To my parents and brother who*
*have always been next to me.*

# Chapter 1

# Introduction

Nowadays, homes are getting smarter and smarter via internet-connected devices such as sensors, lights and locks, controlled by voice or other user-defined ways.

Even though these devices help users during their daily life, people are often unaware of the implications of their ubiquitous presence. The average level of knowledge is decreasing due to companies' tendencies to create interfaces that hide the complexity to the end user, making them unaware and uninvolved in the security processes. This happens for two main reasons: first, users try to avoid security decisions [6, 18] and second, because it is more convenient, economically speaking, to avoid implementing any security into products [16].

## 1.1 Problem

Studies [6, 15, 20] have shown poor adoption of security measures due to incomplete or missing *mental models*[1].

In a study conducted by Dourish et al. [6] they found out that neutral to negative attitudes dominated the end user experience with security technologies: frustration, pragmatism and futility. People tend to collect different categories of problems under the same umbrella and imagine and seek unitary solutions to these problems (e.g. antivirus protecting all the network instead of just a computer).
This kind of mental model leads to some implications: **first**, a solution that solves just one problem can be rejected by the user for being a "partial" solution, **second**, a solution that solves one problem may be mistakenly interpreted as providing solution for other problems and **third**, the focus on barriers distract from channels (e.g. new firewall but on a non-encrypted WiFi).

Among the practices of security there is delegation [6, 15], which means that people try to avoid security decisions. Usually they rely on technologies (e.g. SSL), other individuals (e.g. friends/family), organizations (e.g. IT department) or institutions (e.g. financial institution) to take

[1] A mental model is an explanation of someone's thought process about how something works in the real world. Source: `https://en.wikipedia.org/wiki/Mental_model`.

care of issues.

Finally, studies such the ones of Anderson et al., Bravo-Lillo et al. and Conzola [1, 3, 5] demonstrated how information presentation can influence the way in which people perceive security and get informed, as long as how habituation can be reduced through better UI design.

The problem seems to be clear now: the way in which users are involved in security processes is not good enough if not existing at all. Non-tech experts are often uncomfortable around security matters due to too many knowledge requirements, bad user interfaces design and improper information presentation. Also, with increasing demand of smart technologies more and more companies, both small and big, are trying to penetrate the market with their solutions. This process evolves to a race to the market, where little companies trying to compete with big ones, avoid implementing security measures to increase their profit margins.

This thesis' objective is to answer the following questions: is there a better way to involve the user in security processes? Doing so, what is the proper design of such systems?
Regarding the interaction with the user, what is the most effective way to present information and what is the best way to encode them in such a way they are form independent?

## 1.2 Limitations of current approaches

Many studies just took into account participants living in smart homes by their own choice, not those who deliberately decided not to use smart devices for security or privacy (or other) reasons.
Also, it is important to remember that smart home technologies are still new and under development. Thus, participants to studies are *early adopters* and they may be more willing to choose convenience, or be generally more tech-savvy, over security and privacy.

Under the point of view of education, there is a clear lack of adequateness about threats that users face and they are not sufficiently explained, instead the focus is on practical and actionable advice, leaving space for the advice to be ignored if not perceived as necessary. It is required to expose potential threats to users in ways they can understand and take action when needed.

As mentioned above, some companies may try to compete with others by producing cheap devices and not implementing any security. This is a clear sign that regulation is required from government agencies both in security requirements before bringing a device to the market and informing the user about potential threats and security issues. Obviously, this process is very complicated due to fast-changing technologies that often cannot even be standardized before becoming obsolete.

Since security is too important to let a software decide all by itself, automatic prevention systems are not yet widely used. The correct path to follow would be to include users in the decision processes taken by these automatic systems by leaving him/her the final choice about security measures, making sure he/she completely understood the risks. This is what this thesis'

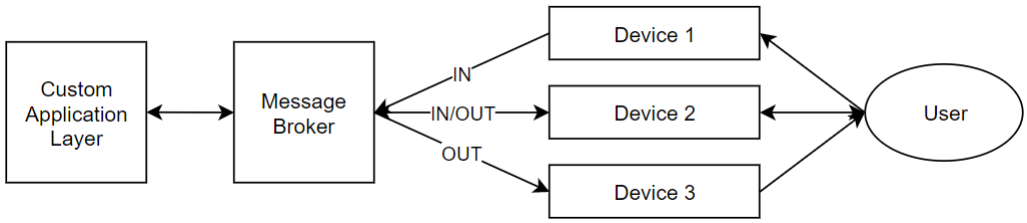



Figure 1.1: Solution concept

project is aiming to do.

## 1.3 Proposed solution

After all the consideration regarding security involvement, the path chosen is to create a new way to include the user in security processes through direct interactions.

This is achieved through the design and implementation of a message-broker-style middleware that links a *custom application layer* to different devices (e.g. smartphone, smartspeaker, smart tv) who will work as input/output interfaces from and to the user. The concept is better explained in Figure 1.1. The role of the custom application layer is to take care of the security processes (detection and intervention) and it is not of interest of this thesis.

Custom applications can send messages or request for actions to involve the user, the Message Broker will decide what is the best way to interact depending on the devices registered (and available) as interfaces.

This thesis' project is going to be part of a bigger work which involves an *Artificial Intelligence* capable of detecting intrusions or malfunctions inside a network. The above mentioned AI will be the application layer, generating messages and action requests to inform the user about threats and prevent/fix issues with user intervention when needed.

The development was split in two parts: the **first** part is the above mentioned *Message Broker* that works as core of the system, holding all the intelligence and taking the form of a runnable computer program, the **second** part is a *framework* used on smart devices, allowing them to be part of the system, taking the form of REST API exposed by the Message Broker.

More in detail, during the development, the Application Layer was emulated, the Message Broker ran on a computer and three devices were selected to test the framework on: **(i)** Smartphone with a simple Android application capable of showing push notification, **(ii)** Smart TV, emulated by a web-app ran on a Raspberry Pi and **(iii)** SONOS speaker. For each of these devices an application had to be developed. The choice was made upon the idea to cover as much capabilities as possible.

## 1.4 Main challenges

The main limitation was due to time constraint. There were not big problems during the development of the Message Broker. After properly planning on what technologies to use and the time schedule, the development went smooth with minor issues with some of the devices.

To speed up development a simple **discovery protocol** was developed and integrated into the Smart TV and Smartphone apps. It works on both platforms but with some difficulties on **Android** because of the OS management of background services. To solve the issue some practical test were done that led to a configuration of messages' timeouts and number of packets sent.

Another point to consider was the emulation of the **Smart TV**. It would have been more difficult to learn the Android TV framework, although it is similar to Android for smartphones, we decided to emulate it with an Angular web-app.

Some problem arose with the **SONOS Speaker**. The official HTTP API were partially working and the non-working end-points were not displaying any debug information. Also, the official community was not able to address the problem so we decided to switch to a third-party node library to handle the communication with the speaker with the downside to work only locally.

## 1.5 Summary of evaluation/result

We conducted a *user study* to evaluate the utility of the system and receive feedback on it.

Most of the participants were younger than 25 **years old**, which is significant because the younger they are the more prone to have lived their lives closer to technology. About 70% of the participants declared they have very low **understanding** about **network security**. Almost all of them stated they were **comfortable with the interface** used, even if they said there is space for graphic improvements. More than 60% said "a lot". This led to have more than 75% of the participants **understand the actions required** by the notification system. Regarding the **feeling of being in control** of the situation, we have more variance in the answers with 4 people (30%) saying not at all, and 6 people (45%) saying a lot. In general, 60% of the participants will be **willing to use** a system like this one inside their own house, 30% is neutral and just 8%is closer to the "no side" of the decision. Those who are in favor though would like to know more about the limitations of the system and the benefits it brings. Someone pointed out that the system itself may be prone to security issue, while someone else seemed to be bored by the "constant" notifications. Some of them would have **preferred to be notified** by a text instead of a push notification (regarding the phone), but those who were fine with the notification would have wanted it to be more "invasive", meaning that the phone should vibrate more and have a bigger icon in the notification bar. The **suggestions** about what can be **improved** mainly focused on the graphical aspect. Some of the participants pointed out that the notification should be more "educational".

The result are better discussed in Chapter 4.

## 1.6 Summary of rest of thesis

In **Chapter 2** we will explore what has already been done, talking about attitudes towards security and what influences them; security in IoT devices; some proposed solutions; overview of what HCI is and some concerns related to non-security issues.

In **Chapter 3** we will go further with the explanation of the system. Starting with some real life scenarios to better understand the utility of the Message Broker, we will then analyze all the parts that compose it. We will also take a look at the internal representation of the objects used by it to exchange messages.

In **Chapter 4** are presented the result of the user study conducted to test the usability of the system. We will also talk about the limitations to take into account when analyzing the data.

In **Chapter 5** is presented an overview of the process that led to the successful completion of this work. It will also be discussed the main limitations and what can be done in future developments to improve the Message.

In **Chapter 6** some final thoughts about the Message Broker and all the work that has been done that led to a working prototype of a new communication system.

## 1.7 Notes

Professor **Riccardo Sisto** from **Politecnico of Turin** as supervisor helped the student Antonio Mignano find a great growing opportunity conducting a thrilling research in a university abroad. The university was the **Worcester Polytechnic Institute** (WPI), a private research university in Worcester, Massachusetts. Professor **De Carli Lorenzo** from WPI as supervisor helped the student through all the phases needed to understand and learn how to conduct a proper research work.

# Chapter 2

# Related Work

In this chapter a summary of what has already been done is presented.
In Section 2.1 an overview of what are the main attitudes toward security. In Section 2.2 are discussed the factors that influence security behaviours. In Section 2.3 security in IoT devices is treated in a deeper way. In Section 2.4 are discussed the most relevant proposed solutions and advice to problems related to security. In Section 2.5 an overview of what has been done in the field of Human Computer Interaction and which are the best ways to design a warning message. In Section 2.6 are discussed some of the concerns that regard privacy and non-security matters.

## 2.1 Attitudes Towards Security

Many studies state that home computers/networks are insecure because they are administered/-operated by untrained users, such users are considered to be the weak link in computer security for several reasons: they are subject to social engineering and usually know very little about security and policies.

### 2.1.1 "Looking for Trouble: Understanding End-user Security Management"

Gross and Rosson [9] challenge their view, arguing that users often have complex strategies about how they approach sensitive data. Study's data show that users worry about security, however they experience some frustration or confusion with security practices and how to put them into practice. Another important aspect Gross and Rosson consider is the general inadequacy of *usability engineering* (UE) methods for addressing the problem of end-user security management. From the study emerged that "users are not interested in technical details, and when they are forced to understand these details to operate a computer properly, there are often breakdowns". When asked the question "Who is responsible for securing IT?", users identified three main categories: the **technical** perspective, obviously, is that security should be a concern of the IT staff of an institution. In the **organizational** perspective users believe that the organization they work for is responsible and should take care of security, with the organization putting effort in employing trustworthy, responsible and technologically knowledgeable personnel to respect organization's policies. In the **social** perspective both technical and organizational perspectives come together, while also including the end-user into the security process. Some

of the participants expressed their idea that IT staff, manufacturers and organizations are important but it is the end-user who is ultimately responsible. Participants who expressed a social perspective were more interested in security issues and more willing to take care of their own security.

### 2.1.2 "Security in the Wild: User Strategies for Managing Security As an Everyday, Practical Problem"

Dourish and Grinter [6] state that "effective security solutions depend not only on the mathematical and technical properties of those solutions, but also on people's ability to understand them and use them as part of their work". In their study they present different findings concerning the attitudes towards security, the expectations and practices of the users. From their data, unsurprisingly, neutral to negative attitudes dominated end-user experience, more precisely: **frustration** from those who were less exposed to computer systems and hence had less confidence in their abilities, **pragmatism** and **futility**. The study revealed that people tend to collect different categories of problems under the same umbrella (e.g. spam with viruses). They also imagine and seek unitary solutions to these problems that lead to three immediate implications: **(i)** a security solution that solves only one problem but not others could be considered as "partial solution" and therefore rejected, **(ii)** a technology deployed to solve one problem may be confused with a "complete shield", meaning that it can solve other problems than the one the technology is meant for, **(iii)** the focus on barrier or "choke points" distract from channels, meaning people may think that a protective system that operates on a network segment will work on the whole network (e.g. a new firewall installed but then running non encrypted WiFi communications). Similarly to what is found in [9], delegation is among the security practices, and in particular four forms were identified: delegate to **technology** (e.g. SSL encryption), delegate to **another individual** (e.g. the "technical friend"), delegate to an **organization** and delegate to an **institution** (e.g. financial institution). Some secure actions are discussed: **(i)** people using institutional means to secure communications (e.g. email from someone that has an attached signature file that states the legal and illegal uses of the contents of the message), **(ii)** people using context to secure (or obscure) their email messages, meaning that they use information that is known only by them (e.g. the color of the car they discussed together the previous day) **(iii)** other less technical solutions like recognizing that phone calls could be more secure (less traceable) than text messages or staff of an organization deciding to use a shared folder with access control instead of exchanging file via email.

### 2.1.3 "Mental Models of Privacy and Security"

Camp [4] has found five widely used conceptual models:

- **Physical Security Model**: Physical security processes are well understood and often brought to digital systems as well. When talking about computer security, concepts could change meaning, for example a computer within a room is physically protected, but if an ethernet cable reaches it, that security is undermined, hence the concept of security wall does not apply anymore in that context.
- **Medical Model**: The public health metaphor communicates important elements of network security. One of the concepts it expresses is that the person most harmed may not

be the one initially infected. It also communicates that each person is likely to be targeted. Some studies of network security have stressed the concept of the network as an ecosystem of security.

- **Criminal Model**: Users may better experience themself as potential vulnerable victim if computer security is linked to larger crimes. There is a tight connection between the crime model and the physical model. If crimes are modeled and presented as intrusions, people tend to better understand the risks but yet the criminal model may require actions for which end users have no tools.

- **Warfare Model**: The model implies the constant existence of an implacable enemy hence perimeter security and constant diligence are required.

- **Market Model**: Another way to see security and network vulnerabilities is as economic failures. Security failures on systems can harm other -better secured- systems. Three ways were identified: **(i)** *shared trust*, that is when a system trusts another one (e.g. when a password of a system is kept on another one), **(ii)** *increased resource*, that refers to the fact that "malicious player" (or attacker) could subvert multiple machines and leverage their processing power (e.g. brute force attacks, DDOS, etc.), implying that **(iii)** *traceability* is more difficult when subverting multiple machines.

### 2.1.4 "Folk Models of Home Computer Security"

Wash [18] identified eight *folk models* that can be broken down to two broad categories: viruses/spyware/adware and other form of malware and hacker and the threat of "breaking into" a computer. He explained how these models are used by users to decide what security software to use and which expert security advice to follow. Most participants made a distinction between "viruses" and "hackers", to them, these are two separate threats that can both cause problems. The eight model identified are the following:

#### Models for viruses

- **Bad**: Very under-developed model of viruses. Users in this category just know that viruses cause problems but couldn't really describe what kind. They seems to know that viruses can be transmitted. In general they heard some concepts somewhere.

- **Buggy Software**: In this model people think getting viruses is possible by just downloading and run untrusted applications/programs. The general belief is that viruses must be actively executed so antivirus programs may not be important. A virus is simply something that causes the same effects of buggy software such as deleting files occasionally or make the computer crash or reboot.

- **Mischief**: Almost same effects of buggy software, barely developed the idea of who created them. They are created just to annoy people.

- **Support Crime**: These people believe that viruses are created to stole information such as credit card information. These viruses could stay undetected. A way to prevent them is to keep the antivirus updated. They spread in different ways, from attachments in emails to automatic diffusion.

**Models of Hackers and Break-in**

- **Graffiti**: In this model people think of hackers as young technical geeks who are trying to impress friends or just cause mischief. It is not specified which kind of problem can be caused.
- **Burglar**: The idea behind this model is sort of incomplete since the identity of the hacker is not clear and the level of organization unspecified. The reason for break-ins is to look for financial and personal information with possible harmful effects on the computer.
- **Big Fish**: The model here evolves into thinking of the hacker as a professional criminal who is part of a criminal organization. The reasons for break-ins are the same for the burglar model, but the targets here are only "big fishs", without specifying who can be considered so. Effects do not include harm to the computer but only exposure of personal information.
- **Contractors**: The hacker is identified as a young technical geek but differently from the graffiti model, the hacker is a contractor for criminals whose job is to look for financial and personal information. Generally the target is a large database.

Wash spent few words talking about botnets, stating some facts that are true about many of the recent and large botnets:

1. Botnets attack third parties.
2. Botnets only need the Internet connection.
3. Botnets don't directly harm the host computer.
4. Botnets spread automatically through vulnerabilities.

He believes that botnet software takes advantage of the incomplete folk model of computer users to spread.

### 2.1.5 "Influencing Mental Models of Security: A Research Agenda"

Wash and Rader [19] extrapolated some lessons on mental models from Kempton's study "Two Theories of Home Heat Control*" [12]. Even if the example in the Kempton's Study are not strictly related to security, they help better understand mental models. **(i)** Users make everyday decision using simplified mental models. Even if these models do not represent a full understanding of home heating and energy used, they were used to make decisions. **(ii)** People prefer using simpler models that make decisions easier and lead to a sufficient outcome instead of more complex (and correct) models that are not worth it. **(iii)** Even if a model is more similar to the correct (or expert) one, doesn't necessarily mean that it leads to better decisions. **(iv)** It is not easy to find a mental model that is easy enough for people to use it and always leads to correct decisions. "Mental models are functional, rather than complete or accurate", Wash and Reader believe that people form security mental models from information they receive via stories told from other people, the media, interactions with experts and from their own experience. Since people usually don't have many examples around or security experts to learn from, actions are required to stimulate people share their experience.

### 2.1.6 "End User Security and Privacy Concerns with Smart Homes"

Zeng et al.[20] conducted a study to understand how people use their smart homes, their security/privacy related attitudes, expectations and actions. The study was done on 15 people, 12 of them were administrator of the systems and 3 guests (or incidental users, meaning that someone else is the administrator inside the house). They found four common smart home use cases:

1. **Increasing physical safety**: e.g. security systems, door locks and smoke detectors.
2. **Home automation**: e.g. automatically adjusting lights, temperature and other devices. This use case can be split into three different type:
   (a) **End-user programming**: allows users program automations for their home on a graphical interface, usually a mobile app.
   (b) **Custom scripting**: e.g. scripts for Raspberry Pi, either written from them or others.
   (c) **Third party apps**: e.g. IFTTT.
3. **Remote control**: e.g. light control, heating system control.
4. **In-home sensing**: e.g.camera feeds, air quality and status of devices.

It emerged that third-party automations are less common than custom programming solutions. Some of the users are reasonably more concerned about their home than their phone or laptop since, for example, they can be locked outside of the house while in case of theft of a personal device they could wipe the device remotely. Talking about **adversaries**, people were not able to identify a specific attacker. The most frequent identified adversaries were the companies that manufactured their smart home devices since they can have access to personal data. Not much was said about **vulnerabilities**, some of the users recognize them and some others have simple thoughts like "if someone can't get your WiFi password they can't watch your cameras". Some of the participants identified a *tradeoff*, accepting security risks in exchange for functionality and convenience of a smart home.

For what concerns **mitigation strategies**, two main category were identified:

1. **Technical mitigations**, examples:
   - Keeping smart home devices on a separate WiFi network from other home electronics, perhaps due to concerns about attacks by compromised smart home devices on other electronics.
   - Mitigate password and WiFi security related concerns with best practices.
   - HTTPS or more granular permissions on the sensors on device.
   - Deleting camera recordings or other logs of behavior to protect their privacy.
2. **Non-technical mitigations**, examples:
   - Altering one's behavior around smart devices (e.g. not talking about personal matters around speakers, or avoiding doing certain actions in front of cameras).
   - Place devices in a different location or turn them on only when not at home (e.g. cameras).

The principal attitudes toward security were discussed so far. It is possible to deduce that generally people, or more specifically technology users, tend to have negative feelings towards security, often trying to delegate decisions to others because of their low self esteem on the matter or difficulty to understand the problems and advice.

## 2.2 What Influences Security Behaviours

Before talking about how to intervene it is important to understand how security behaviours are formed and influenced. In the following section are discussed the main elements that influence security behaviours.

### 2.2.1 "Informal Support Networks: an investigation into Home Data Security Practices"

Nthala and Flechais [15] investigated which factors influence or affect security practices in home environments. In addition to other factors already reported by other studies (knowledge and skill, inconvenience, cost, trust, and influence) they identified three areas:

- **Survival/Outcome Bias**: The analysis revealed a tendency by participants to focus on practices that have overcome security breaches, and remember those who didn't. This is considered as one of the reasons for not implementing security measures.
- **Other Factors That Induce or Undermine Confidence in a Security Measure**: They found that, when a security measure is in place and the participants are confident in its effectiveness, the participants would trust the actions to be secure. Some of the participants presented examples related to internet banking (e.g. unrecognized payments) where the bank itself took care of the issue.
- **Availability and Quality of Security Support**: The focus was on understanding how participants choose where to seek support and/or whether or not to accept any unsolicited support that is offered. Five factors were found:
  - **Perceived Competence**: Participants reported making a comparison between their self efficacy (i.e. the effectiveness of their potential actions) and the perceived competence of a potential source. Of course, perceived competence can vary from person to person. For some of them competence is something achieved when working in data security (86% of participants). Others think that those who work for a technical company, regardless of whether their job is technical or not, are competent (24%). Another metric involves identifying someone with more experience in using technical devices (51%) or have studied in the field (27%).
  - **Trust**: As found in other studies (e.g. [6]), trust plays an important role when deciding where to seek security advice.
  - **Cost**: This splits in two parts: **(i)** cost to the one seeking help, which includes money, favours, and gifts, and **(ii)** the cost to the source of support in terms of effort and inconvenience.

  - **Closeness**: It was difficult to identify if the preference of the source of support is determined by (constant) availability of the source or how close one is to the source, hence closeness and availability are separated.
  - **Availability**: Independently of the type of relationship (e.g. friend-to-friend, parent-to-child, client-to-IT Service), 31% of the participants indicated that they consider availability as as a significant factor.

Participants were also asked to rank the above mentioned factors: **(1)** Competence, **(2)** Trust, **(3)** Availability and Cost to you (money, favours, gifts), **(4)** Closeness, **(5)** Cost to the source of advice/help (effort, inconvenience). They also found some characteristics of security support:

- **Duty of care:** Participants consider security support in the home as moral obligation, this duty is expressed through different modalities: **(i) Delegation**, as found by Dourish et al. [6], people delegate the responsibility for security to competent and trusted ones, **(ii) Motivation**, in the sense that users try to motivate others to behave securely, generally offering unsolicited support, **(iii) Social Responsibility**, meaning that people feel obligated to act for the benefit of society.
- **Continuity of care:** In case of delegation, people tend to seek a continuous caring relationship for two main reasons: first because they were helped the first time and so it is easier to come back to the same person (or entity) and ask again, and second because in case of problem resulting from someone else's support, it is easy to get back and seek further assistance.

### 2.2.2 "Influencing Mental Models of Security: A Research Agenda"

Wash and Rader [19] say that "mental models are functional, rather than complete or accurate [...] they are abstractions that contain inaccuracies when compared with the real world". Mental models evolve unconsciously over time due to new information and experiences. The implication here is that if you don't think about these new information and experiences or experience them directly, they won't be part of your mental model. It is possible to consider mental models as the *process of understanding* in the sense that people know how to react to certain situation based on their previous experience and are also able to generate new hypotheses or explanations. Wash and Rader say that "they [mental models] help us reason about cause-and-effect. They are not always good at allowing us to see effects that might be multiple steps away from the immediate ones", since mental models are not necessarily procedural. Mental model can be wrongly affected by different causes: information coming from past experience and new information coming from reading or hearing are organized based on pre-existing mental models. This means that they are hardly incorporated if they are contradicting the pre-existing mental models. Another cause that may wrongly affect mental models can be due to *coincidences*, which tend to reinforce existing mental models hence we end up in using these mental models because we are comfortable and familiar with.

Summing it up, the main factors that influence security behaviours are related to availability of support along with the way the supporting source is perceived by the user seeking help. The user also seeks continuity by the supporting source. Furthermore, past experiences and information heard from others can take part in the influencing process as well.

## 2.3 Security in IoT Devices

Security experts have raised concerns about security and privacy risks with IoT devices [2, 8, 17]. Concerns are due to bad design and lack of standards in security implementations. Some of the problems are related to pairing and discovery protocols that leak information, insecure communication and vulnerabilities in the devices themselves, that can allow unauthorized attackers to get access/control of the device.

### 2.3.1 "Privacy and Security in Internet of Things and Wearable Devices"

Arias et al. [2] discussed some common design practices and their implications on security and privacy. Throughout their investigation they found cases in which the manufacture over-rely on vendor designs because of lack of familiarity with the hardware. This approach could lead to exposure of interfaces meant for development, allowing attackers to leverage on them. Based on design requirements, availability of support, documentation and amount of security offered, manufacturers have to decide between **open- versus closed-source software**. With open-source code, a potential attacker has to find a vulnerability and exploit it, but the manufacturer doesn't have to rely on any vendor to patch the bug, allowing a faster response to threats. With closed-source code, a potential attacker has to reverse engineer the code to find a bug. Contrary to what happen for open-source code, the manufacturer has to rely on vendor response once vulnerabilities are found. Problems often arise in regard to **cryptography**, where vulnerabilities are found not just because of the mathematics involved, but because of implementation errors. An example made is uploading a malicious firmware to a device by spoofing an update distribution server. As mentioned above, sometimes **interfaces** meant for development are left open because of bad production flows. It is cheaper to buy programmable chips and then upload the software on it, rather than preprogrammed parts, furthermore, devices have to be tested before leaving production. Oftentimes testing is done through debugging interfaces that are not removed at the end of the process.

### 2.3.2 "Security, privacy and trust in Internet of Things: The road ahead"

Sicari et al. [17] analyzed the most relevant available solutions related to IoT security and identified open issues. They pointed out why traditional security measures, i.e. the ones used until today, can not be applied to modern IoT. There are three key security requirements that should always be present in every application: authentication, confidentiality and access control. Since IoT devices are constantly transferring data, it is important to secure communications. Furthermore, limited computing resource (e.g. memory, computational power), and ad hoc nature of networks in which these devices are placed, require to tailor existing techniques. From the study of Sicari et al. it easy to see that a unified vision regarding the key security requirements is still missing.

### 2.3.3 *Security Economics of the Internet of Things*

Schneier [16] points out the difference between security on smartphones and computers versus the ones on IoT devices. It is well known that large companies such as Microsoft or Google

have (and can afford) teams of experts that are in charge of security issues. The situation is different for some of the cheaper IoT devices which are often built by offshore third-parties. Another important difference is that these devices often do not have any way to allow firmware updates, meaning that even if the code of an exploit is made public there is no way to fix the device. This clearly is not a problem for smartphones and computers since they are constantly updated.

From what seen so far it is deductible that the biggest problem that comes when talking about IoT devices is whether a company can economically afford to sustain a proper production line and support its own product in terms of software update.

## 2.4 Proposed Solutions

Past studies tried to give solutions to the current problems related to security, both in general and specifically to IoT devices; in the following section a summary of the most relevant ones.

### 2.4.1 "Influencing Mental Models of Security: A Research Agenda"

Wash and Rader [19] classified solutions and suggestions studies gave, for a better approach to security, in three main categories:

- **Technical solution**: also known as "The stupid user approach", it leverages on the idea that most of the users have not the knowledge and skills to make security decisions hence it is better to remove these decisions from their hands by having default secure settings already selected or by making difficult for the user to act in an insecure way. This is not always achievable since a user may permorm insecure operations for different reasons, e.g. block updates to avoid breaking compatibility with an old software.

- **Education approach**: the point is to provide users with education so they can act securely, this approach is often used in organization environments to train employees. This approach works only when the effort to acquire a different behaviour is not too high, in fact users are not interested in deep technical details and often get frustrated because of these details.

- **Academics studies**: it is important to understand why users behave the way they do, and how support can be provided to them. A recent approach involves studying the behaviour of users and how to better design or create systems to help them understand security. It is not hard to believe that most of the security advice is provided by security experts that do not take into account the effort users have to put in understanding that advice.

### 2.4.2 "Informal Support Networks: an investigation into Home Data Security Practices"

Nthala and Flechais [15] proposed few recommendations based on their study:

- **Leverage existing social relationships**: current practices have focused on teaching end-users how to act securely. The suggestion is to leverage existing social networks, meaning helping people to be more connected between each other. As the study evidenced, this suggestion leverages two characteristics of support: duty of care and continuity of care.

- **Simple and useful tools**: it is important to develop new tools for non-expert users who, citing Dourish et al.[6], tend to seek unitary solution (i.e. a tool or advice that applies to every problem and situation), while currently every device and service has to be configured separately.

- **Evicende-based security**: the study has shown that people tend to be more motivated in changing their attitudes and their behaviour if they have evidence of harmful situations, hence finding and communicating attempts of attack could help users to understand and change perspective towards security.

### 2.4.3 "End User Security and Privacy Concerns with Smart Homes"

Zeng et al. [20] proposed recommendations on the design of smart home platforms and devices, and they also made some observations:

- **UI/UX for User Awareness and Control**: Some applications already have physical indicators, e.g. recording light on cameras. Future work should focus on providing users different ways of interacting with devices. Some of the participants to the study also mentioned that physical switches on devices should always be present to allow more control, even when these devices are not fully operable (e.g. no internet connectivity). This would also improve multi-user interaction.

- **Design Consciously for Multiple Users**: Since smart home devices are becoming popular, the need for multi-user support arises. In order to prevent power imbalance between primary users and incidental users (often less knowledgeable), future designs should consciously be created for multiple users. The problem is not just in the control of the device but also in the awareness. For example, as mentioned above, a recording light on cameras make people aware that they may be recorded.

- **Reputation Systems for Smart Home Options**: Differently from what happens for apps on online app marketplaces, electronic devices do not have a uniform centralized source of information where users can get feedback from other users.

- **Develop Standard Best Practices for End Users**: There are some best practices regarding security (e.g. how to choose a password for a WiFi network), however these practices do not cover all aspects of security. Future work could focus on the development and communication of such practices. An example could be to unplug or mute a recording device when not needed.

- **Design for Secure and Robust Interoperability**: One of the objectives of smart home appliances is to have different devices work in the same network and possibly "collaborate" between each other, hence the necessity to develop some interoperability interfaces. Often security breaches happen at the boundaries of systems, meaning that researchers and developer should better study these integration between devices (or services).

- **Minimize Tradeoffs for Security and Privacy**: Finding a trade-off between privacy and security is necessary. Zeng et al. challenged smart home devices designers to minimize this tradeoff.

Most of the solutions or suggestions given focus on the way security is designed by devices/services producers and exposed to users. It is clear that everything related to security is often cumbersome and difficult to comprehend by users because of the technical knowledge required to understand the risks, hence the necessity to revise the way the users are involved into security processes.

## 2.5 Human Computer Interaction

*Human Computer Interaction* is a multidisciplinary field of study that focuses on the design of computer technology and, in particular, the way humans interacts with computers. In the following section are presented some studies that show which are the most effective ways to design warning messages.

### 2.5.1 "A Communication–Human Information Processing (C–HIP) approach to warning effectiveness in the workplace"

Conzola and Wogalter [5] analyzed the C-HIP Model (Communication-Human Information Processing) which describes what are the steps for a warning message to be effective and influence the behavior of an individual. They focused more on workplaces' warning but the same concept can be applied to this work. Warnings are "vehicle for communicating important safety or safety-related information [...] that attempt to promote safe behaviour" and should posses four textual components: **(i)** a signal word (e.g. Danger, Warning or Caution) that attracts attention to the warning and give idea of seriousness of the hazard, **(ii)** a statement that briefly describe the nature of the hazard, **(iii)** a description of the possible consequences associated with noncompliance, **(iv)** instructions for how to avoid the hazard. Figure 2.1 shows the C-HIP model as a series of stages that must be completed successfully for compliance behaviour to occur:

- The first stage is the **source**, the "entity" with knowledge of the hazard and from which the warning is coming from. The source should expose some characteristics such as credibility, expertness and likability in order to maximize the chances of influencing behaviours.
- The source transmits the message through one or more **channels** to catch **attention** and to prevent the maintenance of attention on other environmental stimuli.
- Attention to the warning must be held long enough to proceed to the next stage: **comprehension**, where the warning receiver starts to comprehend the warning. Although the information is received, it may be not consistent with the receiver's preexisting **beliefs and attitudes** causing a bottleneck (i.e. preventing further steps in the model).
- Even if the warning messages pass through the Attitudes and Beliefs stage, the receiver must be **motivated** enough to engage in the directed action. When the **behaviour** is simple and easy to perform, the receiver could be more likely to adopt it.

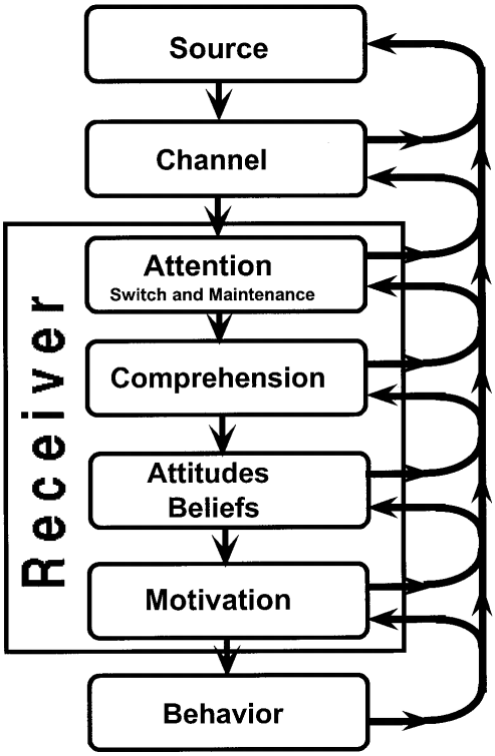



Figure 2.1: C-HIP Model [5]

### 2.5.2 "Harder to Ignore?"

Bravo-Lillo et al. [3] tested the effect of habituation[1] on six different attractors[2]:

- **Inhibitive attractor**: prevents users from performing potentially dangerous actions until a certain amount of time has passed or the user has performed a certain action (e.g. going to the bottom of a text area).
- **Animated connector + delay**: at the beginning the triggering option (button in their examples) is highlighted, then an animation moves the highlighted part toward the salient field.
- **Reveal attractor**: content is hidden at the beginning and progressively showed to be sure the user reads every letter.
- **Swipe attractor**: the trigger option (button) is disabled until the user swipes with his/her mouse over each letter of the salient field (from left to right).
- **Type attractor**: the user is requested to type the same text of the salient field, under the assertion that it would be difficult to type the text without paying attention to it.
- **Non-inhibitive ANSI attractor**: the salient field has a black background with a high-contrast yellow text to draw attention to it.

[1] Habituation: a form of non-associative learning in which an innate (non-reinforced) response to a stimulus decreases after repeated or prolonged presentations of that stimulus [10].

[2] Attractor: a modification in UI that tries to force attention on a salient field.

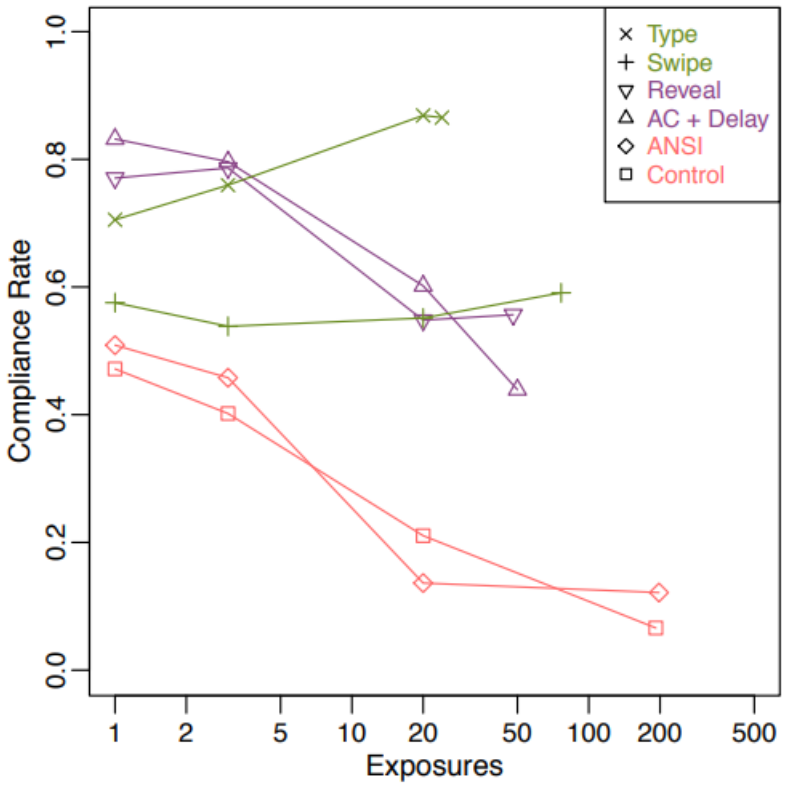



Figure 2.2: Percentage of participant who chose the "correct answer" to dialogs over many exposures. [3]

The testing procedures involved asking participants to perform a task in which they would respond to as many dialogs as possible for a fixed time period in a way that made possible to test habituation. Results are shown in Figure 2.2, where it is easy to see that *Type* and *Swipe* attractors are the most effective over multiple exposures. They tend to remain constant, while *Reveal* and *AC + Delay* are better at the beginning but degrade after some exposures. It is important to note that, even if the Type attractor was the one performing best, it also was the one who imposed the greatest usability burden.

### 2.5.3 "How Polymorphic Warnings Reduce Habituation in the Brain: Insights from an fMRI Study"

Anderson et al. [1] performed an investigation into security warnings and how habituation can be seen as one of the reasons why users ignore them. More specifically the objectives of their study were: **(i)** investigate how and where habituation develops in the brain in response to security warnings, and **(ii)** design a security warning "immune" to habituation. Among the contributions of this paper, the most important are:

- They demonstrated how *Repetition Suppression*[3] can be directly measured by fMRI[4], while previous study were measuring it indirectly.

[3]Repetition suppression (RS): a reduction of neural response that is often observed when stimuli are presented more than once.

[4]fMRI: Functional magnetic resonance imaging or functional MRI measures brain activity by detecting changes associated with blood flow [7].

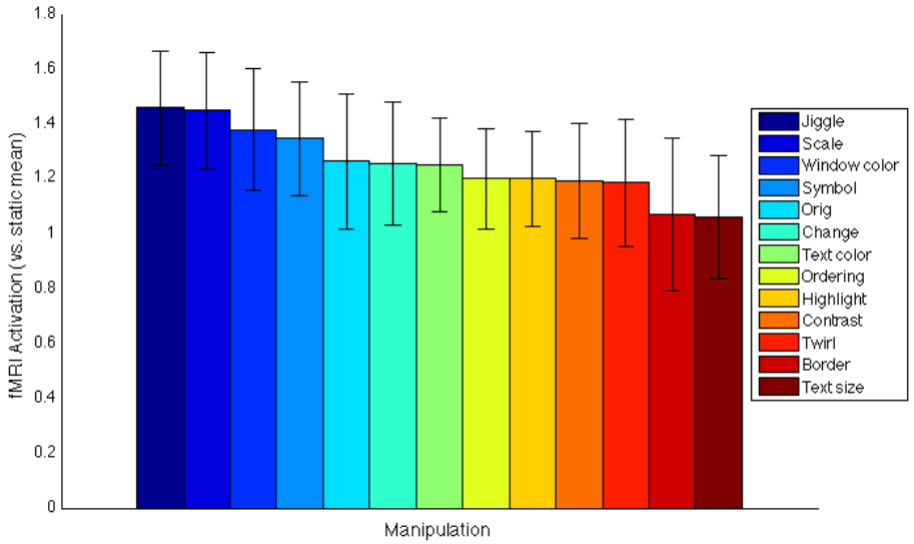



Figure 2.3: Polymorphic Warnings. [1]

- They demonstrated how **polymorphic** warnings are more resistant to Repetition Suppression than static warnings. They also derived and tested 12 polymorphic variations.

Polymorphic warnings garner more attention over time due to the novelty of their changing appearance. With this in mind they hypothesized that polymorphic warnings tend to have a different response, in specific brain regions, compared to static warnings. Results of their study are shown in Figure 2.3 where each column represent a different kind of polymorphic warning and the mean fMRI activation, i.e. neural activity.

As shown in this section, studies have already been done on which are or could be the best practices to implement warning messages. Two main categories were presented: the logic flow a warning message has to go through and the technical characteristics a warning message has to implement to be effective.

## 2.6 Concerns non-security related

Even if this thesis is focused on security related issues and how users interact with IoT devices, it is important to understand that other problems may arise when these devices are put in **multi-user environment**. Non-primary users may hold different concerns about security and privacy than the primary users.

### 2.6.1 "End User Security and Privacy Concerns with Smart Homes"

Zeng et al. [20] discussed three main differences between primary and incidental users:

- **Differences in Mental and Threat Models**: incidental users of smart home technologies have simpler mental models and less awareness about security and privacy issues than principal users. This seems reasonable since people who are habitual users of smart

devices tend to be more tech-savvy and curious about the technology, while other member of the house limit themselves to tolerate the primary user's "hobby".

- **Differences in Access**: incidental users do not always have full access to smart home devices. They are not interested in using them and setting them up for automation. Often they don't even have the proper apps installed on their phone and use just the "physical part" of the device, e.g. talking to a smart speaker.

- **Differences in Power and Control**: problems arise when principal users have, intentionally or unintentionally, more power and control over smart devices. Few examples were made in which is easy to see that devices can be used to track/spy on people and prevent them from performing actions (e.g. changing the temperature of the thermostat). Some of these scenarios could also be related to domestic abuse.

Among other considerations about non-security concerns they pointed out:

- **Reliability**: Participants want their devices and automation to work even during network or power failures. No one made a connection between reliability and security implications though. An example of one of these implications could be a burglar exploiting a power failure to access a door lock.

- **Interoperability**: Some of the participants expressed the need to have interoperability between their devices. They expect to control their devices through a centralized hub, smart speaker (e.g. Google Home or Alexa) or app. Some third parties, such as IFTTT and Stringify, entered the market with the idea to make as many as possible devices interoperable. This implies expaning attack surface and hence creating more security issues.

- **Cost**: Among the barriers identified by the users there is the cost of smart devices. This led some of the participants to develop their own solution instead of buying an existing one, increasing the risk of user error and potentially creating new security vulnerabilities.

### 2.6.2 "Privacy Expectations and Preferences in an IoT World"

Naeini et al. [14] studied what are the privacy expectations and preferences of IoT devices users. They also created a predictive model that allows, after observing decisions on three different scenarios, to predict preferences of users in other scenarios with an accuracy of 86%. One of their main results is that "privacy preferences are complex". Some result show that participants were in agreement on some preferences where social norms were in place that define what is acceptable and what is not. Some other results show that there are still environment in which it is needed to find a tradeoff between privacy and the possible advantages a certain technology provides (e.g. better heating system management, better security due to surveillance). The study also shows that there is need for technology to support **awareness of data collection**. People tend to be more comfortable sharing data in an anonymous way or more in general when data are not immediately identifying. There is a gap in understanding that, even if IoT collected data may be anonymous, it is not impossible to re-indentify the owner of that data. This should be kept in mind by vendors in their privacy information.

### 2.6.3 "User Perceptions of Smart Home IoT Privacy"

Zheng et al. [21] interviewed smart home owners, investigating the reasons why people buy IoT devices, what and how they feel about privacy risks and what they do in order to protect their privacy. From their study, three main themes emerged:

- **Convenience and Connectedness are Priorities**: The way in which smart devices make life "easier" and more convenient was mentioned as one of the main reason why people buy those devices. They help stay connected with the house, family and pets. It is easy to see that convenience outweighed participants' concerns about privacy and security issues.
- **Opinions about Data Access Depend on Perceived Benefit from External Entities**: Generally speaking, participants had different perceptions on home data collection depending on the entity collecting/accessing the data. The entities identified were **(i)** Manufacturers, **(ii)** Advertisers, **(iii)** ISPs and **(iv)** Government, with some overlap (e.g. an internet provider may advertise for its new streaming service, or Google may advertise for its thermostat).
  - *Manufacturers*: Participants were not significantly concerned about manufacturers accessing their data stating that "it is ok if it helps improving the devices and services", acknowledging that they will be the final beneficiary of this process. Some other participants would prefer the data to be anonymized to lower their concerns.
  - *Advertisers*: There is not a precise preference, half of the participants said they agree to share their home data with advertisers because they could benefit from targeted ads, while the other half disagreed. The question came down to whether they could benefit from it or not, but in general those who can control what data can be collected felt more comfortable sharing it.
  - *ISPs*: There is a general negativity when talking about ISPs. The bigger concern is due to the fact people think they cannot prevent their ISPs from seeing all their data. This practice is considered unnecessary since they would not have any profit from it.
  - *Government*: There is a general belief the Government would use data only for persecution, so the concerns mostly regard civil liberty. Some of the participants thought that local (e.g. better viability, smart grids) service could be improved if the data would be shared with the Government.
- **Trust in Manufacturer Privacy Protections**: One of the main factors influencing in participants privacy behaviour is the trust in companies. People are influenced by brand reputation when buying IoT devices and tend to prefer large technology companies that (could) have technical means to protect their data, even if they cannot verify that (e.g. if companies perform encryption or anonymization). Among the trusted companies there are those that produce home appliances and lighting brands, despite the fact that those companies have limited experience in the IoT field.

Summing it up, most of the problems non-security related fall under three categories: **(i)** multi-user interactions, in which it should be necessary to make all the users involved understand how to control smart devices, **(ii)** trust in companies that produce smart devices and may collect private data, and **(iii)** technical factors such as interoperability and reliability.

# Chapter 3

# Design and Implementation

This project will be part of a larger work which involves a "custom application layer" (see Figure 1.1). This custom application will be the part generating warnings, action or information requests and giving instructions. The system designed is thinkable as a *Message Broker*[1]: a software application that acts as intermediary between different parties. The job of the Message Broker (MB from now on) is to receive messages, called interaction requests, from the custom application layer and forward them to users' devices.

To better understand how a message broker works some people's profiles and real world scenarios will be presented in section 3.1. In section 3.2 are described different kind of interactions, that is the different ways in which the user is involved. In section 3.3 the Message Broker internal data representation is shown and finally in section 3.4 there is an overview of the prototype design.

## 3.1 Scenarios

Few scenarios are going to be presented to motivate the design. Considering Figure 3.1 as example network, all the examples consider a house with three members: two working adults and a teenager student.

The house consists of two bedrooms, a bathroom and a living room. The devices inside the home are 3 **smartphones**, a **smart speaker** inside the bathroom, a **smart thermostat** at the house entrance, a **router** and a **smart TV** in front of the couch in the living room and a **surveillance camera** outside the house.

In all the following scenarios, the application layer is a software capable of discovering anomalies on the local network.

[1] `https://en.wikipedia.org/wiki/Message_broker`

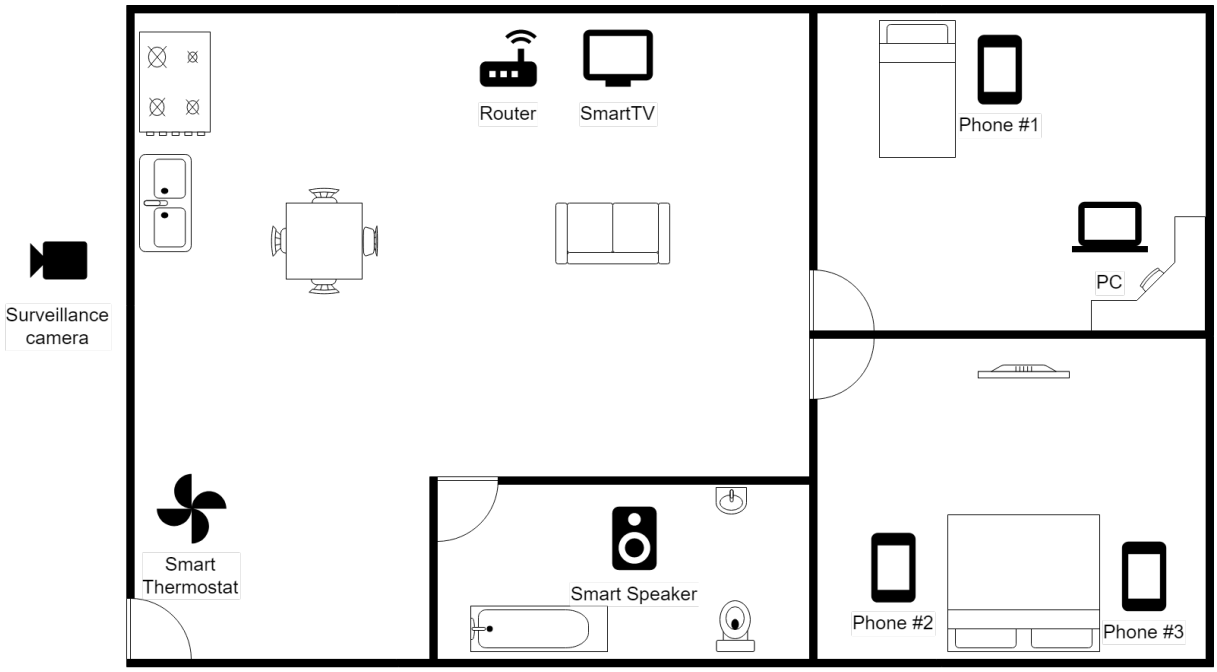



Figure 3.1: House network structure

## Profiles

**Person #1** - Carl (Father)
**Age**: 45
**Work**: Office job
**Experience**: Work email, office software, basic usage of web browsers.
**Devices used**: Smartphone #2, Smart Thermostat, Smart TV, Surveillance camera.
**Description**: Middle aged man who works 9-5 for a company that produces curtains. His job is to keep track of the deliveries with a management software for which he was trained. He also uses his smartphone for working purposes: emails and video conferences.
After some news came out with statistics about an increase of break-ins in the neighborhood he decided to install a *surveillance camera* on the outside. While looking for a camera he found a *smart thermostat* and decided to buy it, knowing that his son could install them without involving any technician.
During evenings he enjoys watching his favorite show on Netflix with his Smart TV.

**Person #2** - Joanne (Mother)
**Age**: 42
**Work**: Teacher
**Experience**: Basic usage of web browsers, social networks (Facebook, Twitter, YouTube), wearable device (bluetooth).
**Devices used**: Smartphone #3, Smart Speaker, Smart Thermostat, Smart TV.
**Description**: Middle aged teacher at high school who loves to cook and always seeks new recipes online to surprise her family. For her 42nd birthday she received a bluetooth smart band from her son, to keep track of physical activities and heartrate.
Another thing she really enjoys is taking long baths listening to her favorite songs through the *smart speaker* placed in the bathroom.

Table 3.1: Devices

| Icon | Device Name | Wired/WiFi | Data Usage[2] |
|---|---|---|---|
| | PC | Wired | 3-6 Mbps (Gaming/Streaming) |
| | Smartphone | WiFi | 3-5 Mbps (Social Media/Streaming HD) |
| | Router | * | |
| | Smart Speaker | WiFi | 0.25 Mbps (Audio streaming) |
| | Smart Thermostat | WiFi | 50 MB/month |
| | Smart TV | Wired | 25 Mbps (Streaming 4k) |
| | Surveillance camera | Wired | 60-140 GB/month |

**Person #3** - Robert (Son)
**Age**: 16
**Work**: High school student
**Experience**: Online gaming, social networks, good usage of web browsers, email, word editing and presentation software, very basic knowledge of networking
**Devices used**: PC, Smartphone #1, Router, Smart Speaker
**Description**: Teenager who loves hanging out with his friends on online gaming platforms, he uses his computer through the night.
As his mother does, he really enjoys listening to music while taking showers.
He is the most experienced technology user in the house, he knows how to access the router to open TCP/UDP ports, see devices connected to the network and install new devices.

## Devices

In Table 3.1 are listed all the devices inside the house.
For some devices such as PC, Smartphone, Smart Speaker and Smart TV makes sense to express the **Data Usage** in term of average bandwidth since they are constantly used. For others, Smart Thermostat and Surveillance camera it is better to express that quantity in terms of total data transferred in a month since they are not constantly using the network.

### 3.1.1 Scenario #1

Carl and Joanne are outside and Robert decides to have a friend over. As soon as the friend arrives, he asks for the WiFi password to connect his laptop and once the access to the network is granted a notification message is issued to Carl's phone saying that a new device has been connected to the home network.

The message broker decided to send the informational message to the father under the form of

[2]Source: "Does Internet of Things Disrupt Residential Bandwidth Consumption?" [13], [11]

a push notification for two reasons: **(i)** Carl is the one in charge of the security of the house (network administrator) and **(ii)** he is not at home so the phone was the only way to warn him.

Since the new device is a simple computer (legitimate client) no actions are required and Carl can return to what he was doing. The network continues to be monitored and in case of suspicious behaviours other interactions may happen.

### 3.1.2 Scenario #2

Someone told Robert about a new free game he should try, so as soon as he was able to play with his computer he went straight to the website to download it. The browser showed a warning message but since his friends told him that website was safe he decided to ignore the message and proceed to the website, downloaded the executable and ran it. Similarly with Windows, a prompt appeared saying that the operation (of executing the application) could have been harmful but again he decided to ignore it. Luckily for him the game started so he thought that was not a virus after all.

A few days later he noticed that the internet speed on his computer was slower than the usual. He tried to look at the task manager to see if there were strange processes or unusual use of the network when the computer was idle, but he found nothing. Actually the game had the proxy settings changed on the computer because of some internal optimization. The traffic was therefore redirected.

The following day while Carl was watching his TV show, a notification appeared saying some actions were required, so he expanded the notification and an explanation appeared, saying that traffic from and to the computer was redirected somewhere outside the country (proxy) and for that settings must be changed.

Something that can be seen as a problem is the case in which the TV is watched by someone who is not the network administrator. This won't be a real problem since the action is not requested do be executed immediately and in case it is ignored the message broker would send another message to another device.

### 3.1.3 Scenario #3

Holidays have finally come, that means staying up until late at night. Robert didn't hesitate to take advantage of the situation and decided to have a long night playing online with his friends. Everything sounds great if not that this time, while playing he used a new software to chat with his friends.

From the application layer a warning arose since unusual and unrecognized traffic was detected, the warning was sent by the message broker to the network administrator (Carl) to warn him and ask if the traffic was legit. Since Carl knew his son was playing on his computer he just dismissed the notification.

## 3.2 Interaction types

As seen in the scenarios, there could be different interaction types depending on the "amount of involvement" required by the user.

- **Notification without requiring action:** this one is the simplest and the one requiring the least effort on the part of the user. Usually it refers to a warning that can be ignored (see Scenario #1). Examples: new legitimate device connected to the network, connection from outside denied, automatic change of configuration.

- **Notification with action required:** this applies when it is necessary to perform an action that could be either physical or virtual (see Scenario #2). Examples: reboot router or PC, change device configuration, change WiFi password.

- **Request for information:** this is the case in which information is required to better understand the status of the network and decide if further measures have to be taken (see Scenario #3). Examples: if a new unknown device is connected to the network then ask for the device type, if suspicious activity is detected then ask to confirm.

- **Instructions:** this is a sub kind of interaction type that can be included to others. For example, when notifying required intervention, a list of action could be provided to better understand the the actions the user has to perform.

The kind of interaction is decided implicitly by the *application layer* while the *message broker* is in charge of deciding which is the best way to involve the user, i.e. through which device.

## 3.3 Data format

There are two main "objects" that need to be stored inside the Message Broker system, they are used to represent all the data needed for the system to work. The first one regards the format of the **interaction** requests, the second one represent the **devices**. Both are JSON objects and are the same format used for exchanges between all the parties (Custom Software Layer, Message Broker and Devices).

#### Interaction & Sub-objects:

The object used to represent the requests is the *Interaction object*, it is exchanged between the custom application layer and the Message Broker and between the Message broker and the devices. Table 3.2 shows the data structure of it. Some sub-objects were defined to better represent the data. The Interaction object holds both the request and the answer, which could have been treated as two different objects but instead it was decided to leverage on the flexibility of JSON (and in future of MongoDB) to avoid joining operation between different data structures (or documents).

Notes on the structure:

- There are two fields that refer to IDs, **_id** is the ID automatically generated by a third part library (Mongoose) discussed later, while **us_id** stands for unique sequential id, which is an ID generated by the Message Broker to uniquely identify Interactions and the order in which they arrive.
- **status** represent the current status of the notification. Its value will be 1 in case of a pending notification, 2 for a completed notification and 3 for a canceled notification.
- **title** and **description** are the two main elements, always present, which represent the title of the notification and the main text respectively.
- **InteractionInput** object represent the inputs requested, if any. The **type** can be "text", "textarea", "checkbox", "select" and "button".
- The **InteractionResponse** will hold all the values for each input requested. The field *name* will have the same value of the *name* in the InteractionInput.

For a complete description of each field please refer to Appendix B.

Table 3.2: Interaction object

| **Interaction** | |
|---|---|
| **Key** | **Value Type** |
| _id | String |
| us_id | String |
| timestamp | Date |
| status | Number |
| title | String |
| description | String |
| specific | String |
| instructions | String[] |
| level | Number |
| inputs | InteractionInput[] |
| image | Image |
| response | InteractionResponse[] |

| **InteractionInput** | |
|---|---|
| **Key** | **Value Type** |
| title | String |
| name | String |
| type | String |
| elements | [{text: String, value: String}] |
| required | Boolean |

| **InteractionResponse** | |
|---|---|
| **Key** | **Value Type** |
| name | String |
| value | String |

**Device & Sub-objects:**

Table 3.3 shows the data structure of the Device object.

Few notes on the structure:

- The **_id** is generated by a third part library (Mongoose) discussed later, while the **deviceID** is an optional field generated by the device and used by the driver.
- **Capabilities** is a sub-object containing the input and output capabilities of a specific device. The input capabilities are "text", "microphone", "camera". The output capabilities are "text", "video", "audio", "vibration", "notification".
- **driverID** represent the driver with which the specific device has to be handled.

For a complete description of each field please refer to Appendix B.

Table 3.3: Device object

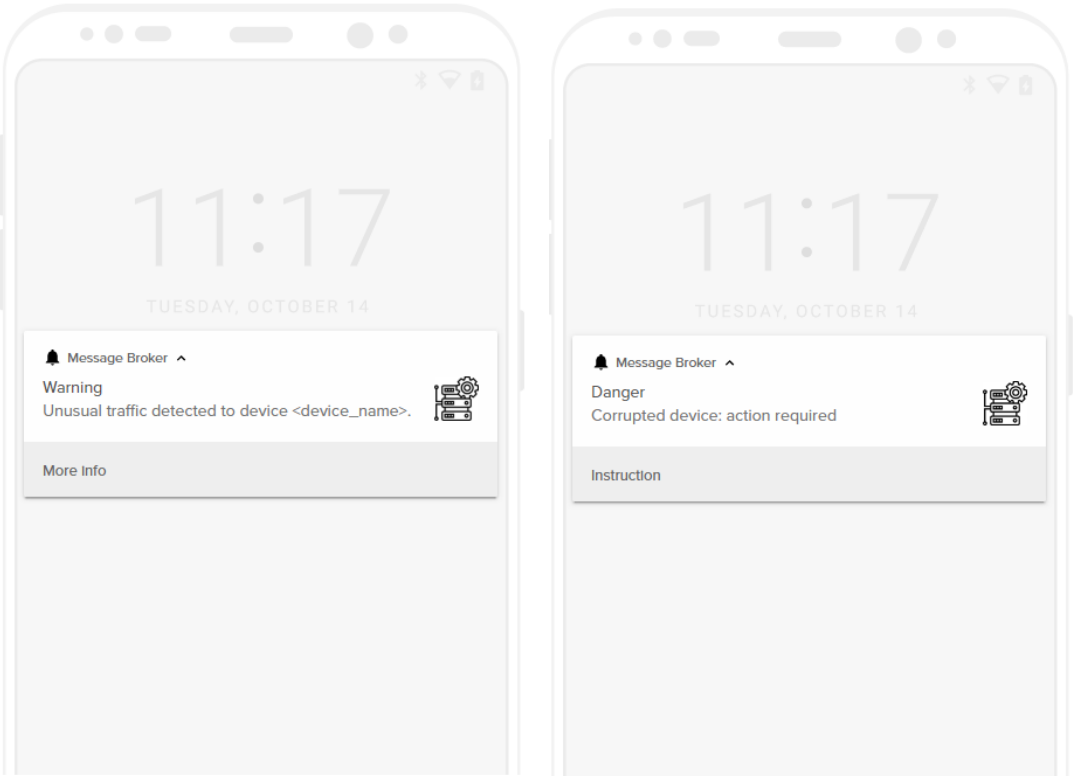
| **Device** | |
|---|---|
| **Key** | **Value Type** |
| _id | String |
| deviceID | String |
| name | String |
| description | String |
| capabilities | Capabilities |
| lastActive | Date |
| address | String |
| driverID | String |
| online | Boolean |

| **Capabilities** | |
|---|---|
| **Key** | **Value Type** |
| in | String[] |
| out | String[] |

### 3.3.1 "Output" format

**Smartphone:** Figure 3.2a represent a notification sent to a smartphone. No action is required from the user. More information about the issue and how it was resolved could be found clicking on "More Info".

Another example in Figure 3.2b shows a *danger message* in which the "Instruction" button will lead to the action the user need to take in order to solve the problem.




(a) Warning message (b) Danger message

Figure 3.2: Push notifications on Smartphones

**Smart TV:** Figure 3.3a represent how an action requested is showed on a Smart TV in addition with information request (fillable form).
Figure 3.3b shows a simple warning message with the possibility to have more information through a "Reveal attractor"[3] [3].

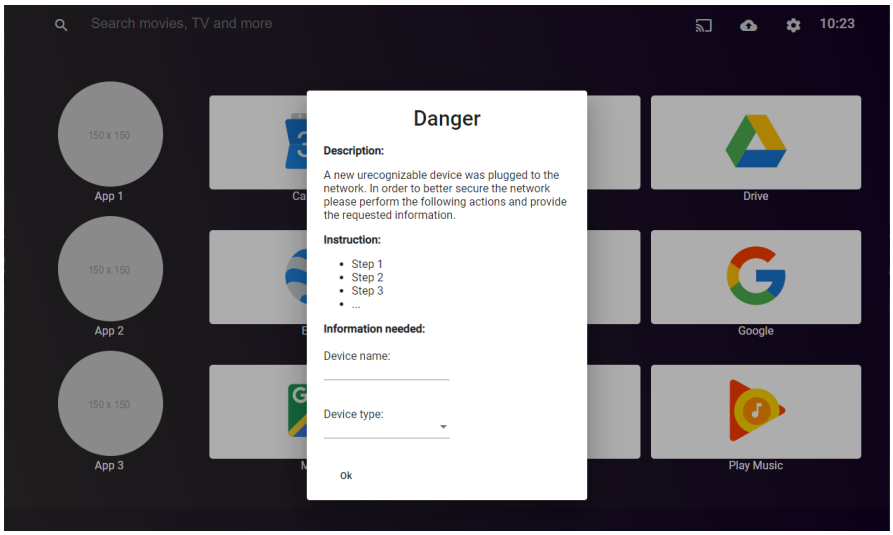
**Speaker:** Clearly, it is not possible to have a mockup for this kind of device. The idea is to warn the user with a simple message and possibly redirect him/her to a different device.
An example of a message could be: "Danger, an action is request to protect the network. Please open the Message Broker app.", another example could be "Warning, unusual traffic was detected and blocked on the network. More info available on the Message Broker app.".

## 3.4 Prototype implementation

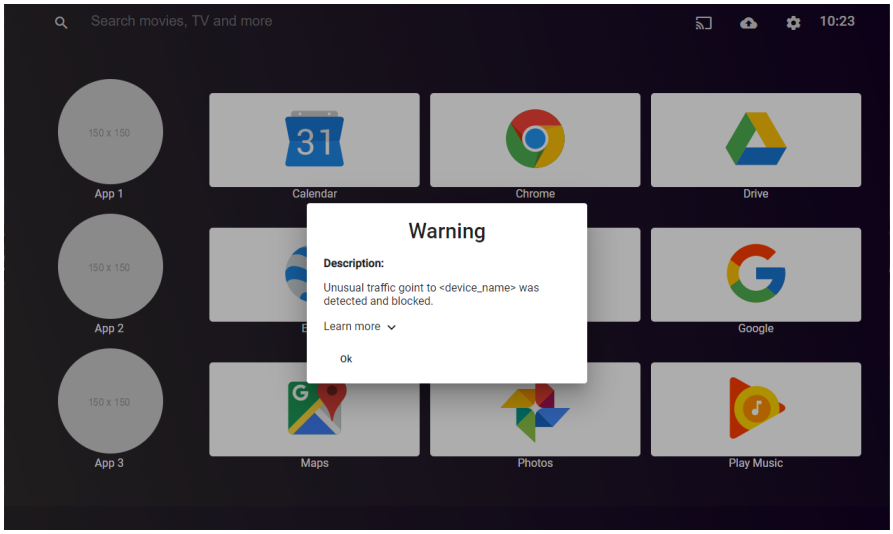
In this section an overview of the prototype implementation is presented: **(i)** how devices join the Message Broker network; **(ii)** how requests from the Application Layer are handled; **(iii)** overview of the web API and **(iv)** overview of the devices's app.

Before starting explaining the design of the system, it is important to keep in mind that, referring to Figure 3.4, the focus of the project is on the Message Broker and the way it uses the devices to interact with the user. Every time there is a reference to a "new device joining the Message

[3]Text hidden at the beginning and then showed through an animation.




(a) Danger message




(b) Warning message

Figure 3.3: Notifications on Smart TV

Broker network", we signify that the device is meant to interact with the user on behalf of the Message Broker. The development was not limited to the MB system, for testing purposes three devices were chosen and different apps were made to make them communicate with the MB network.

First of all a distinction between the Message Broker intended as core system and the Message Broker Administration Panel has to be made. Implementation of devices will be discussed later in Paragraph 3.4.4. The development of the Message Broker (both front-end and back-end) is based on part of the *MEAN Stack*[4]. The choice was made based on the flexibility provided by the stack, both in scalability and simplicity to develop a functional web interface without too much experience in web design. The MEAN Stack includes the following technologies:

- **M**ongoDB : Document database - used by the application to store its data as JSON (JavaScript Object Notation) documents. Even though MongoDB would be useful, it

[4] `http://mean.io/`

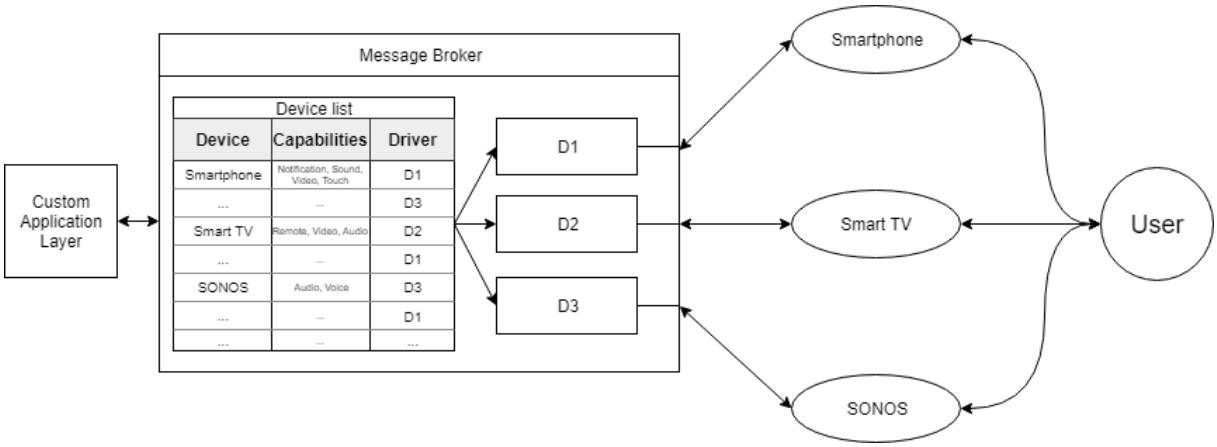



Figure 3.4: Message Broker Concept

was decided not to deploy it to speed up the development. Simple arrays were used to store the data on the RAM memory of the process.

- **E**xpress (sometimes referred to as Express.js): Back-end web application framework running on top of Node.js, used to develop web API.
- **A**ngular: Front-end web app framework; runs JavaScript code in the user's browser, allowing application UI to be dynamic.
- **N**ode.js : JavaScript runtime environment – let use JavaScript as programming language for server applications.

### 3.4.1 Message Broker Core System (back-end)

Figure 3.4 provides a conceptual representation on how the back-end works. Starting from the left of the image, we see a *Custom Application Layer* which interacts with the MB through web APIs. The MB holds a table of which devices are or were connected to the network and currently available to being used as interfaces to and from the user. Since the MB needs to decide which device has to be used for each request from the Application Layer, it also stores the capabilities for each one of them. For the back-end part **Express** was used to implement RESTful API, on top of **Node.js** coupled with other libraries: **Mongoose** to manage the data models, **Socket.io** to enable real time messaging between back-end and front-end via web sockets. In the following we review the components of the Message Broker back-end:

#### Drivers

To allow better customizability and scalability within the project, it was decided to add another level of abstraction between the MB and the devices. Every device is coupled with a **driver** who implements its own logic to communicate with the device. In this way even non-standard devices or custom IoT applications can be used as devices in the MB network. This further level of abstraction allows expert users (or companies) to implement their ad hoc interaction flow. Example: to allow a phone to be reached even if when not connected to the home WiFi network, a custom driver can be created to establish a VPN to the phone. Every driver has to

implement different methods to enable a device to be part of the network, the methods are the following:

- *Search/Scan for device* - This method is used when a user wants to add a new device to the Message Broker network. It returns a promise[5] containing a list of devices discovered.
- *Check availability* - Before sending an interaction it is necessary to check devices availability, this method receives the *device* object to test connectivity on and returns a promise containing an object with the id of the device and its online status.
- *Send interaction* - It receives the *interaction* and the *device* objects and perform the operation to send the interaction to the corresponding device.

It is important to understand that the development of the Message Broker is completely decoupled from the development of the apps for the devices used to test the Message Broker itself.

### Message Broker API

Express was used with its **router** module that allows to better organize the routes (i.e API endpoints). The two main routes are:

- */api/interactions* - Used by the application layer and from the front end part of the Message Broker. Important to notice that we are talking about REST APIs, so every method (GET/POST/PUT/DELETE) is exposed. This surely is not good practice, it should require some form of authentication, but since this is just a prototype authentication and authorization were not included in the development.
- */api/devices* - Used just by the front end part of the Message Broker. Its peculiarity is that it is not completely REST since it has a sub-route which represent an action, in fact *GET /api/devices/scan* and *POST /api/devices/scan* are used to perform a scan of the network and add a device previously found.

As said, both routes follow the REST model which means that calling them with the *GET* method returns the collection and appending the *id* of the object at the end of the route gives the single object, example: *GET /api/devices/5d35dbc4f775811744653cd4* will return the object representing the device with id *5d35dbc4f775811744653cd4*.

For a complete overview of the API, please refer to Appendix **??**.

### Front-end interactivity

In order to make the Administration Panel more interactive, it was decided to use Socket.io, a library that enables real-time, bidirectional and event-based communication between the browser

[5]The Promise object represents the eventual completion (or failure) of an asynchronous operation, and its resulting value. Source: `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise`

and the server through web sockets. Its usefulness lays in the fact that the client (which often is a web-app running on a browser) does not have to perform polling, but whenever there is new data the server can start the communication.

Even though Socket.io could work in a bidirectional way, it is used just to communicate changes of states of devices or interactions requests from the server to the admin panel. As said, the same updates provided by Socket.io could be obtained with polling requests to the APIs provided by Express. The two main events are:

- *newInteractions* - Used to push all the interactions to the client.
- *newDevices* - Used to push all the devices to the client.

In our case, the client is the administration panel. Of course it is not a good practice to push all the data at once; this real time kind of updates would be more efficient if just the updated part of an object is pushed. Such optimizations are left as future work.

#### Data management

MongoDB[6] is a document database, it provides the flexibility needed to store data without a predefined schema. This is useful both for representing devices and interactions: devices may have different characteristics, requiring to store different kind of information (e.g. storing screen size is not suitable for a smart speaker), interactions could be of different type, some of them may require input hence the necessity to store the form structure.

The two main collections are:

- *Interactions* - To store interactions information.
- *Devices* - To store devices information.

Collections, data models and queries are handled by Mongoose[7], which provides a straightforward, schema-based solution to model application data. It includes built-in type casting, validation, query building, business logic hooks and more.

During the development it was realized that MongoDB is not strictly necessary to demonstrate prototype functionality, for this reason it is not implemented in the prototype. In further development will be easy to integrate MongoDB since Mongoose is already handling the data model and it just requires a connection to the DB to effectively store the data. Until that point data is locally stored on the RAM that the Message Broker process uses and erased when the process stops.

[6] `https://www.mongodb.com/`
[7] `https://mongoosejs.com/`

#### Discovery protocol

Since some devices are unable to execute code (e.g. SONOS Speaker), the initial idea was to have two ways in which a device could join the Message Broker network: actively by having the device perform some discovery protocol or passively by letting the Message Broker discover the device. This idea was abandoned after the introduction of *Drivers* that allows the Message Broker to delegate the process of finding new devices or be found by new devices to a custom piece of code. The discovery protocols implemented by the drivers will be discussed later in Paragraph 3.4.4. The implication here is that, in this way, the user has to "manually" add the device to the Message Broker network. "Manually" means that the user needs to click on a button on the Administration Panel. Another possibility would have been to perform some sort of polling on the devices in the network and automatically add them but I preferred to make every step clear.

### 3.4.2 Message Broker Admin Panel (front-end)

As already mentioned, Angular[8] was used to develop a Single Page Application both for flexibility and simplicity of development. It was decided to adopt this framework and develop a web interface because of its simplicity in moving all the application to a cloud based system and the ease of connecting it to a Node back-end.

#### Angular

Angular is a web framework used to develop multi-platform *single page applications* using MVC architecture that leverages on modern coding patters such as declarative templates and dependency injection. In addition, for what concerns the user interface, the prototype uses **Angular Material**[9] which is a UI library that provides pre-defined fully customizable components in line with modern design principles (Google's Material Design).

Another important feature about Angular is that it uses **Web Components**[10]: "a set of web platform APIs that allow you to create new custom, reusable, encapsulated HTML tags to use in web pages and web apps. Custom components and widgets built on the Web Component standards, will work across modern browsers, and can be used with any JavaScript library or framework that works with HTML". Web components are based on existing web standards. Features to support web components are currently being added to the HTML and DOM specs, letting web developers easily extend HTML with new elements with encapsulated styling and custom behavior.

#### Main components

To show the current status of the system and test it, three main components were developed:

[8] `https://angular.io/`
[9] `https://material.angular.io/`
[10] `https://www.webcomponents.org/introduction`

- **Connected devices** (Fig 3.5a): it shows all the devices that were and are connected to the network. Each device is displayed inside an *Expansion Panel*[11] that contains all the information about the specific device (e.g. last time connected, IP address of the device, current status). All the data is shown in real time.

  Next to each device, even when the expansion panel is closed, there is a dot which represents the current status of the device: the dot will be green in case of device currently connected, or gray if not currently connected (i.e. reachable).

  Through this component, it is also possible to add a new device which means asking the back-end part of the MB to scan the network for new devices. After the scan, a modal[12] is showed with the list of the addable devices.

- **Latest interactions** (Fig 3.5b): a list of the latest interaction requests is available to be explored. Each interaction request is wrapped inside an expansion panel that will show all the information about it once expanded.

  Next to each interaction request, there is a dot which represents the status of the request: the dot will be green when the interaction is completed, yellow when it is pending and red when there was some error.

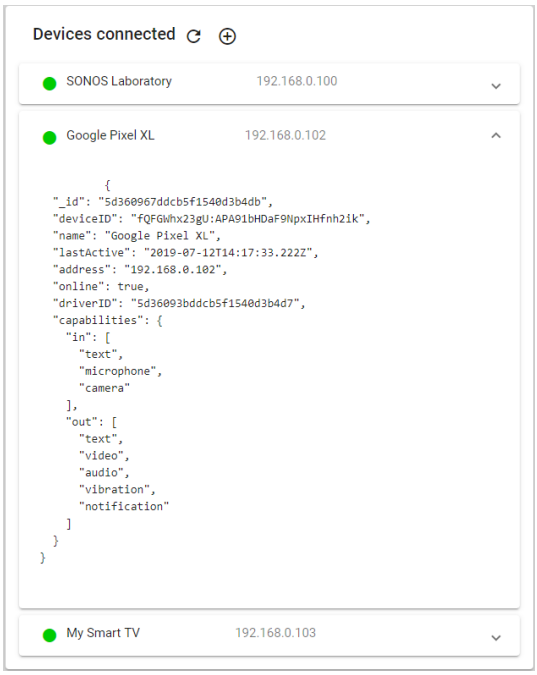
- **Test area** (Fig 3.5c): this component was designed for the mere purpose of testing the whole system.
  Through it is possible to generate an interaction request via a graphic interface (instead of programmatically using the APIs), specifying all the field: title, message, etc... It is also possible to select a specific device for which is needed to test the functionalities; if no device is selected the message broker will decide itself based on the capabilities of the devices available: if an interaction requires some input from the user, it will send the request to a device capable of handling input.

### Service & API

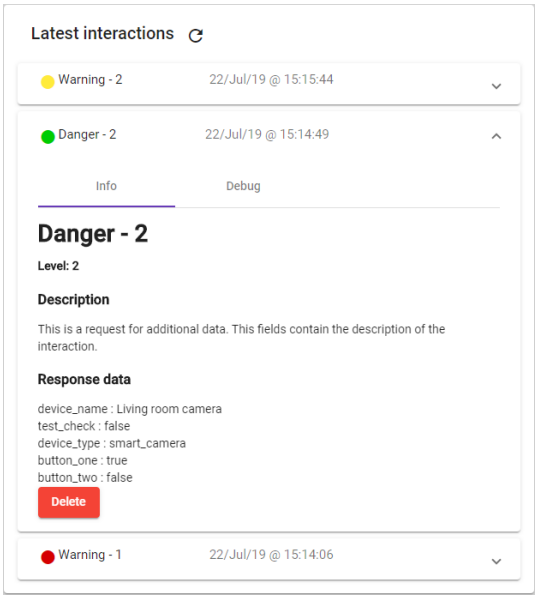
Angular **services** are singleton objects which get instantiated only once during the lifetime of an application. They contain methods that maintain data throughout the life of an application, i.e. data does not get refreshed and is available all the time. The main objective of a service is to organize and share business logic, models, or data and functions with different components of an Angular application[13].

Inside the front-end part of the Message Broker there is just one service, this service is in charge of fetching the data from the back-end. At the instantiation of the service object the service starts fetching the data from the back-end and then registers two handler on socket's events. Both handlers propagate the new data from the back-end to the components. The propagation is

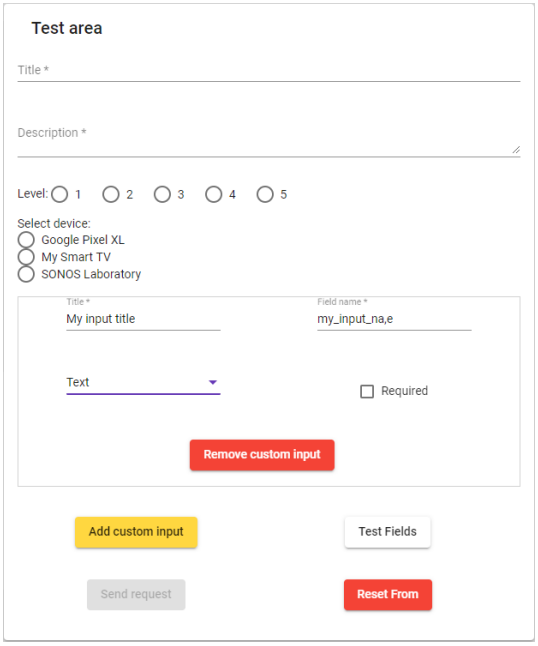
[11] Expansion Panel: one of the Angular Material component that allows to show and hide some content.
[12] Modal: similar to a popup but intended for the user to perform some action on it.
[13] Source: `https://dzone.com/articles/what-is-a-service-in-angular-js-why-to-use-it`




(a) Connected devices




(b) Latest interactions




(c) Test Area

Figure 3.5: Admin Panel Front End

done through two **Subject** Objects[14] to which each component (Connected devices and Latest interactions) subscribe at their creation.

Method available:

- **getDevices()** and **getInteractions()** - Return the Subject object for devices and interac-

[14]**Subject**: a special type of Observable that allows values to be multicasted to many Observers. Source: `https://rxjs-dev.firebaseapp.com/guide/subject`

tions respectively.

- **postDevice(device: Device)** - Used to add a new custom device to the network. This method is not currently used, left here for future development.
- **scanDevices()** and **addDevice(id)** - The first one is used to perform a scan for new devices on the network, while the second one is used to add one of the devices found, by id.
- **postInteraction(interaction: Interaction)** - Receives an Interaction object and send it to the back-end to create a new interaction request.
- **deleteInteraction(interaction: Interaction)** - Receives an Interaction object from which the *id* is extrapolated. It performs a delete operation to remove the requested interaction.

All of the above methods, except for the getters, return an observable[15] of the http request.

### 3.4.3 Generation of request from Application Layer

Figure 3.6 shows the process in which a generic Application Layer can raise a request and receive a response. In this example it is taken for granted that the Application Layer already knows how to contact the Message Broker (API endpoints). Step by step:

1. The Application Layer generates the request.
2. The request is sent to the Message Broker trough the correct API endpoint.
3. The Message Broker decides through which device it wants to send the request to the user.
4. The device availability is tested.
5. Availability is confirmed (or timeout if not confirmed).
6. Request is sent to the Device.
7. The app on the Device handles the request and interact with the user (if needed).
8. Response is sent from the Device to the Message Broker (if any).

A clarification on point #3: the decision is based on the capabilities of the devices currently registered as interfaces. If the interaction request requires inputs it is sent to the first registered devices that has input capabilities, otherwise it is sent to the first device with just output capabilities.

The method used to notify the Application Layer about the fulfillment of the interactions' responses is through polling. When creating a request for the Interaction (in the API call) the object representing it is returned. The id of the interaction is available to the Application Layer to poll the MB looking for changes in the object, remembering that, as stated in Section 3.3, the

[15]Observables provide support for passing messages between publishers and subscribers. Source: `https://angular.io/guide/observables`

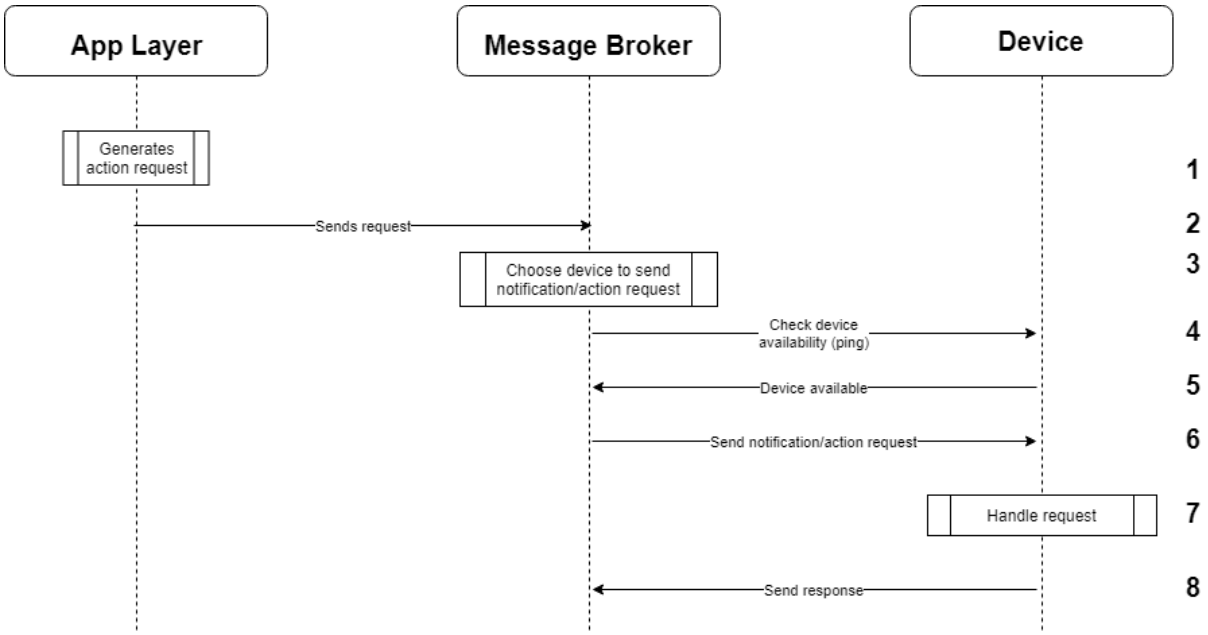



Figure 3.6: Request flow

response to an interaction is stored within the interaction object itself.
Another approach could have been to use web-hooks but that would have been based on the assumption that the Application Layer would implement a web server, which complicated the design.

### 3.4.4 Clients (back-ends)

This sub-section presents an overview of the backends implemented. Recall that, in order to test the Message Broker functionalities, it was decided to pick three devices with different capabilities.

#### Smartphone

The smartphone used was a *Google Pixel XL*. An Android app was developed to enable the device to be discoverable and receive interaction requests. The main components are:

- **Main Activity**: The entry point of the app when clicking on the icon. Not much work is done here since the main purpose of the app is to let the device receive notifications, hence there is no significance to opening the app. The main activity is used just for testing purposes, showing some debug information and a simple form to locally test push notification.
- **Discoverable Service**: A background service that allows the device to be discoverable. It implements a simple discovery protocol which will be discussed later. Since this service uses the network it declares some permissions in the *Manifest*. The permissions are: *INTERNET*, *ACCESS_NETWORK_STATE* and *ACCESS_WIFI_STATE*.
- **Notification Activity**: This may be considered the real main activity because it is the one used when a notification arrives (under the form of a push notification) and the user

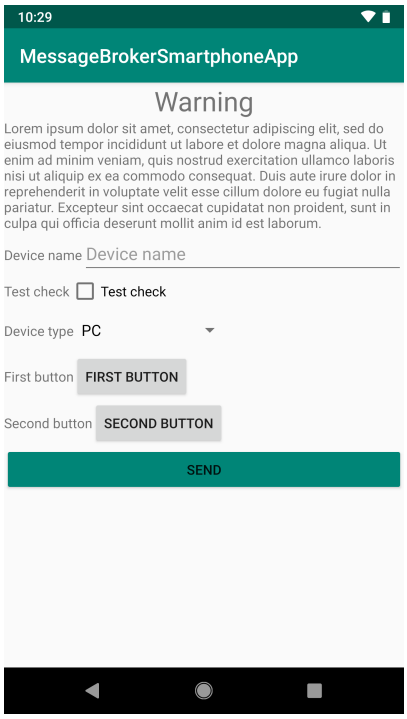



Figure 3.7: Notification Activity

clicks on it. After clicking on the notification the notification activity is opened. Its job is to render the notification request (Fig 3.7) and send back the user answer (if any) to the Message Broker.

- **Notification Service**: A background service that extends *FirebaseMessagingService*. It is used to receive the data through the *Firebase Cloud Messaging Service*[16] and create the push notification.

Summarizing: the Message Broker and the Android app exchange data in three different ways. For the discovery protocol they exchange UDP messages within the same network; The Message Broker uses the Firebase Cloud Messaging Service to send data to the Smartphone app, which is through a cloud service; The Smartphone App uses the HTTP REST API of the Message Broker to send back to it data inserted from the user (if any). During testing the HTTP APIs were used inside the same network but it could be possible to enable them to run over the internet.

### Smart TV

A Raspberry Pi was used to run an Angular web app to emulate a Smart TV. The app consists of two parts, the front-end, that is an Angular Single Page Application and a back-end, that is a nodeJS server. The front-end and the back-end communicate through web-sockets (with Socket.io). The interesting part is how the Message Broker communicates with the Smart TV App: for the discovery protocol they exchange UDP messages on the same network; from the Message Broker to the Smart TV messages are exchanged through HTTP API exposed by the TV (on port 80); from the Smart TV to the Message Broker (for user replies) the REST API of

[16] `https://firebase.google.com/docs/cloud-messaging`

the MB are used (port 3018).

Figure 3.3 shows examples of notification on the Smart TV app. The main components are:

- **NodeJS server**: It has four tasks.
  1. Enable the device to be discoverable implementing a custom discovery protocol that will be discussed later.
  2. Expose HTTP API to allow the Message Broker to send messages to it.
  3. Handle the replies of the user from the front-end and forward them to the Message Broker.
  4. Communicate with the front-end part of the Smart TV through web-sockets with Socket.io.

  Like the Message Broker itself, the Smart TV App uses Express as framework to expose its single HTTP endpoint which handles the interactions sent by the Message Broker.
- **Socket service**: Angular service in charge of handling the messages coming from and to the back-end through the socket. Its own socket endpoint event for the socket is *'interaction'* which receive the interaction object and propagate to the main component through a RxJs Subject[17].
- **Main component**: Called App Component, is is in charge of generating the main interface of the application and showing the dialog component when a new interaction arrives through the RxJs Subject to which it subscribed at the application start.
- **Dialog component**: It is an Angular Material Dialog[18] that is in charge of rendering the interaction request.

### SONOS Speaker

The SONOS Speaker is a peculiar device since it can't run a custom app, but luckily it exposes some API for interaction. To further facilitate the development it was decided to use a node package that wraps the SONOS API; the name of the package is *node-sonos*[19]. Since the objective of using a speaker is to reproduce warning messages it was necessary to implement various features in the SONOS's driver. These features are a voice synthesizer and a simple HTTP server whose only tak is to share an audio file with the speaker. More specifically the *Google Text To Speach API* was used. If the SONOS is used as interface to the user, the *title* and the *description* of the Interaction are concatenated, converted to an .mp3 file and then exposed though a public link to be accessible by the speaker.

### Summary of Devices' Drivers

Some of the functionalities implemented by the drivers are shared among some of the devices while some others are not. Here is a summary of what has been done.

[17] RxJs Subject: like an EventEmitters can emit values to the listeners, called subscribers in this case.
[18] `https://material.angular.io/components/dialog`
[19] `https://www.npmjs.com/package/sonos`

- Scan:
  - **Smartphone and Smart TV**: Both implement the discovery protocol designed for the project which uses UDP broadcast messages. Through experiments it was noticed that it was complicated for the smartphone to receive broadcast messages. Reminding that UDP messages may be lost because UDP is an unreliable protocol, the problem is due to the unpredictable behaviour of the Android service. To solve the problem an UDP broadcast message is sent from the MB every second for a maximum of three seconds just for the smartphone.
  - **SONOS Speaker**: The sonos node package already implementend a discovery method, hence the scan for this device is a wrapper for the library.
- Available: This behaviour is shared by all the devices. A simple ping request, to the local address of the devices, is used to test device connectivity. Devices availability is tested every 5 seconds and updated in real-time in the front-end part of the Message Broker.
- Send:
  - **Smartphone**: As anticipated, to send interaction to the Smartphone, the *Firebase Cloud Messaging Service* is used. A simple HTTP call to the *FCM* API is enough to transfer JSON data from the MB to the client.
  - **Smart TV**: The emulated Smart TV exposes an HTTP API whose address is exchanged during the discovery protocol. A single HTTP call is enough to transfer the data.
  - **SONOS Speaker**: For the speaker the complexity increases. First, it is necessary to generate an audio file through *Google's TTS* API and then, using the *node wrapper* for the official API, a command to the speaker is sent to play the audio that is reachable through a HTTP server started by the driver for this purpose.

**Discovery protocol**

The initial idea was to have two possible way to join the Message Broker network, depending on the device: **(i)** devices capable of running an *application* were considered the "active" part in the protocol, meaning that they were responsible to contact the MB and ask to join, **(ii)** devices incapable of running custom software were "passive" and need to be added **semi-manually** by the user, connecting on the admin panel and scan for devices or **automatic** by allowing the MB to automatically scan the network every X seconds (delayed add).

The plan was reworked to avoid unnecessary complexity. The final implementation consists of a simple discovery protocol that uses UDP Broadcast messages to find the devices. The devices that implement this protocol are the Smartphone and the emulated Smart TV. In Figure 3.8 it is shown the flow diagram representing the interaction between the Message Broker and a Device using the discovery protocol. Step by step:

1. The Message Broker sends a broadcast message specifying the kind of devices for which that message is for (TV or Smartphone) and its API URL in order to bee reachable. See Table 3.4 Server side message.

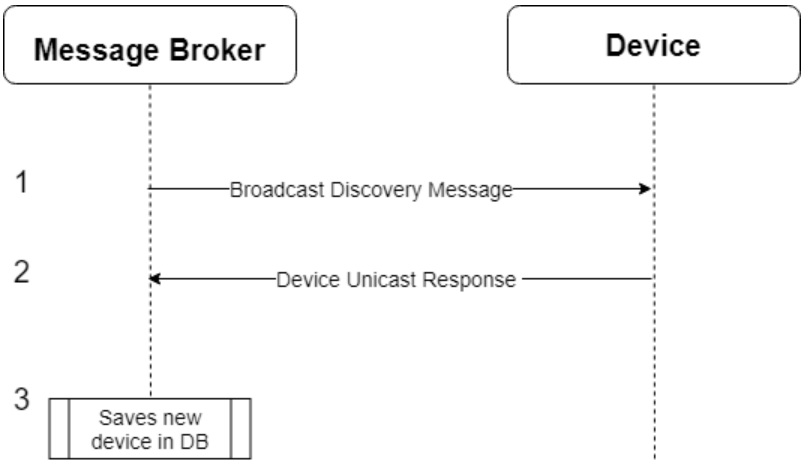



Figure 3.8: New device joining message broker network

2. The device that receives the message answers it with its information. See Table 3.4 Devices side message.

3. The Message Broker now temporarily stores the device and waits for the user to decide if that device will be part of the network or not. If so, it will be saved permanently and considered for the next interaction.

Table 3.4: Discovery protocol messages

| **Server side message** | |
|---|---|
| **Key** | **Value Type** |
| deviceType | String |
| API_URL | String |

| **Devices side message** | |
|---|---|
| **Key** | **Value Type** |
| deviceID | String |
| name | String |
| address | String |
| timestamp | Date |
| capabilities | Capabilities |

# Chapter 4

# Evaluation

A *user study* was conducted to evaluate the utility of the system and receive feedback on it, both in terms of user experience and possible improvements. Throughout this chapter we will see how the study was designed and explore its result.

Section 4.1 describes how the study was designed. In section 4.2 we will explore the result of the study and in section 4.3 we will discuss the limitations.

In summary: most of the participants were younger than 25 **years old**, which is significant because it is reasonable to expect that younger people are more familiar with technology. About 70% of the participants declared they have very low **understanding** about **network security**. Almost all of them stated they were **comfortable with the interface** used, even if they said there is space for UI improvements. More than 60% said "they were comfortable with the interface a lot". This led to have more than 75% of the participants **understand the actions required** by the notification system. Regarding the **feeling of being in control** of the situation, we have more variance in the answers with 4 people (30%) saying not at all, and 6 people (45%) saying "a lot". In general, 60% of the participants would be **willing to use** a system like the proposed one inside their own house, 30% is neutral and just 8% would not use it. Those who are in favor though would like to know more about the limitations of the system and the benefits it brings. Someone pointed out that the system itself may be prone to security issue, while someone else seemed to be bored by the "constant" notifications. Some of the participants would have preferred to be notified by a text instead of a push notification (regarding the phone), but those who were fine with the notification would have wanted it to be more "invasive", meaning that the phone should vibrate more and have a bigger icon in the notification bar. The suggestions about what can be improved mainly focused on the graphical aspect. Some of the participants pointed out that the notification should be more "educational".

## 4.1 User study design

In the following we review the steps taken to design and conduct the user study:

1. **Study goal**: first of all we needed to make an hypothesis, decide what we wanted to prove

with this user study. At high level our goal was to prove that the designed system is useful and can improve communications about security matters to non-expert users.

2. **Design of the experiment**: to prove the effectiveness of the system, we decided to have participants try all the devices that were implemented (Smartphone, Smart Speaker and Smart TV). These considerations guided the design of the scenarios and tasks to be presented to the participants. We decided to simulate an intrusion in the home network where the devices were connected and having different interactions with the participant in order to resolve it through two tasks:

   - Smartphone + Speaker: during this task the participant was requested to interact with the smartphone and the speaker to simulate normal use, allowing him/her to use the web browser (Google Chrome), apps like Youtube or Google News and the SONOS app to control the smart speaker. The different interactions were:
     (a) The speaker playing audio with a voice message telling the participant to reach out to the phone to check for warning messages, while the phone receiving a notification requiring to reboot the smartphone. In the smartphone notification a description was present, explaining why the problem occurred and why the reboot was necessary. More in detail the notification warned the user that the ip address assigned to the device was in conflict with another device and that new ips were assigned to the devices in conflict and hence, for the changes to be effective, the smartphone had to be rebooted.
     (b) The smartphone receiving a notification asking for some information. In particular it was requested if the participant used Youtube and if he/she rebooted the phone. Again a description was added explaining why these information were needed. More in detail the notification asked the user, through a form, if he/she used Youtube in the previous 10 minutes, if so for how long and if he/she rebooted the phone in the previous 10 minutes. All the questions were answered through a drop down list.
     (c) As last interaction a message was displayed to thank the user for being collaborative and saying that an intrusion attempt was successfully blocked.
   - Smart TV: the participant was requested to interact with the smart tv by watching different (pre-loaded) videos. The security-related interactions were:
     (a) A notification asking if the participant changed the network configuration. As before for the smartphone and speaker, the message had a description explaining why the message was received. More in detail the notification asked the participant if he/she changed the settings in the previous 10 minutes giving information about the automatic action taken by the system, consisting of assigning new ip addresses to devices in conflict.
     (b) A notification asking how many videos were watched and for how long.
     (c) And finally, a message to thank the user for being collaborative, saying that an intrusion attempt was successfully blocked.

   For the complete task scripts, please refer to Appendix D. During the design of the experiment it was also decided how to set up the room for the tasks and organize the interview. We omit the details.

3. **IRB approval**: “the Institutional Review Board is a type of committee that applies research ethics by reviewing the methods proposed for research to ensure that they are

ethical"[1]. IRB had to approve the user study by reviewing the final goal of the study, the proposed tasks, the way personal data was collected, treated and stored. After its approval we proceeded recruiting participants.

4. **Recruiting participants**: recruiting was mainly done by advertising the study on public boards within the CS building of WPI. The advertisement poster is showed in Appendix E. Some of the participants were directly recruited by the researcher, paying careful attention to avoid bias.

5. **Experimental procedure**. For each participant:

   (a) We gave information about the experiment. To ensure participants understood what was expected of them it was explained that the system was designed to improve communication of security matters to non-expert users.

   (b) We presented a consent form, that stated the purpose of the study, the procedures to be followed, rewards for the participants and how data was treated and stored.

   (c) We gave instructions concerning the tasks.

   (d) We let user perform the tasks while monitoring interactions from the administration panel of the Message Broker.

   (e) We administred a survey on *demographics* and *questions* related to the experiment. For the full survey, please refer to Appendix F.

## 4.2 Result

Since the number of people interviewed was relatively low, the study is more qualitative than quantitative in nature. We interviewed a total of 13 people, discarding the first 3 because of problems in the way the tasks were explained; in fact, those first 3 participants did not completely understand the tasks and failed to accomplish what was requested by the Message Broker. After revising the language in which the tasks were presented the problem did not persist.

Most of the questions of the survey were presented used a *likert scale*[2] from 1 to 5. Considering 1 and 2 as low/bad, 3 as neutral and 4 and 5 as high/good, we can now proceed exploring the result.

#### Age

Most of the participants (70%) were younger than 25, which is significant because younger people can be expected to be more prone to have lived their lives closer to technology, or be technology-dependant.

[1]Source: `https://en.wikipedia.org/wiki/Institutional_review_board`.
[2]Likert scale: `https://en.wikipedia.org/wiki/Likert_scale`.

### Technical background

Among the participants 70% stated their background on network security was low while only 10% said to have some understanding of the matter. This reassures us that the data is generally not biased by knowledge on security systems.

Both participants with high and low understanding of security expressed their willingness to use a system like the proposed one and their comfort using the current interface. The ones with scarce knowledge suggested that the notification/interaction should be more educational.

### Comfort with the interface

50% of the participants were comfortable with the current interface of the notification system, 40% neutral and 10% expressed low comfort with the interface even if he/she understood the actions required by the notification system. Quoting one of the participant that expressed low comfort, "If I was frequently alerted to IT security issues, I would simply stop using the internet. [...]". We could say that the discomfort was caused by the numerous notification received during the tasks. However it is important to take into account that the environment and conditions of the tasks were simulated and all the interactions were compressed into a little more that 30 minutes.

### Understanding of the actions required

80% of the participants understood the actions required by the notification system. The remaining 20% preferred to stay neutral (3 on the likert scale) on this question. During the tasks the researcher who performed the tests was monitoring the answers and actions of the participants from the Administration panel of the Message Broker.

### Feeling in control of the situation

For this question the answers are more spread out. 40% of the participants did not feel in control of the situation, 40% in control and the remaining 20% neutral. One of those who did not feel in control of the situation answered "I don't know what else I should expect, the vocal message is a bit worrying. [...]" to the question "Would you have liked to be notified in a different way?", this suggests that the feeling of not being in control may be due to scarce information given by the system itself, however from other open answers it was not possible to reach a more specific conclusion.

### Willingness to use this system

50% of the participants would be willing to use a system like this one, 40% neutral and 10% would not be willing to use it. Those who were neutral would like to know more about the limitations of the system and how vastly it can improve security. Someone pointed out that the system itself may be prone to security issues, making a valid point to be analyzed in future work. Another one said "[...] the notifications themselves were rather vague [...]" suggesting that the notifications should better explain the problem occurred.

#### Different type of notification

30% of the participants would like to be notified by a simple text message and in particular for urgent matters. 50% stated that they were comfortable with the current notification method even though, in case of the smartphone, the notification should be more noticeable, meaning a bigger icon, more vibration or a different sound.

#### Suggestions

Among the suggestion to improve the system we can find:

- Authentication process: the same participant who pointed out that the system itself may be prone to security issues suggested to implement some kind of authentication process without any further explanation. It certainly is something important to implement in a system that communicates automatic settings adjustments and action required that may be exploited by an attacker.
- More educational notification: it was expressed by more than one participant that it could be useful to have the notification to be more educational in order to help them better understand why the information required are needed and why certain countermeasures are taken automatically.
- Graphical aspect: there is a general sentiment about the need to improve the graphical aspect. Even though UI design was not the subject of this project it is important to consider it in future work since the way information is presented, both in terms of content and aspect, is relevant to make the user more comfortable in trusting the source and handling the requests.

## 4.3 Limitations

To better give significance to the result obtained by this study it is important to take into account the limitations.

First of all we need to consider the **number of participants**. There were only 10 (out of 13) participants from which we could extrapolate valid data. The limited number does not allow us to average the results and identify outliers.

The participants were from 20 to 30 **years old**, giving us idea of how younger generations approach security issues. Being younger, the participants were more likely to have used a computer or other form of technologies. It was not possible to conclude anything about older users.

Another important aspect to consider is that most of the participants, due to the recruiting process, were selected from within the Computer Science department of WPI, and the ones who were not, were from other departments or from other universities. The implication here is that all the participants have a **high level education**, and as matter of fact, were exposed to new

technologies and more likely to know more about security than people with lower level education and/or coming from contexts that do not require interacting with computer technology.

# Chapter 5

# Discussion

Through this chapter we will first examine the process of development and then discuss the limitations of the prototype and possible future developments. Most of the limitations are due to limited time, hence could be resolved in future developments. There were no limitations due to the technologies used, and all the requirements were fulfilled.

## 5.1 Workflow

The whole thesis work was performed in approximately 5 months. To better organize the work we decided to go through 6 phases:

1. **Thesis outline**: at the beginning an outline of the whole project was prepared, this included: estimating the timing for each of the following phases, deciding which devices to use as interfaces from and to the user and what technologies to use. The result was an approximate time table. As tool to keep track of the progress we used *Trello*[1].

2. **Research/related work**: the very first part of the thesis involved performing background research related work. People's attitudes on security matters and HCI were the main topics of the research. For this phase, we mainly used *Google Scholar*[2], *IEEE Xplore Digital Library*[3] and *ACM Digital Library*[4]. The research stopped when most of the articles/publications started to reference each other. During the process notes were taken, then reflected into Chapter 2.

3. **System Design**: once the preliminary part was completed we proceeded to designing the system to solve the problem stated in Paragraph 1.1, that is devising a new way of interacting with the user. The main challenge during this phase was deciding which technologies to use, the tension was between using more mature technologies with a wider support community but based on old programming patterns, and new cutting edge technologies, saving time but potentially deal with lack of reference. Picking an option was not easy. A table of features needed for the project was prepared and the whole system

[1] `https://trello.com`
[2] `https://scholar.google.com`
[3] `https://ieeexplore.ieee.org`
[4] `https://dl.acm.org`

was designed as a set of small components (e.g. front-end, server, DBMS, socket service) independent one from each other. This enabled us to pick different technologies for each of those components allowing to swap them in case of unsolvable problems.

4. **Implementation**: this phase was fairly straightforward since the design phase took into account most of the problem that could have risen during the implementation. The implementation phase was further divided into 4 sub-phases:

   (a) **Message Broker**: front-end and back-end, initially developed separately to initialize the frameworks (Angular, Node, Express) with their libraries and then connected together.

   (b) **SmartTV App**: since this component is emulated with Angular and Node, the same process used for the MB was followed, first implementing the interface and then enabling it to join the MB network and receiving interaction requests. It was decided to developed this device first since the technologies used were the same used for the Message Broker system.

   (c) **SONOS Speaker**: for this device it was not necessary to learn a new programming language or a new framework since it is not capable of running an application but just receives HTTP API calls. After trying with the official HTTP API resulting in a non-working system it was decide to switch to a third-party library that. This generated minor limitations about device discoverability as we will see in Section 5.2.

   (d) **Smartphone App**: the Android app was challenging because of the nature of the device. A smartphone is not wired to the network (like the other devices) hence its connectivity caused some issues with the discovery protocol. Also, background services (like the one for discoverability) are handled by the operating system and their running status is unpredictable. We will see in Section 5.2 that this led to some limitations. Luckily a large online community and reference was available to find solutions for some of the problems encountered.

5. **User study**: during this phase we designed and conducted a user study to evaluate the utility of the system and receive feedback on it. The main steps to complete this study were: **(i)** set goal, the hypothesis we wanted to prove; **(ii)** design the experiment, how the participants to the study were going to use our devices; **(iii)** obtain the authorization by the IRB to perform the study; **(iv)** recruiting participants; **(v)** perform the experiment and finally **(vi)** analyze the result.

6. **Wrapping up**: last phase, consisting of reviewing all the work, collecting all the documents to make appendixes and making sure everything was consistent.

Writing the thesis was performed in parallel with the phases above.

## 5.2 Prototype limitations & Future work

Through this section we will see the limitations of the prototype and what could be done in future work to improve the whole system. First the Message Broker system will be discussed and then each one of the devices used as interfaces.

An issue that surely has to be considered for future work and affects all the parts of the system is a better **error handling**. Some times, proper error handling was left behind to not extend the codebase too much and increase its complexity, even if this may lead to some confusion to those who will use the system without a deep knowledge of the technologies used and encounter an error. Although some edge cases were left behind, there is consistent logging of all the actions the system performs, therefore it is easy to backtrack problems to their source.

### 5.2.1 Message Broker

No significant problems arose during the development of the Message Broker. As already pointed out, the only limitations were due to shortage of time.

We decided not to use **MongoDB** (or other DMBS)in order to speed up the development, but it will be easy to integrate it in further development since the *Mongoose* library is used and already takes care of the data structure and will take care of the connectivity to the database. As for now data will be reset when restarting the NodeJS server (i.e. Message Broker back-end).

Another important aspect to be considered for future developments is the possibility to run the Message Broker on **Cloud environments**. It is not the Message Broker per se that needs changes but the drivers (see paragraph 3.4.1) of the devices. In fact, currently, the drivers only work on local networks even if some of their functions already go through cloud systems (e.g. firebase cloud messaging system, google tts). On the Message Broker side, the only thing needed would be a proper network configuration to allow the API to be publicly accessible. Of course, making the API publicly accessible will require some sort of **authentication** of the clients using them. A task that will not be complicated because of the integration between NodeJS/Express and libraries like *Passport.js*[5].

Improvements are also possible for the **decision system** of the Message Broker. So far the MB picks a random device among the ones who have enough capabilities to handle an interaction request. It would be better to choose the device to which the user is paying more attention and is more willing to interact with. In a non technical view the system should be adaptive and reactive to the user preferences. A critical aspect to implement is the re-sending of an interaction after the previous one was ignored after a certain timeout. The current configuration just marks the interaction request as timed out after 30 seconds, without taking any further step. It may be useful to send the same interaction to another device (if present) or change the severity level (field implemented but not yet used) to show/play the interaction in a different form.

For what concerns interactivity of the admin panel there is a possible optimization in the messages exchanged from the back-end to the front-end with Socket.io. As said in Section 3.4.1, currently, the whole array representing interactions and devices is sent while the optimal way to update data would be to send just the updated part of it. A problem caused by this is that whenever an expansion panel is open and new data is received the panel auto collapse because Angular renders the whole component again.

[5] `http://www.passportjs.org/`

Several other small improvements could be made:

- The *Access-Control* of the server should be set properly to avoid Cross-origin requests.
- Imports can be optimized creating a module containing all the dependencies needed.
- Environment variables such as port numbers and static addresses should be set inside a '.env' file, for example using the *dotenv* node module.
- The server should log all the actions on a file for latter debugging.

### 5.2.2 Devices

Keeping in mind that the devices weren't the focus of this project we needed to develop some applications for some devices in order to test the Message Broker capabilities. In particular, the devices used were an Android smartphone, a Raspberry Pi emulating a Smart TV through an Angular web-app and a SONOS speaker. These three devices covered almost all the possible ways of interacting with a user. In addition to the possible code optimizations, some other improvement are going to be discussed in the following.

#### Smartphone

The smartphone used was a Google Pixel XL running Android 9. The only limitation to be considered for this device regards the possibility to keep a **service running** in background because the OS may randomly kill processes when more memory is needed. A background service is used to implement the discovery protocol and the only way to make sure the service is running is to have the app open. This leads to a situation in which it is necessary to open the app to make it discoverable by the Message Broker. Once added it is not necessary to keep the app open, not even in background, since the rest of the functionality is run through a system service that is never killed. There may be two possible solution for the problem: **(i)** create a foreground service which is harder to kill (not impossible) that has some other limitations such as displaying a persistent notification; **(ii)** use *network service discovery*[6] that is a protocol already implemented to let the device being discoverable from other devices using the same protocol.

For what concerns the notifications sent to the smartphone, a limitation is present since the nofitication is sent to all the smartphones that have the app. Since during the test phase just one smartphone was used there was no problem, but in multi-user and/or multi-smartphone scenarios this would be a problem. This is easy fixable by sending to the Message Broker the device ID when exchanging information with the discovery protocol and using it when sending the notification. In the Message Broker internal representation of the device (see Section 3.3) there already is a field where to store this kind of ID.

#### Smart TV

It is somehow incorrect to talk about a "Smart TV" because this component is a web-app that emulates an Android TV. Here, the limitation was again the time. It would have been harder

[6] `https://developer.android.com/training/connect-devices-wirelessly/nsd`

to develop for a different platform, even if Android TVs use the same architecture of Android Smartphones. The result is that the user cannot interact with it as if it was a real TV, but it is sufficient for testing purposes. We leave as future work to integrate a real Smart TV, Android or not, as device in the Message Broker network.

#### SONOS Speaker

Even if there are official API for the SONOS Speaker, during the development, some issues were found. After following all the steps to obtain access to the API, some of them were not working. In particular those not working were the ones that change the status of the speaker such as changing volume or playing a different track, while the ones to obtain data (get volume level, get track number) were working. After receiving no answer on the problem from the official community on their forum and on StackOverflow, it was decided to use an unofficial library that solved all the problems but with the limitation that it only works locally. We leave as future work to better understand the reason why official API are not working and the API calls return a 500 status code with no explanation.

# Chapter 6

# Conclusion

The goal of this thesis was to create a new way of interacting with users regarding security matters, trying to better involve people without making them feel incapable of understanding and taking the proper actions and countermeasures when needed. The solution we wanted to create was meant to take the form of a computer program, runnable both in local and cloud environments, scalable in terms of devices that can be used as interface from and to the users and utilizable as test system for other researchers, hence with a easy to understand and use interface.

The approach we followed was to first understand what has already been done on the matter, performing research which is documented in Chapter 2 Related Work, then we tried to model the project objective under the form of questions we wanted to answer. After identifying the main features we wanted the final project to have, we chose the technologies that would have better allowed us to achieve our goal.

We managed to make what we planned, resulting in a multi-platform system capable of involving the user in security processes, both with the user intervention required and not. Some modification and improvements still have to be done but the overall system is fully functional. With promising feedback obtained by the user study we conducted, we hope the project will have a follow up. In fact, people who participated to our user study expressed their willingness to use a system such the one we designed and prototyped.

As said there still are modifications and improvements to be implemented in the system. The codebase should be refactored and better organized; run time errors should be better handled providing information to the users and not just developers who know how to code; data persistence should be added through the use of a non-relational database; the decision system (which device to use as interface for a specific interaction) can be improved; support for more devices has to be done and some other small changes that would make the system perform better, both in terms of usability and performances.

We are happy to say that this project was successfully completed and we hope it can be used for future researches and/or projects. The overall experience was gratifying and helped the student learn how to perform research and organize and plan a big project.

# Appendix A

# Documentation

In the following the documentation of the project is presented: all the steps required to install and make the system work. All the parts of the system are included: Message Broker, emulate Smart TV, SONOS speaker and the Smartphone App. It is also present a description on how to use the *Administration panel* of the message broker to send and check the status of interaction requests.

# Documentation

In the following it will be discussed how to start the Message Broker system (back-end and front-end) and how to run all the applications developed: emulated Smart TV, Smartphone Android APP and SONOS Speaker.

Index:



## 1. Starting Message Broker

GitHub repository: https://github.com/Xfox1/message-broker.

After downloading and extracting it to a folder, open the project with any editor. Using Visual Studio Code[1] is highly suggested.

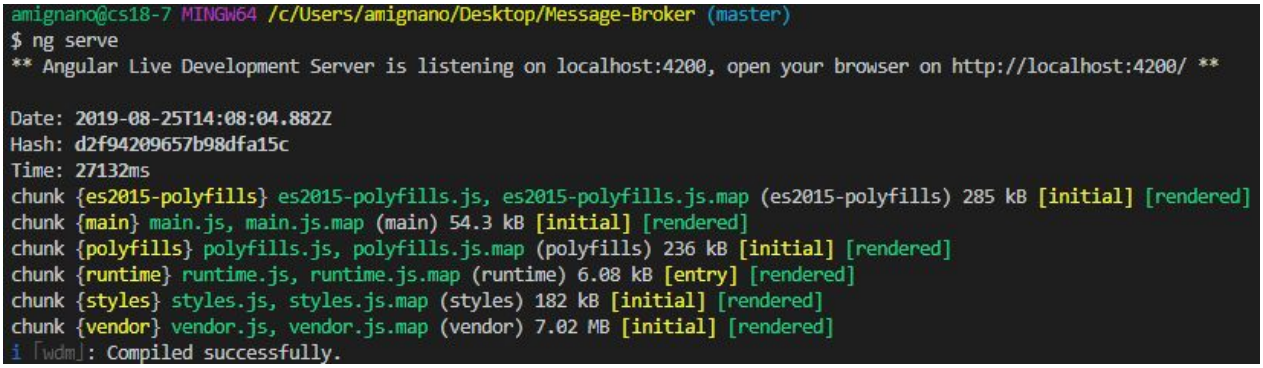
1. First of all, it is necessary to install all the dependencies. Open a terminal inside Visual Studio Code (*CTRL + `*) or a system terminal, move to the project folder and execute:

```
npm install
```

2. Now run *Angular* development server (front-end):

```
ng serve
```

It should show the following result:




[1] https://code.visualstudio.com/

3. Now it is time to run the back-end, the real application. On another terminal (click the + icon on the top right corner of the terminal in VSC) run the following command:

   ***nodemon server.js***

   It should show the following result:


4. The system is now up and running, to access the interface open a web browser and go to:

   ***http://localhost:4200/***

## 2. SONOS Speaker

No actions are required for the speaker to be discoverable by the MB. It has just to be on the same network of the Message Broker.

## 3. Smart TV

GitHub repository: https://github.com/Xfox1/thesis-tv-app
Since the Smart TV app is emulated using Angular, the steps to run it are almost equal to the ones of the Message Broker:

1. Install dependencies:

   ***npm install***

2. Run Angular development server:

   ***ng serve***

3. On another terminal move to the ./server folder and execute:

   ***nodemon index.js***

4. On a web browser visit:

***http://localhost:4200/***

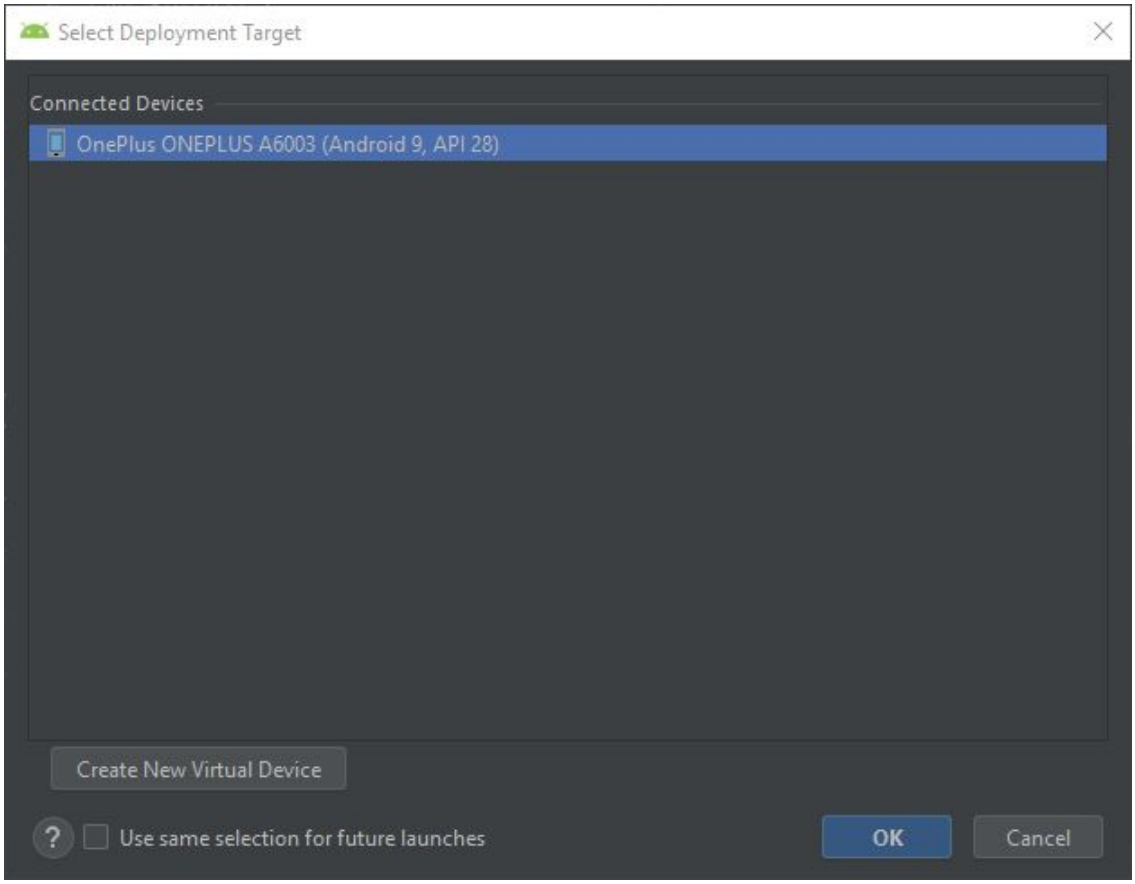
## 4. Smartphone APP

GitHub repository: https://github.com/Xfox1/thesis-smartphone-app

1. After downloading and opening the project with Android Studio[2] connect an Android device via USB to the computer and press the green run ( ) button in the top right corner.

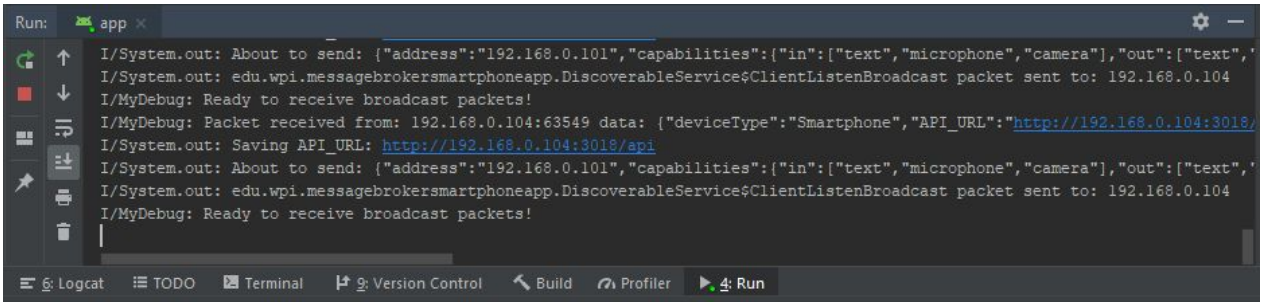

2. Wait for the new prompt to show to show the phone connected via USB. If it is not shown, it may be required to enable USB debugging from the phone's settings.

3. Select the device and wait for the app to be installed on it.



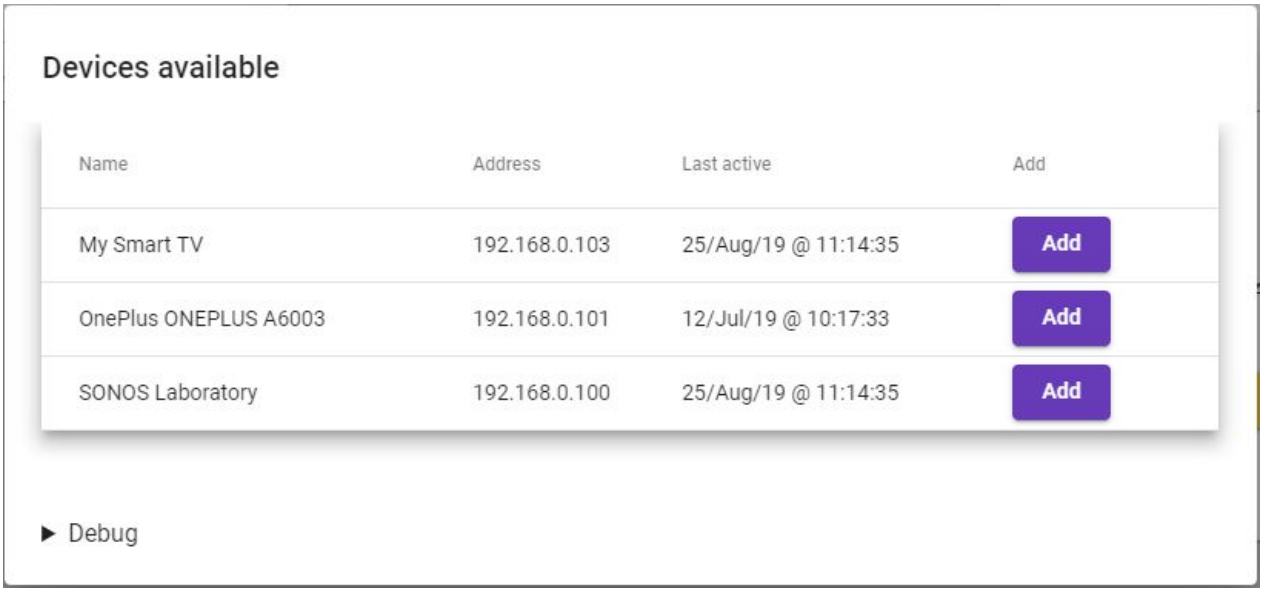

[2] https://developer.android.com/studio

4. After the application is installed, it is not required to leave the phone plugged, although it may be useful to go on the Run tab ( 4: Run ) in the bottom left corner, to inspect the debugging messages sent from the app.

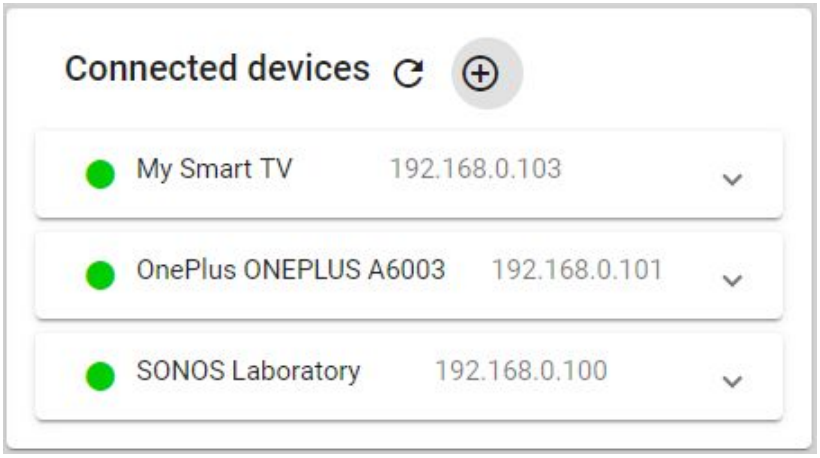



## 5. Add devices to the Message Broker network

1. On the Connected devices card click the ⊕ sign, and wait for the popup to load the discovered devices.

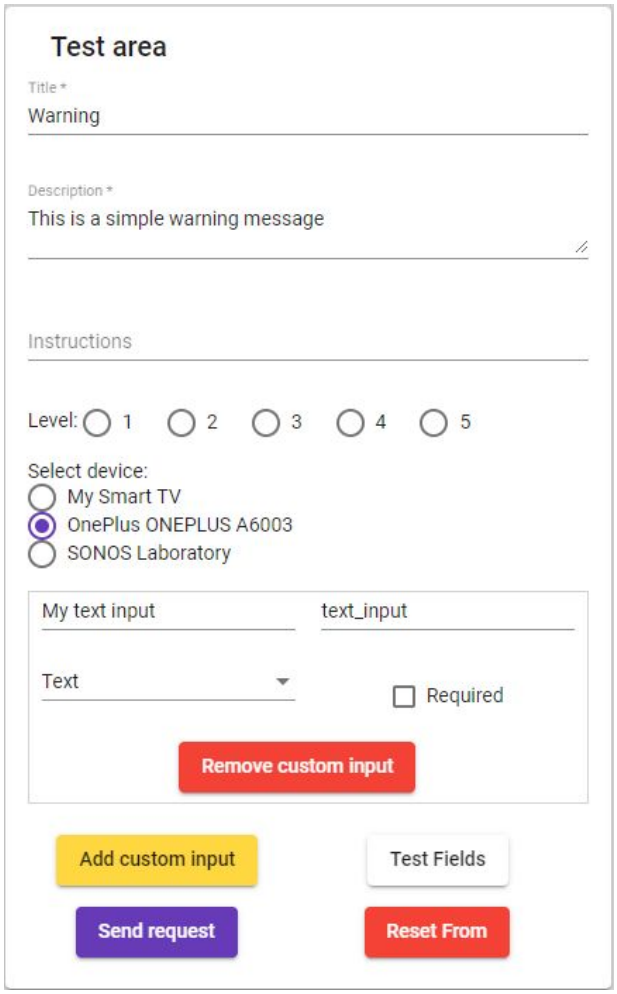



2. Click Add next to all the devices that are needed to join the Message Broker network. They should appear in the Connected devices card.

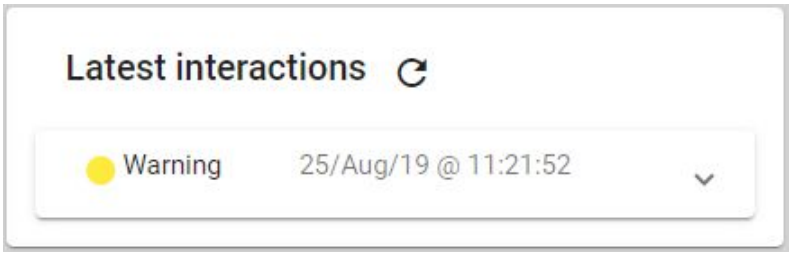



## 6. Send interaction

1. In the Test area card write complete all the mandatory field (title and description). It is also possible to choose to which device the interaction has to be sent. Custom inputs can be added or a set of predefined test fields can be user (clicking on Test Fields button).



2. After sending it, it should appear as pending interaction (yellow dot) in the Latest interactions card.



# Appendix B

# Data Structure

In the following the data structure used by the Message Broker is presented. All the data was handled by *Mongoose* and treated as JSON objects. To better understand, please see Section 3.3.

# Data Structure

| Device | | |
|---|---|---|
| **Field name** | **Type** | **Note** |
| _id | String | Automatic id given from mongoose |
| deviceID | String | id generated form the device or driver |
| name | String | Name of the device |
| description | String | Optional description of the device |
| capabilities | Capabilities | List of input/output capabilities |
| lastActive | Date | Timestamp of last recorder activity |
| address | String | IP Address |
| driverID | String | id of the driver controlling the device |
| online | Boolean | Online flag |

| Interaction | | |
|---|---|---|
| **Field name** | **Type** | **Note** |
| _id | String | Automatic id given from mongoose |
| us_id | String | Unique Sequential id |
| timestamp | Date | Timestamp of creation |
| status | Number | 1 = pending, 2 = completed, 3 = canceled |
| title | String | Title of the interaction |
| description | String | Description of the interaction |
| specific | String | Shown in “more/less” hidden text |
| instructions | String[] | List of instructions, separated by colons |
| level | Number | Severity level: [1-5] |
| inputs | InteractionInput[] | Object representing input form |
| image | Image | For further development |
| response | InteractionResponse[] | Object containing the response if any |

| Capabilities | | |
|---|---|---|
| **Field name** | **Type** | **Note** |
| in | String[] | text \| microphone \| camera |
| out | String[] | text \| video \| audio \| vibration \| notification |

| InteractionInput | | |
|---|---|---|
| **Field name** | **Type** | **Note** |
| title | String | Text next to the field |
| name | String | Name of the field, single word (no spaces) |
| type | String | One of the following values: text \| textarea \| checkbox \| select \| button |
| elements | [{text: String, value: String}] | Only if *type* is set to *select*, list of elements |
| required | Boolean | Specify if field is required or not. |

| InteractionResponse | | |
|---|---|---|
| **Field name** | **Type** | **Note** |
| name | String | Name of the field, single word (no spaces). |
| value | String | Value inserted by the user. |

# Appendix C

# API Calls

In the following the API calls and endpoints exposed by the Message Broker are presented. All the calls provide examples of requests (when not obvious), examples of responses and possible status codes.

# API Calls

| **GET** | /api/interactions | Get all interactions |
|---|---|---|

| **Response codes** | |
|---|---|
| **Status Code** | **Meaning** |
| 200 | List of devices returned |
| 500 | Internal Server Error |

**Response**

```
[
  {
    “title”: “Warning: intrusion”,
    “message”: “An intrusion was detected and blocked by the system.”,
    “level”: 2,
   },{
     “title”: “Danger”,
    “message”: “Reboot required”,
    “level”: 5,
  }
]
```

| **POST** | /api/interactions | Request interaction |
|---|---|---|

| **Params** | | | |
|---|---|---|---|
| **Name** | **Req.** | **Type** | **Note** |
| *unnamed* | yes | Interaction | The interaction object that is intended to be created |

| **Response codes** | |
|---|---|
| 201 | Request created |
| 500 | Internal Server Error |

**Request's body example**

```
{
  "title": "Warning: intrusion",
  "message": "An intrusion was detected and blocked by the system.",
  "specific": "On address 192.168.1.52, TCP port 89 was detected suspicious traffic related to a trojan known by the name of 'virtus'",
  "level": 2,
}
```

| **Response** |
|---|
| Interaction |

| **Response example** |
|---|
| Same as the body in case of success |

| PUT | /api/interactions/:id | Response to interaction |
|---|---|---|

| Body Params | | | |
|---|---|---|---|
| **Name** | **Req.** | **Type** | **Note** |
| data | yes | InteractionResponse[] | |

| Response codes | |
|---|---|
| 202 | Interaction updated |
| 404 | Interaction not found |
| 500 | Internal server error |

**Request's body example**

```
[
  {"name": "input_field_1", "value": "Kitchen camera"},
  {"name": "checkbox_input", "value": true},
]
```

**Response example**

```
{
  "message": "Interaction updated successfully",
  "obj": request_object
}
```

| GET | /api/devices |
|---|---|

| Response code | |
|---|---|
| **Status Code** | **Meaning** |
| 200 | Ok |
| 500 | Internal Server Error |

| Response |
|---|
| Device[] |

**Response example**


```
[
   {
      "online": true,
      "name": "Smartphone",
      "description": "Description",
      "content": "Content"
   },
   {
      "online": false,
      "name": "Smart TV",
      "description": "Description",
      "content": "Content"
   },
   {
      "online": true,
      "name": "SONOS Speaker",
      "description": "Description",
      "content": "Content"
   }
]
```

| GET | /api/devices/scan | Search new devices |
|---|---|---|

| Response codes | |
|---|---|
| **Status Code** | **Meaning** |
| 200 | List of new devices |
| 500 | Internal Server Error |

| Response |
|---|
| Device[] |

**Response example**


```
[
   {
      "_id": "5d2e1cea7e8b78379c50ffa4",
      "name": "SONOS Laboratory",
      "lastActive": "2019-07-16T18:52:26.091Z",
      "address": "192.168.0.101",
      "online": true,
      "driverID": "5d2e1abe7e8b78379c50ffa2",
      "capabilities": {
         "in": [],
         "out": [
            "audio"
         ]
      },
      "deviceID": "5d2e1cea7e8b78379c50ffa4"
   }
]
```

| **POST** | /api/devices/scan | Search new devices |
|---|---|---|

| **Body Params** | | | |
|---|---|---|---|
| **Name** | **Req.** | **Type** | **Note** |
| id | yes | String | The id of the device that is needed to be added |

| **Response codes** | |
|---|---|
| **Status Code** | **Meaning** |
| 201 | Device added |
| 500 | Internal Server Error |

| **Request example** |
|---|
| {“id”: “5d2e1cea7e8b78379c50ffa4”} |

| **Response** |
|---|
| Device[] |

# Appendix D

# Task Scripts

In the following the task scripts used during the user study are presented. See Chapter 4 for a discussion about result.

# Task scripts

1. **[Smartphone + SONOS]**
   a) **Information for the user:** “You’ll be given this smartphone and you can use Google Chrome, Youtube and the SONOS app to control the speaker. Please do not turn off the device unless requested, disable the WiFi, install new software or change any settings.

   There is a Speaker too that, for sake of simplicity, will already play some music and from which some alert message may be played. Feel free to press the button to play or pause it or use the app on the smartphone to control it.”

   During the test, one ore more devices may receive and display notifications about IT security problems happening within the network to which the device is connected.. All the messages refer to simulated problems (in other words, you will not be dealing with actual instances of hacking, only simulated ones). Through these notifications, the system may request that you insert some data referring to the activities you performed using the devices . Let me also remind that any of the information you may insert will not be correlated to your name or any other personal identifiable information. However, we ask that you refrain from entering usernames, passwords, or any sensitive personal information in the provided smartphone.

   b) **Notifications:**
      i) [Sonos]
      **Title**: Warning
      **Message**: please check your phone for warning messages.

      ii) [Smartphone]
      **Title**: Warning - Action required
      **Message**: The network settings on this phone were automatically updated to resolve a problem. . Please turn this phone off, then turn it on again. This will cause the phone to refresh its settings.
      Technical Details: the ip address assigned to this device was in conflict with another device. New static ips have now been assigned to the devices in conflict.

      iii) [Smartphone]
      **Title**: Warning - Information request
      **Message**: The system has detected potentially suspicious network activity and needs your help to understand this activity.
      Details: The following information will help to perform traffic analysis to detect the presence of intruders in the network.
      **Input**:
      - Did you use Youtube in the past 30 minutes? Yes | No

- If you used Youtube, for how long did you do so? Less than 1 minute | Between 1 and 10 minutes | More than 10 minutes| I don’t remember
- Did you reboot the phone in the past 30 minutes? Yes | No

iv) [Smartphone]
**Title:** Information
**Message:** Thank you for providing all the information required. An intrusion attempt was detected and blocked.

## 2. [SmartTV]

a) **Information for the user:** “You are asked to interact with this emulated Smart TV. Emulated means that not all the functionalities are available. In particular, this TV is not connected to streaming services such as Netflix and Hulu; however, you can browse a selection of pre-loaded videos and watch as many as you want.
Please refrain from turning off the device unless requested, disable the WiFi, install new software or change any settings.

During the test, the device may receive and display notifications about IT security problems happening within the network to which the device is connected. All the messages refer to simulated problems (in other words, you will not be dealing with actual instances of hacking, only simulated ones). Through these notifications, the system may request that you insert some data referring to the activities you performed using the devices Let me also remind that any of the information you may insert will not be correlated to your name or any other personal identifiable information. ”

b) **Notifications**:

v) **Title**: Warning - Information required
**Message**: An unexpected change in the settings of this television was detected. Please answer the question below.
Details: the ip address assigned to this device was in conflict with another device.

**Input:** Did you change the network settings of this device in any way in the past 30 minutes?Yes | No

vi) **Title**: Warning - Information required
**Message**: Unexpected and potentially suspicious traffic coming from this device was detected in the past 30 minutes. Please fill the following form to help the system detect potential problems.

**Input:**
- How many videos did you watch? 1 | 2 | 3+ | I don’t remember
- If you watched one or more videos, for how long did you do so? Less than one minute | for one to ten minutes | for more than ten minutes | I don’t remember

vii) **Title:** Information
**Message:** Thank you for providing all the information required. An intrusion attempt was detected and blocked.

# Appendix E

# Advertisement

In the following we can see the poster used to advertise the user study.

( $ Rewarded $ )

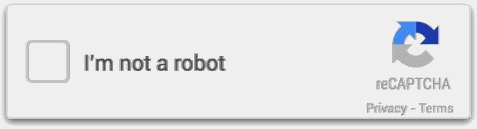
# 💻Participate to a Study on IoT security🌐

WPI

Fuller Labs
Worcester Polytechnic Institute

## Eligibility criteria:

- Being human :-)




## Effort (time required, etc.)/task

- 30mins - 1 hr
- No competences needed
- Follow simple directions to secure smart home devices (IoT)

## Purpose of the research

- Understand usability of notification system for smart home security

## Reward
## $15 Amazon Gift Card

## More Info

Antonio Mignano - amignano@wpi.edu
Lorenzo De Carli - ldecarli@wpi.edu

# Appendix F

# Survey

In the following we can see the survey the participants to the user study had to fill after the experiments.

# Survey

The information provided won't be in any way correlated to your name or any identifiable information.

* Required

1. **Age** *

2. **How much background/understanding do you have on network security?** *
   *Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| low | ◯ | ◯ | ◯ | ◯ | ◯ | high |

3. **How comfortable were you with the interface used to display notifications and request information?** *
   *Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| not at all | ◯ | ◯ | ◯ | ◯ | ◯ | a lot |

4. **Do you feel you understood the actions required by the notification system?** *
   *Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| not at all | ◯ | ◯ | ◯ | ◯ | ◯ | a lot |

5. **As you were interacting with the system, did you feel in control of the situation?** *
   *Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| not at all | ◯ | ◯ | ◯ | ◯ | ◯ | a lot |

6. **How likely would you be to use a system like the one you interacted with - which can help detecting and resolving IT security issues - in your own home?** *
   *Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| not at all | ◯ | ◯ | ◯ | ◯ | ◯ | a lot |

7. **Please briefly explain your answer to the question above** *

8. **Would you have liked to be notified in a different way?** *

9. **What do you think can be improved?** *

# Bibliography


[1] Bonnie Brinton Anderson et al. “How Polymorphic Warnings Reduce Habituation in the Brain: Insights from an fMRI Study”. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI ’15. Seoul, Republic of Korea: ACM, 2015, pp. 2883–2892. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702322. URL: http://doi.acm.org/10.1145/2702123.2702322.

[2] O. Arias et al. “Privacy and Security in Internet of Things and Wearable Devices”. In: *IEEE Transactions on Multi-Scale Computing Systems* 1.2 (Apr. 2015), pp. 99–109. ISSN: 2332-7766. DOI: 10.1109/TMSCS.2015.2498605.

[3] Cristian Bravo-Lillo et al. “Harder to Ignore?” In: *Proceedings of the Tenth USENIX Conference on Usable Privacy and Security*. SOUPS’14. Menlo Park, CA: USENIX Association, 2014, pp. 105–111. ISBN: 978-1-931971-13-3. URL: http://dl.acm.org/citation.cfm?id=3235838.3235847.

[4] L Camp. “Mental Models of Privacy and Security”. In: vol. 28. Feb. 2009, pp. 37–46. DOI: 10.1109/MTS.2009.934142.

[5] Vincent C. Conzola and Michael S. Wogalter. “A Communication–Human Information Processing (C–HIP) approach to warning effectiveness in the workplace”. In: *Journal of Risk Research* 4.4 (2001), pp. 309–322. DOI: 10.1080/13669870110062712. eprint: https://doi.org/10.1080/13669870110062712. URL: https://doi.org/10.1080/13669870110062712.

[6] Paul Dourish et al. “Security in the Wild: User Strategies for Managing Security As an Everyday, Practical Problem”. In: *Personal Ubiquitous Comput.* 8.6 (Nov. 2004), pp. 391–401. ISSN: 1617-4909. DOI: 10.1007/s00779-004-0308-5. URL: http://dx.doi.org/10.1007/s00779-004-0308-5.

[7] *Functional magnetic resonance imaging*. URL: https://en.wikipedia.org/wiki/Functional_magnetic_resonance_imaging.

[8] J. Granjal, E. Monteiro, and J. Sá Silva. “Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues”. In: *IEEE Communications Surveys Tutorials* 17.3 (thirdquarter 2015), pp. 1294–1312. ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2388550.

[9] Joshua B. Gross and Mary Beth Rosson. “Looking for Trouble: Understanding End-user Security Management”. In: *Proceedings of the 2007 Symposium on Computer Human Interaction for the Management of Information Technology*. CHIMIT ’07. Cambridge, Massachusetts: ACM, 2007. ISBN: 978-1-59593-635-6. DOI: 10.1145/1234772.1234786. URL: http://doi.acm.org/10.1145/1234772.1234786.

[10] *Habituation*. URL: https://en.wikipedia.org/wiki/Habituation.

[11] *Keeping an eye on the data usage of all those smart home devices*. URL: https://corpblog.viasat.com/keeping-an-eye-on-the-data-usage-of-all-those-smart-home-devices/.

[12] Willett Kempton. "Two Theories of Home Heat Control*". In: *Cognitive Science* 10.1 (1986), pp. 75–90. DOI: 10.1207/s15516709cog1001_3. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1207/s15516709cog1001_3. URL: https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1001_3.

[13] Y. A. Mtawa, A. Haque, and B. Bitar. "Does Internet of Things Disrupt Residential Bandwidth Consumption?" In: *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. Aug. 2018, pp. 1–5. DOI: 10.1109/VTCFall.2018.8690652.

[14] Pardis Emami Naeini et al. "Privacy Expectations and Preferences in an IoT World". In: *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*. Santa Clara, CA: USENIX Association, 2017, pp. 399–412. ISBN: 978-1-931971-39-3. URL: https://www.usenix.org/conference/soups2017/technical-sessions/presentation/naeini.

[15] Norbert Nthala and Ivan Flechais. "Informal Support Networks: an investigation into Home Data Security Practices". In: *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*. Baltimore, MD: USENIX Association, 2018, pp. 63–82. ISBN: 978-1-931971-45-4. URL: https://www.usenix.org/conference/soups2018/presentation/nthala.

[16] Bruce Schneier. *Security Economics of the Internet of Things*. 2016. URL: https://www.schneier.com/blog/archives/2016/10/security_econom_1.html (visited on 05/01/2019).

[17] S. Sicari et al. "Security, privacy and trust in Internet of Things: The road ahead". In: *Computer Networks* 76 (2015), pp. 146–164. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2014.11.008. URL: http://www.sciencedirect.com/science/article/pii/S1389128614003971.

[18] Rick Wash. "Folk Models of Home Computer Security". In: *Proceedings of the Sixth Symposium on Usable Privacy and Security*. SOUPS '10. Redmond, Washington, USA: ACM, 2010, 11:1–11:16. ISBN: 978-1-4503-0264-7. DOI: 10.1145/1837110.1837125. URL: http://doi.acm.org/10.1145/1837110.1837125.

[19] Rick Wash and Emilee Rader. "Influencing Mental Models of Security: A Research Agenda". In: *Proceedings of the 2011 New Security Paradigms Workshop*. NSPW '11. Marin County, California, USA: ACM, 2011, pp. 57–66. ISBN: 978-1-4503-1078-9. DOI: 10.1145/2073276.2073283. URL: http://doi.acm.org/10.1145/2073276.2073283.

[20] Eric Zeng, Shrirang Mare, and Franziska Roesner. "End User Security and Privacy Concerns with Smart Homes". In: *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*. Santa Clara, CA: USENIX Association, 2017, pp. 65–80. ISBN: 978-1-931971-39-3. URL: https://www.usenix.org/conference/soups2017/technical-sessions/presentation/zeng.

[21] Serena Zheng et al. "User Perceptions of Smart Home IoT Privacy". In: *Proc. ACM Hum.-Comput. Interact.* 2.CSCW (Nov. 2018), 200:1–200:20. ISSN: 2573-0142. DOI: 10.1145/3274469. URL: http://doi.acm.org/10.1145/3274469.

# Thanks

Throughout the writing of this thesis I have received great support and assistance.

I would first like to thank my parents for their wise counsel and sympathetic ear, my brother for sharing with me his experience and giving me advice and all my relatives who were always there to support me when needed.

Special thanks go to my closest friends Eva, Fausto, Kristel and Sebastiano, capable of giving me encouragement when facing difficult times.

I would also like to thank other friends of mine who were always listening to my ideas and projects, keeping my innovative spirit alive: Christopher S., Marco M.M., Stefano G. and Vito D..

Other special thanks to my supervisors Professor Lorenzo De Carli and Professor Riccardo Sisto who gave me the opportunity to do this experience.

Thank you