Bachelor thesis
Computer Science

Radboud University

Investigation into Google's voice search technology

Author:
Phil Geelhoed
S4118901

First supervisor/assessor:
Prof. dr. B.P.F. Jacobs
Institute for Computing and
Information Sciences (ICIS)
bart 'at' cs.ru.nl

Second assessor:
drs. P.N. Meessen
pmeessen@cs.ru.nl

June, 2018

Abstract

Today, Google is steadily introducing voice controlled services in their products. Simply saying "Ok Google" allows you to interact with your device in numerous ways. However, for a device to 'hear' these words, the microphone has to be activated before the interaction actually starts. This realization is our motivation to shed some light on voice controlled technology. By combining technological research and qualitative experiments to research an Android device, we reveal that the microphone is actively listening when the screen is turned on, which raises privacy concerns. These concerns led us to investigate Google's updated privacy agreements using the European GDPR as a guideline. This paper reviews the different aspects of voice controlled search to give a complete view of how the vocal data is processed and if this is legal.

Contents

Chapter 1

Introduction

The use of voice controlled search to command or interact with digital devices is not new. Worldwide, people make daily use of the voice controlled search application on mobile devices. In the early 1950s the first voice recognition system, so called "Audrey", was developed by Bell laboratories. This system could understand digits of a single human voice. This does not sound impressive anymore, but back then it was as 'innovative' as mixed reality is today. By now voice processing technology can recognize millions of words from multiple voices even with noisy surroundings. The tasks you can fulfill also extended through the last years. For example, Googles speech recognition solution, called 'Google Voice Search', is used to set alarms, make appointments, call other people, calculate simple mathematical problems and of course search on the greatest search engine of all.
Modern day voice search is based on cloud-based processing. This means that all audio is sent to a server where the central server gives "meaning" to the user's voice. The server makes a best guess what the users said by using the input, personal data and search history. To activate this functionality on Android phones and tablets the only thing you have to say is "Ok Google" and your mobile device starts listening to your commands and search queries. This advanced technology offers you the opportunity to control your Android device without even touching the screen. "Google voice search makes my daily life a lot easier", one could say. However, there is one remarkable thing about Google voice search. Namely, to make it work the microphone on your mobile device has to be active and listening all the time. Otherwise the device cannot detect when you're talking and so it cannot respond. Probably a lot of people are not aware that the microphone on their mobile device is always turned on and so their mobile device is always listening.

If you're interested in listening to your own voice search history, Google provides a platform where all this is stored. It's called the "My activity" platform. When you crawl through your old search memo's you notice some-

thing strange. One should expect that the phone starts recording after the user says the words "Ok Google", however the audio fragment begins a couple seconds before you said those magic words. This means that the phone must be recording all the time too. Actual technical details of this process are not publicly available and so it is not clear what happens with those audio data. It could be a simple audio buffer but more technological advanced solutions are possible too.

The aim of this paper is (1) to explore the technical process behind the Google voice search function on Android mobile devices, and (2) to analyze if it raises privacy concerns Therefore, in the first chapter of this paper we will shortly describe what Google voice search is. After that, the technical functionality behind the voice controlled search application will be examined. This will mostly be done via the Xposed framework[1], which makes it possible to control and scan the phone without using any extra Android Package(APK). Subsequently, a privacy analysis of these technical details is given. Finally, in the last chapter we will draw conclusions.

1.1 Research questions

To answer the question of what happens with this audio data and thereby possibly find any privacy concerns, our research question has been constructed as follows:

What are the privacy issues arising in voice controlled search, with Google's voice search as an example?

Choosing Google as an example has a couple advantages over the other "big sharks" in voice search, Cortana by Microsoft and Siri by Apple. First, Google's smartphone market share is the last three years almost 80% over all smartphone users[2]. This makes Android the most common mobile operating system (OS) in the digital world and ultimately, it's voice application as a example will get the most coverage of users globally. The other two options, Cortana and Siri, are at least as interesting but require for each OS the same amount of extra work, which adds up as approximately three times as much research, for only a 15% to 20% coverage. Second, to answer this research question we need to know which data is gathered at what stage. To know this for sure, we need to make a couple adjustments to the device, to look 'under the hood'. Monitoring this data require several user rights that are usually disabled by the manufacturer. To undo this a device needs to be rooted to 'unlock' these rights for a user. Eventually, all three voice-system

[1]Source: http://repo.xposed.info/ — *visited on November 2016, last checked on June 2018*

[2]Source: https://www.idc.com/promo/smartphone-market-share/os — *visited on January 2018, last checked on June 2018*

options have a way to get administrator rights (called: rooting) in its OS. But as everyone knows, Apple is the most secure and probably the hardest to crack. Windows phone may be easy to root, but it is for certain that the big user base for Android, has done a lot more rooting, and that has resulted in a lot of easy-to-use software for users who lack that technical expert competence.
As already mentioned above, to answer our main question we need to know what data is gathered at which stage. This brings us to the first of our four subquestions, that will give answers and information in this research area:

- Which data is collected at what stage?
- What happens with the generated data?
- What are the legal privacy requirements associated with this data?
- Are there hidden purposes with this audio data?

To answer the first two, the audio-data flow that is generated on the server-side is described in chapter 2. This chapter will scratch the surface of academic research that was needed to tackle the technological difficulties for realizing this new way of communicating with a smart-device. To discover the client-side, measurements can be made on an Android device, to actually see what happens. With the help of some open-source software, microphone usage as well as network traffic are monitored while using the Google application. The methods to do this will be explained in chapter 3. As a side-note: all methods used are only used for academic purposes and should not be used on personal devices. Some methods may break warranty agreements from manufacturer to end-user and require a technical understanding. In chapter 4 the results of the measurements will be analyzed, not only on a technical level but also on a legal level. The analysis is combined with some of the latest legal agreements, such as the General Data Protection Regulation (GDPR), that has been adopted and will be enforced globally after May 2018 by the European Union. This will tackle the third subquestion immediately and will leave us with one more subquestion to be answered. The technical possibilities that can be generated with this audio-data are possibly endless and sometimes too personal for one's comfort. We discuss in the last chapter if there are any hidden purposes that can be revealed, just like the privacy-violation with the Facebook like-button, where hidden trackers gathered data of user's web-browsing behavior, discovered by A. Roosendaal [4]. This will be combined with a conclusion of the discovered results and possible future work for further research.

Chapter 2

Google Voice Search

The basic principle behind voice search is easy to understand. (1) The user says a query to a web enabled device. (2) The device captures its content and sends it to a server. (3) The server processes the query and return the results to the device which (4) gives, by displaying or saying, the answer back to the user. But when we go into depth step 3 requires advanced technological methods to make this possible. The main goal is to give the user the right answer or perform the right action, based on his spoken query, in a appropriate time frame. To achieve this there are a lot of problems to take care of. In the last decades devices became a lot more mobile, this causes a lot of different environments and activities where people can speak from. There could be a lot of background noise e.g. traffic or a busy room where a lot of people are talking. The user also could be driving or biking which also effects in a lot of disturbing sound input. This is one challenge where only the 'appropriate' audio data needs to be filtered out of the audio data stream. Another example is the fast growing number of smartphone users. "By 2018, over a third of the world's population is projected to own a smartphone, an estimated total of almost 2.53 billion smartphone users in the world"[1] . Many more people around the world are getting access to voice enabled services and and therefore many more languages and accents are wanted. Google, and other voice platforms like Siri by Apple and Cortana by Microsoft, need to be prepared to process this new languages and filter out different accents if they want to keep their services available to every user. For Google, one main goal according to researchers is "to make spoken access ubiquitously available" and therefore speech technology through mobile access must accommodate the users needs "any time, any place, any usage scenario, as part of any type of activity" [11]

[1]Source: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/ — *visited on April 2018, last checked on June 2018*

But what does it mean to be ubiquitously available? First of all the main focus of this service must be on "*availability*(i.e., built into every possible interaction where speech input or output can make sense) and *performance* (i.e., works so well that the modality adds no friction to the interaction)"[11]. If this is not achieved by the voice search service, one is just talking to a dead digital assistant. If availability lacks, there is a high probability that the voice service one is talking too is offline and therefore no text processing is done or responsive feedback of the query can be produced. On the other hand, if performance is deficient voice interactions with a device of a user may encounter some disturbance. This could be due to slow feedback from the server-side, where text processing or the connection is too slow to produce responsive feedback which results in a lot of latency in an interaction. Another effect of lacking performance could be that the initial interaction is poorly delivered (a lot of noise, gaps in the audio-stream, large amount of packet loss etc.). This results in unprocessable data, and it is easy to see that the user probably never gets an appropriate answer back. The search query must be said again, and probably another time (the performance is still inadequate here) until the user had enough of using the service, stops speaking and just types it in himself. This is why *availability* and *performance* are essential elements for Google to reach a ubiquity of their speech recognition services.

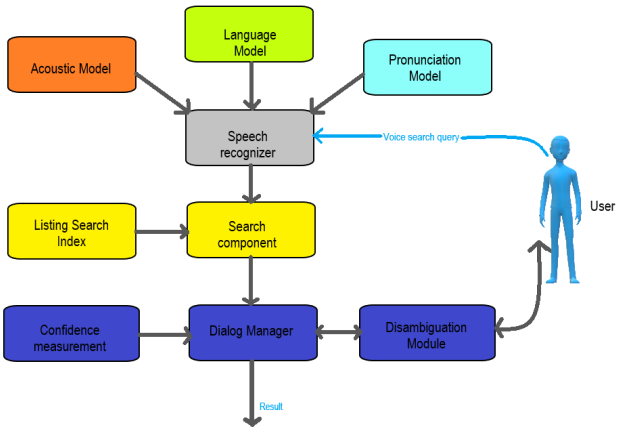

Figure 2.1: Basic architecture of voice search system with user interaction

To understand this more deeply, and why ubiquity is of such importance, we take a look at the server-side architecture and different metrics Google uses in their audio interaction mechanism. As you can see in figure 1 (based on: [12]), the basic architecture of the server-side consist of three main stages. A spoken query will arive at a *Speech recognizer*, Then the top results

will be combined with a *Search component*, which uses a large amount of search query data from the world wide web. The *Dialog manager* will finally decide which result is presented to the user.

2.1 The Speech Recognizer

The first stage of the server-side processing is the speech recognizer. It uses an acoustic, a language and a pronunciation model to analyze the voice input from a user.

There are a lot of obstacles when it comes to speech recognition and the use of these 3 models. For example, when you say "Fill" it could be recognized as "Fill", "Phil" or "Fail" by the recognizer. Another example could be the query "Run to the hills", where Google could respond with the fastest route and time to run to the nearest hills, or just information about a rock song called the same way as the query. Both examples are simple but logical misinterpretations by a speech recognizer and should be avoided by the one developing it. A little more than a decade ago IBM noticed, a error rate of 35% of general speech recognition performance[8]. This means that the other 65% are related to the three models the recognizer uses. To be precise, 22% were errors due to pronunciation and another 31% were noise related failures [8]. So it is very essential to train these three models so the error rate is at its lowest, thereby the user will get the right answer and not some information one did not ask for. So how is such a model trained to get the desired results? To not go in too much detail we take the acoustic model as an example. According to [11] acoustic models used in the speech recognizer "provide an estimate for the likelihood of the observed features in a frame of speech given a particular phonetic context. The features are typically related to measurements of the spectral characteristics of a time-slice of speech". So in simpler words, the language model provides the real textual context of the query, where the acoustic model provides all other characteristics of the query (tonality, volume, speech dis-fluency and frequency). Thereby it models the relationship between the audio signal and the phonetic units in the language.
To test the quality of their acoustic models and improvement, Google has a rather smart metric, called WebScore [11]. Instead of measure the word error rate (WER), the metric is used to give a semantic quality indication of the speech recognizer. It hereby focuses on how many spoken queries are guessed correctly by the recognition hypothesis compared to a human transcription. The better the Recognizer, the higher the WebScore. Google preferably focuses on developing a higher WebScore instead of focusing on WER because it automatically filters out missing functions words like "or", "of" and "in". It also prevents lacking information, and thereby mis-recognition, of a plural form and miss a certain "s" at the end of a word. The semantic

focus will give a lot more info about the whole sentence so the plural form can be decided from there. Closer listening is not needed, because of this. The WebScore of these semantic models, originally based on a combination of hidden Markov Models (HMMs) and Gaussian mixture models (GMMs), recently increased tremendously . This growth is due to new discovery in using deep neural networks for acoustic modeling. According to Geoffrey Hinton & co. [2] the addition of deep neural networks to Google's speech recognition shows a relative reduction of WER with 23%, compared to the best Gaussian based system at that time. This technological breakthrough gave much more accurate models for the speech recognizer to use, for determine a set of results of what the user could have said. This set is passed through to the search component.

2.2 Search Component

The several best results of the speech recognizer are matched with search data to approximate the ideal result. When the speech recognizer has done its job, and what the user said is transformed in usable digital data, the voice search problem basically becomes a normal search problem. The general search mechanisms used in modern information retrieval systems is vector space modeling. According to C. Platzer and S. Dustdar the underlying concept of a Vector Space Model (VSM) "..is quite easy to understand. A document is split up in keywords. Each of this keywords constitutes a dimension in a n-dimensional vector space. Therefore, a document can be seen as a vector within this 'term space'. The position of this vector to other vectors within the same vector space describes their similarity to each other".[9]
To use VSM's for speech recognition a couple problems arise, since speech differs in certain ways for text-based search queries. Speech queries are often much shorter. Most users will probably speak to their device if it is a digital assistant, spoken to with short and simple commands. So a couple enhancements to the general VSM's are made to deal with this speech-text-differentiation to maintain the high accuracy of the IR system [12]. The first enhancement Ye-Yi Wang introduces is focused on the shorter query length of spoken search request, as addressed above. Because little noise in a small query has much more impact on the actual results, the surface term frequency of the query "may not be a reliable estimate of the true underlying distribution." Therefore each duplicate of a word is marked and treated as a different term. For example; the query "8 ball 8" is replaced by "8 ball 8_2nd". The duplicate word is now marked and treated as a actual duplicate. What we mean with *treated* is 8_2nd gives a new dimension of terms for the VSM. It searches now for queries that actual used this duplicate as a purposely intended duplicate. A second enhancement is the

use of a search database and the spoken input to categorize these queries. The query "Jacobs Pizzeria" should list *Pizzeria Jacobs Italiano* much higher than *Jacobs Cyber Security inc.*. The VSM uses "Pizzeria" not only as a term but also as a category to filter the search results for the name "Jacobs". In this example, it is easy to see that categorizing the search query improves accuracy of this vector-based search models.

There is another problem with spoken speech; a lot of words are acoustically confusable. *But, this is already taken care of by the acoustic model used by the speech recognizer, right?* Yes it should, but the usage of VSM's give more opportunity to add acoustic accuracy and robustness to the table. Instead of using words as terms, a word is sliced into different character n-grams, each covering a different part of the original word. Those character n-grams are used as terms by the VSM to counter this acoustically confusable problem. The example of Y Wang describes the purpose of this accurately: "..the listing 'Lime Wire' is rewritten as a sequence of character 4-grams—$Lim Lime ime_ me_W e_Wi _Wir Wire ire$.." (Here "$" indicates the boundaries of the query and a "_" a separation.) "..If a caller's query *Lime Wire* is incorrectly recognized as *Dime Wired*, there is no word overlapping but still much character n-gram overlapping.." [12]. So the character-based approach will more likely still find the correct answer while the word-based VSM cannot.

With the use of Vector Space Models the search component combines the recognized speech results with a lot of search data. This creates new, more fine-tuned results to present to the user. But now the last question is: How does this system communicate with the user in a way the user understands the results in a natural way? Does the system uses text? does it speak back? does it open the navigation application to help the user further? Welcome to the job of *the Dialog Manager*, which handles the last part of the users interaction with a Voice Search application.

2.3 The Dialog Manager

The use of smart phones and computers gave voice technology a wide range of possibilities to interact with the user. The results can differ from just searching the web, calling a friend, making an appointment in the calendar, playing a song, providing the shortest way to a nice restaurant in Google Maps or start your daily work-out routine. But all those possibilities of interaction come with a lot of technical problems. When a user said "Better call Saul", does he want to know information about the so called TV-series or does he just want to call his best friend Saul? Or asked in a more general way, how does the voice search application predict the user's intent?

First, the system has to track if their is any ambiguity in what the user said. The query can be very clearly in just one category e.g. a restaurant

but, queries often match with much more different topics or actions. Fortunately, this can already be tracked in the search component. When a query provides results in different categories a disambiguation process is started in the dialog manager. In early voice applications this was just a simple model that could distinguish names and addresses in a user's sentence. Today, it's much more complex, the ambiguity of a query can cover a wide range of possible categories on the web, in stead of just addresses or names. Also it may use different kinds of functionality on the users device (almost every application on ones phone expands the possibilities to represent the provided answer, such as search results, navigation routes, playing music, a new calendar appointment etc. all require a different application to start). Todays technology enables us to ask the question; is voice the most appropriate way to accumulate human-machine interaction, or are there better solutions, possibly more intuitive ways to make the interaction more fluent? This topic was already a curiosity more than twenty years ago at AT&T research lab. They concluded that design principles to manage dialog for spoken language interfaces (SLIs) are very similar to that of graphical user interfaces(GUIs) [3]. These consist of (1) *Continuous representation* of the interaction and the actions of the system so the user 'knows' whats happening constantly, this provides a natural way of communication for the user, (2) *Rapid, incremental, reversible operations whose operations are immediately visible*, which enable a immediate response that can be corrected if wrong or misinterpreted by the system and (3) the use of *Simple physical actions and labeled buttons* to avoid responses with complex commands for the user as well as for the system[3]

An addition to the three design principles above is also suggested by [3]; "User-specific usage patterns and preferences for system features offer an additional information source for customizing and adapting the interface to these systems." User specific actions are mentioned in this paper a couple times now and is essential to give the user the most fluent, almost 'personal' experience. This includes personal choices and preferences but also other statistics may contribute to the experience of the interaction. An example of such a statistic could be the familiarity with voice technology. The difference between a veteran of voice search (someone who integrated the use of voice enabled search in his daily life) and a user who just discovered there was this new functionality on his phone he never heard about but he certainly could speak to. Figure 2 shows two examples from the researchers at AT&T [3]. Here the difference in response is very clear to see. One can easily imagine that this adaptation in dialog benefits both users in their own way.

Agent: Annie here.
Expert: Get my labels.
Agent: Labels.
Expert: Delete the label for Candy Kamm.
Agent: Deleting Candy Kamm. Labels, what next?

Fig. 11. Expert dialog.

Agent: Annie here. What would you like me to do?
Novice: Get my labels.
Agent: I've got your labels. What would you like me to do?
Novice: Delete a label.
Agent: Say the name or number you want to delete.
Novice: Candy Kamm.
Agent: Are you sure you want to delete Candy Kamm?
Novice: Yes.
Agent: OK, I'll delete Candy Kamm and I'll get right back to you.

Fig. 12. Novice dialog.

Figure 2.2: Difference in dialog between a 'Expert' and 'Novice' user, both examples from [3]

Parallel with AT&T, Google developed its own techniques for tackling the challenges of the speech recognition process. In [11] different user studies from Google are explained. They show various results and patterns in user behavior that help them train and optimize their voice interaction system. Also different methods to predict the users intent are displayed in this article. For their analysis they used 28 days of search data of all the Google mobile app users in the USA and compared it with users 'Non-mobile' desktop behavior. Among several similarities there were some interesting differences. For example, mobile users have much more search queries containing food and drinks locations. The frequency of "WH" questions[2] is doubled with mobile usage. Sensitive content on the other hand, "Categories that consist of adult themes, social networking, and health"[11], are less present in mobile usage compared to desktop behavior.

To counter old fashion 'hard-wired' disambiguation strategies, studies have shown that the ambiguous character of a query can be used as a advancement to predict the user intent [10]. They use ambiguity listings of categories from a processed sentence to make summaries. The constructed summaries will be combined with frequently used attributes, such as geographic location, price and quality. To illustrate its benefit Figure 3 shows the example used by the researchers of this Refiner algorithm. This interaction between user and system uses ambiguous properties of the users input to ultimately give the user their best option based on his current state (in stead of use personal statistics based on previous sessions).

[2]A term Google uses in [11] to summarize "What", "Where", "Who" and "How" questions

User: *Tell me about restaurants in London.*
System: *I know of 596 restaurants in London. All price ranges are represented. Some of the cuisine options are Italian, British, European, and French.*
User: *I'm interested in Chinese food.*
System: *I know of 27 restaurants in London that serve Chinese cuisine. All price ranges are represented. Some are near the Leicester Square tube station.*
User: *How about a cheap one?*
System: *I know of 14 inexpensive restaurants that serve Chinese cuisine. Some are near the Leicester Square tube station. Some are in Soho.*

Figure 2.3: Interaction between user and system, using the Refiner algorithm from [10]

Companies that are improving voice technology systems want to use as much data as possible to improve their systems, and they say they capture it minimally, according to the End User License Agreements. Indeed, this will actually benefit the user's experience with their interactive system. However, does the user want all their data to be captured and used behind closed doors? The problematic down-side of systems using user-specific data is a very well discussed topic nowadays, called Privacy. This paper will dive more deeply into privacy concerns about this technology in chapter 4, where the server-side description above, privacy laws, user agreements and the results of this paper will form a discussion. In the next chapter we put of our Google-glasses and look through the eyes of the user, to describe the method used to discover what happens "really" on a Android device. What can we see and measure from the user-side of Google's voice search functionality? And especially, what happens on the user-side that we don't see?

Chapter 3

Method

In the previous chapter we gave a little insight into how Google voice search works internally and described some of the different challenges for Google to improve this kind of technology. We took a glimpse at this multinational company and its technological development. In this chapter we take back our original research topic, from the user's point of view. Are there privacy issues arising with this advanced audio processing technology that could possibly hurt the user? Which personal data is actually used and when do we know our device is listening?
To answer these questions, information is needed about (1) which data is sent over the network when using this voice enabled application, and (2) when the microphone is activated and used by the same application. This chapter is split up respectively to describe how this information is gathered, together with the frameworks, methods and tools used to monitor these data. This chapter will use a couple of familiar cyber-security methods and terms. We will only scratch the surface of these methods so that a reader with limited IT knowledge would understand .
A disclaimer: These steps are only taken for academical purpose and should not be re-used for any other. We do not want to encourage any of the steps taken below. Some of the methods used are against the manufacturers condition of use and break warranty agreements. So executing these methods is at your own risk.

3.1 Getting root access

As a first experiment, the purpose is to find a way to see which data is sent back and forth between users phone and server. There are multiple ways to capture this information but one thing is certain, it has to be a man-in-the-middle operation to capture its content. The phone used for this experiment is a factory reset Samsung device, updated to the latest versions available at the place and time of the experiment. Here are the specifications of the

device used:

Model:	Samsung SM–G900F (Galaxy S5)
Cpu:	Qualcomm Snapdragon 801 (2.5GHz)
Memory:	2GB
HDD:	12GB
Android version:	6.0.1
Google app ver:	6.4.31.21.arm

In general, Android devices for personal use don't give you root access to your device. It is disabled by default with almost any manufacturer, and they sure have their reasons to. In a market full of users that lack knowledge of the back-end of such devices, it is not advised to give every user full access rights over their phone or tablet. To attain privileged control over a android device, that means; get the same digital rights as an Administrator on a Windows PC, or superuser on UNIX systems, the system needs to be 'rooted'.
Rooting a phone is unfortunately a bit harder then just typing "sudo" in a linux terminal. There are several things to keep in mind; the devices architecture, its software versions and possibly extra restrictions placed by the manufacturer. As it planned out, the internet is full of detailed rooting guidelines for every type of phone, ranging from beginner to expert detail.

What is very important to mention, is to make a full back-up of the device before any of this rooting. The recovery tool for Android used here is TeamWins TWRP recovery. This enables to back-up the phone completely and reset to its default, even if the device will be completely bricked (crashed/freezes completely, literately turn into a brick of metal and glass). This happened immediately in the first try. The back-up was already needed for recovery when the phone bricked installing the required framework on the device. This open-source framework is called Xposed framework. With this software installed on one's phone, a variety of modules can be downloaded that can monitor and alter the behavior of certain applications and the system itself. This includes some functionality that can help us capture network traffic or microphone usage.
At first, the original rom from Samsung blocked the installation of Xposed very quickly and abruptly. After a failed attempt the device created a constant boot loop. A technical phenomena where a device restarts itself before fully starting up, and thereby constantly showing the android starting screen. In short terms: The phone got bricked! The cause of this problem is a security assurance introduced by Samsung called KNOX. A platform installed on every Samsung smartphone since Android version 4.4 that, explained in their own words, "..is Samsung's guarantee of security, and a secure device gives you the freedom to work and play how, where, and when you want. Samsung Knox consists of a highly secure platform built into Samsung devices and a set of solutions that leverage this platform. Whether

you would like to keep your personal photos private, or remotely manage a batch of business smartphones, Knox has you covered." [1] Unfortunately, if you'd like to install an unknown open-source framework on your device, Knox covers you much less. The back-up was used, and also was the only option, to save the phone's functionality from the security measures in place. This also meant Xposed and Samsung's stock rom, with KNOX included, could not be combined to measure the results needed. A custom rom had to be installed on the phone.
Out of a wide range of custom rom variety, an open-source Android operating system developed by Steve Kondik called CyanogenMod[2], stood out. It covers a wide range of possible devices with their corresponding Android version, including the Android 6.0 empowered Samsung device used for this experiment. It gives access to functionality that was hidden by the original manufacturer and with a large number of global users (Forbes estimated around 50 million in April 2015[3]) a large amount of online support, feedback and guidelines comes with. After researching and choosing a compatible custom rom to use, the installation went smoothly. Cyanogen mod together with the Xposed framework and a seperate package including the normally pre-installed Google applications, were all on the device within one hour using the same TeamWin's recovery tool described above. Here is a list of all the software used, with corresponding versions, to achieve having a usable phone for the experiment:

Software:	Version:
Google apps (Gapps)	6.4.31.21.arm
TeamWin TWRP	android_device_samsung_klte
Cyanogen mod	cm-13.0-20151227
Xposed framework	SDK23

3.2 Capturing the conversation

To perform a man-in-the-middle attack on the Android device, a extra Linux device was used to capture all the traffic going in and out of the phone, with a bash tool called `mitmproxy` (all bash commands to realize this are presented in the appendix). But there was just one small unconsidered problem, that still needed to be removed from the equation. Apparently, the Google search app which includes its speech recognition (later referred as the Google app) uses a security measure called certificate or SSL pinning.

[1] Source: Samsungknox.com — *visited on February 2018, last checked on June 2018*

[2] Source: https://web.archive.org/web/20161224194030/https://www.cyanogenmod.org/ — *visited on October 2016, last checked (archive) on June 2018*

[3] Source:https://www.forbes.com/sites/miguelhelft/2015/03/23/meet-cyanogen-the-startup-that-wants-to-steal-android-from-google-2/#26dd93786809 — *visited on October 2016, last checked on June 2018*

In layman's terms: This system can assure if a user or application is communicating to a trusted authority with a trusted certificate. It uses a public key pin code (PKP) that checks if the hostname and the corresponding certificate are genuine, before it enters a certain web domain. Unfortunately, certificate pinning specifically is introduced to make man-in-the-middle attacks much harder. The application almost immediately stopped responding when the capture started because of this. However, the Xposed framework (shown in figure 3.1) already gave a perfectly fitting solution. The module `SSLUnpinning-2.0` uses certain tricks and alters some of the SSL configuration classes in an application to bypass its certificate based security. This way, Google's search application can be used with a different certificate and will still work when another round of capturing is started.

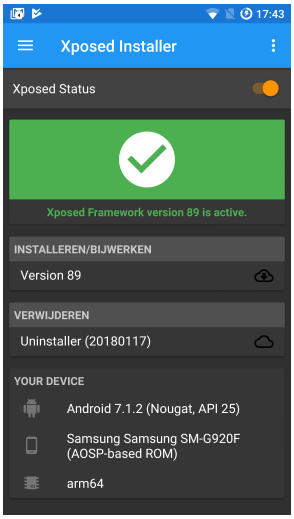

Figure 3.1: Screenshot of a fully installed Xposed framework on an Android device

To define what should be captured different test cases of how the voice search application can be used are composed. First, a set of three long captures with three consecutive search queries are recorded. The captures distinguish themselves in the way of usage. The first two use voice as input, whereas one of them closes the application after usage, to see if there is any request still sent after using Google voice search actively. The last one uses only text queries, to analyze the difference between text and voice over the same application.
Alongside those captures with extensive usage multiple cases with simpler usage are composed that differ in two aspects. (1) The initial state of the device when saying the trigger word "Ok Google". Four different states are considered: The home screen, while using the Google app, while browsing

on the internet and when the device is in stand-by mode. And (2) the length of the search query, using short and long queries to notice any difference in request and response. The short variant consist only of the word "hello". The longer request varied from the sentences; "Is my phone safe enough?", "Is chrome the best internet browser?" and "what is the best car ever?". The reason of this variation is to see if a possible cache is used on the phone to give responses much faster when it is already searched at a prior moment. Finally, extra information is gathered through a couple of unusual test cases. For example, saying "Ok Google" without a following search request, or setting a alarm on "3 O'clock", which requires an action as response in stead of a list of search results. The behavior of the app when it is not used at all, is captured as well to see if data still is flowing without any active usage. All captures with a short description are listed below:

1. `3-requests`
   3 search queries consecutively without closing the Google app.

2. `3-requests-with-closing`
   3 search queries consecutively with closing the Google app at the end of every query.

3. `3-request-without-speech`
   3 search queries using text in stead of speech.

4. `home-active`
   A speech query, with the Google app activated.

5. `home-inactive`
   A speech query, with the Google app deactivated

6. `homescreen-hello`
   A short speech query, with the homescreen as starting point.

7. `homescreen-long`
   A long speech query, with the homescreen as starting point.

8. `homescreen-very-long`
   A very long speech query (couple sentences), with the homescreen as starting point.

9. `browser-hello`
   A short speech query, with the devices browser as starting point.

10. `browser-long`
    A long speech query, with the devices browser as starting point.

11. `standby-hello`
    A short speech query, starting from the devices stand-by mode.

12. `standby-long`
    A long speech query, starting from the devices stand-by mode.

13. `ok-google-only`
    After speaking hotword, nothing has been said.

14. `set-alarm`
    A internal device instruction as input.

15. `random-capture`
    Capturing random activity of the device.

All captured files give different information of how the data flows from the device, via the Google application, to the servers and vice versa. It separates query length, initial state of device, kind of response (an internal action or search results), query type (voice, text or nothing) to give a complete overview where the data is going, what type of data is sent and how it is returned to the device.

3.3 When do you listen to me Google?

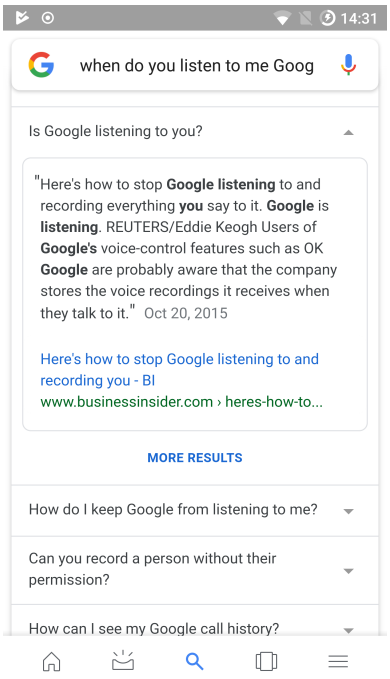

Figure 3.2: The title of this section used as voice input in Google Voice Search

For the second experiment it is required to monitor the microphone activity on the device, to intercept when the Google application is actively listening. To attain this functionality the installation of the Xposed framework, mentioned in chapter 4.1, is required. Because of a different point in time conducting this experiment, other hardware is used. To maintain as much characteristics as possible, the successor of the Galaxy S5, the Samsung Galaxy S6 is prepared according to the same strategy described above, in chapter 3.1. This means, the Xposed framework, a custom rom and the matching Google application package are installed on the device. Keep in mind that some versions will differ with the previous experiment to attain a working condition due to the change in hardware.

Here is the specification list:

Model:	Samsung SM-G920F (Galaxy S6)
Cpu:	ARM Cortex A53 & A57 Octacore (2.1GHz)
Memory:	3GB
HDD:	32GB
Android version:	7.1.2 (Nougat)
Google app ver:	7.1 arm64
Cyanogen mod	LineageOS 14.1
Xposed framework	SDK25 v89

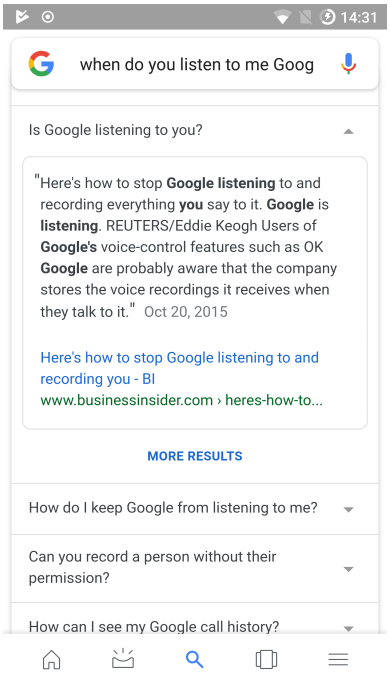

Figure 3.3: The title of this section used as voice input in Google Voice Search

To monitor the microphone usage prior knowledge is needed of which action in Android triggers the device to listen. Xposed has got multiple modules to perform such a task. For instance, the module *App Settings* is a useful tool to search for permissions used by applications and alter their settings. To capture audio, two permissions are qualified: `android.permission.RECORD_AUDIO` (later notated as `RECORD_AUDIO`) and `android.permission.CAPTURE_AUDIO_OUTPUT`. The latter option is not for third party application usage, which means only the built in telephone uses this permission. When the permission filter is activated on `RECORD_AUDIO` all application currently installed are displayed. Several list entries originate from Google's application repertoire including Maps, Keep, Chrome, YouTube, Play services and the target of this experiment `com.google.android.googlequicksearchbox` (this is the general Google search box, which also handles the voice search activity on an Android device).

Another tool, separate from Xposed framework, is used to monitor and debug different characteristics from an Android device. It is frequently used when developing Android applications and a built-in feature in the Android Software Development Kit (SDK), named the Android Debug Bridge (ADB). With a simple installation of the disjunct stand-alone version, ADB works together with the command prompt or terminal on a PC to debug all capable connected devices. If a rooted phone is connected to ADB, only the command `adb shell` opens a Android shell (very similar to a basic UNIX shell) from the root folder of the device. This provides all UNIX functionality is directly applicable on the rooted Galaxy S6, enabling a great variety of monitor tools and access to log files for capturing microphone behavior. The capability to alter different application permissions and settings by the Xposed framework in combination with root access to the device's internal log files provides all information required to determine what happens with the audio data input.

Chapter 4

Analysis

Our analysis is divided in three parts. The results of the experiments described in the previous chapter will cover the first two sections. The last part covers a brief privacy analysis, taking the European GDPR[5][6][7] into account.

4.1 Network

In Chapter 3.2 a brief explanation of how the queries are constructed is presented. All files are captured using the UNIX command line tool `mitmproxy` and can be reviewed, replayed and analyzed by its included companion `mitmdump` [1]. Together, the tools create a powerful web traffic investigation platform for analyzing encrypted captured data with the use of a man-in-the-middle setup. This setup places itself between the acting device and the application security certificate verification. But the full power of `mitmproxy` was not utilized in this case. More security checks from Google where in place. After a search query, wrapped in a html POST request, was sent, some certificate verification errors were sent our way. This means the certificate check is also executed at server-side. This is a good thing from a security point of view, but is blocking further research to get more web-traffic information. Retrieving information of the network data flow when using the Google application is essential. This not only provides a clear view of what audio data is sent to the server. Possibly containing more information than only the search query. It is also required to draw a complete conclusion on our research questions. As future work this experiment can be extended, by finding a way bypassing this verification or finding another method to intercept the raw network data between device and the Google servers.

[1] Source: https://mitmproxy.org/ — *visited on March 2017, last checked on June 2018*

4.2 Microphone experiment

Without prior knowledge, one could reason: "The Google app listens and responds consecutively to my voice and I can use my voice at every moment, so for the application to respond it must be listening at every moment". To discover if this is accurate, we require knowledge of which audio-data is collected at what stage. A combination of the Android Debug Bridge(ADB), a rooted Android device and the Xposed framework, all described in chapter 3.3, is utilized to gather such knowledge. ADB provides root-access to the device's file system and it's internal logging processes. With the familiar bash shell command `grep`, it is very convenient to pinpoint and filter useful information from the device's logging system. Xposed modules are capable to block certain permissions, which then can be used to provoke certain errors in the log. For our research purposes we focus only on the audio related Android permissions, which are:

- `android.permission.RECORD_AUDIO`
  An application can use the microphone to record audio at any time.
- `android.permission.BROADCAST_STICKY`
  An application can send broadcasts that remain on the device, so called sticky broadcasts
- `android.permission.AUDIO_FILE_ACCESS`
  An application has access to recorded messages for internal feedback or analysis of raw audio data

The elimination of these permissions will affect the applications functionality and this will noticeably pass-through to the systems log files. In this way, as much information as possible is gathered to analyze which tools and internal processes handle the audio data flow. A recapitulation of the discoveries during this search is displayed in this section, with only an Android device and some of its developer documentation[2] as source of information. For a more detailed view a section in the appendix is added. Here, a collection of code snippets including the most remarkable discoveries, is shown.
The first notable aspect when looking into the logging files, is that Android is using a mechanism called "audio focus". It is used to manage input and output audio streams between applications on the system. All is managed by an Android media class called `AudioManager`. Every app can either send a request audio focus to the `AudioManager`, which gives them access to the devices speakers, volume, microphone and vibration functionality to do whatever the application is programmed to do, or abandon audio focus which releases this access for other applications to use.
Another remarkable tool an Android system uses is a "hotword detector".

[2] https://developer.android.com/ — *visited on January 2018, last checked on June 2018*

This mechanism, packed with a light-weight speech algorithm, is trained to start an specific action when a pre-programmed hotword is recognized. Google's "Ok Google" and its Apple counterpart "Hey Siri" are a typical and well known examples of these hotwords. At first, it looked like only the Google app was using the hotword detector, but after some more digging, it appears that more applications can insert a hotword with its resulting action and the detector is triggered at the same time as the display of the device. This means that the hotword detection algorithm is always activated when the screen is on, even when the device is locked.
If we take a look at the traditional model of a hotword recognizer, it should behave as illustrated in Figure 4.2. The Google app should start with actually recording one's voice after the hotword detector processed the audio input as an action trigger for starting the application. But in this model of a hotword detector you should think that the trigger word itself is not recorded and only used to start the application.

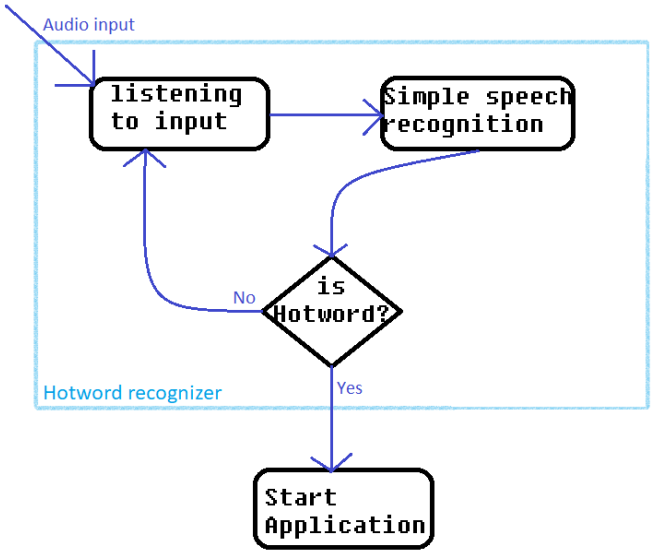

Figure 4.1: simple functional model of a standard hotword recognizer

There is a unnoticed, but very curious fact that contradicts this particular model, which can be found in Google's "My Activity" platform. Here one can overview and manage your search history including his voice search queries. Every prior voice message can be played again. At the beginning of every recording a couple seconds ($\pm$ 3 seconds) including the hotword is played before the users actual search query. Other conversations and environment noises prior the users hotword initiation are also clearly audible.

This means the recording reveals that content from the hotword recognizer ("Ok Google" recording) is saved and in some way combined with the audio data from the search request. Sending audio data was not possible with the model shown in Figure 4.2. There are multiple ways to technically realize this concept but in our view it is most likely one of three scenario's;

1. The processed hotword audio data is sent as parameter/input when the application is started, making it available for the started application, which can save and process this data in its own way. This way, the hotword recognizer(later: H.R.) and application are still seperated.

2. The Google app overrides the standard built-in H.R. with its own alternative. This way all audio data is only gathered by the application, leaving H.R. excluded.

3. Because the application and Android originate from Google, the developers have deep technological access to make both, app and OS, co-operate on a much lower level than third party applications can. This makes new functionality accessible which can optimize the operating systems abilities to the Google applications favor. A small tweak in H.R. can make the audio data available for the application and in local storage.

Which scenario is realized we do not know yet. The information in the ADB did show little signs of audio data saved or sent by the hotword recognizer. The only sign of this, was found when creating an error on purpose. When you stop speaking after the hotword is said, a small message is repeatedly shown:
`pending download info for pending_hotword_model_download_info is null`
`pending download info for pending_xgoogle_hotword_model_download_info is null`
This notion of the `xgoogle_hotword_model` may direct us to scenario 2, where Google is providing its own model, but due to limited research time this cannot be confirmed. Further investigation and low level Android knowledge can provide answers and clarity to this matter. What we do know now is that audio data from "a local small-sized hotword speech recognizer"(which is always on when the devices screen is on), is still capable of sending data to the Google servers together with the actual spoken search query. Here it can be processed and used for their own purposes and new technological solutions.

When the application is started, the hotword recognizer stops and the microphone is activated to record. This is clearly visible in the ABD logs. The entry `HotwordRecognitionRnr: Stopping hotword detection.` is

quickly followed by `MicrophoneInputStream mic_started SR 16000 CC 16 SO 1999`[3]. This way, a lot of different shifts in functionality can be seen. To elaborate the visible process in the ADB with the combination of Xposed modules on the device, we created a timeline (Figure 4.2)

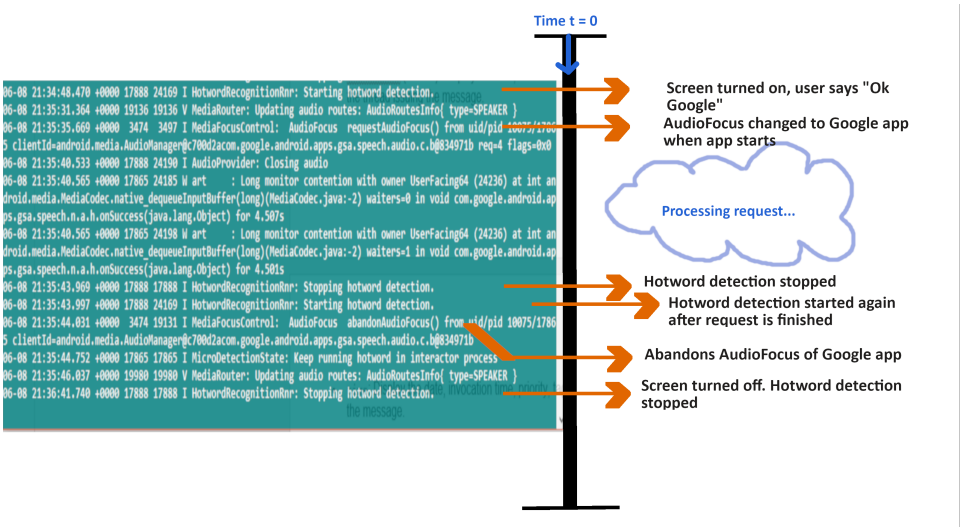

Figure 4.2: Screenshot of ADB console with elaboration

In this figure the log is filtered on the words “speech”, “hotword” and “audio” to see the shifts in `AudioFocus` and the (de-)activation of the hotword recognizer. Because multiple processes are simultaneously logging their status to the same log, not all entries are in the ’right’ chronological order. But the functionality can easily be spotted and analyzed as shown on the right side of figure 4.2. More combinations of filter words and alternation in the devices settings have been considered this way, to provide the results described in this section. Summarizing, the microphone is used by the hotword recognizer when turning on the devices screen, it switches to the Google search application when it is triggered by saying “Ok Google”. Both hotword recognizer and the Google app are capable to record, save and send audio data in some way and at least one of them listens always when the screen is activated. What is actually sent from the device to the server remains unanswered. We continue with a sbrief privacy analysis of the results, by taking the just enforced GDPR (may 2018) in to account

4.3 Privacy Analysis

Recently the parliament of the European Union introduced and approved the General Data Protection Regulation (GDPR). A new impact-full regu-

[3]For the complete log snippets with extra explanation please see the Appendix A.3

lation, enforced after the 25th of may 2018, which ensures transparency in all fields of data gathering and processing. In this research we focus on two elements of this regulation. (1) consent, is it permitted by the user to collect their personal data for a well-defined purpose? And (2), data-minimization. Which means that the collected and processed data should be relevant and limited only to the purpose for which they are processed. Any side activity with this data should be well documented and agreed upon, otherwise a company can face serious charges for violation of the GDPR.
In this section we combine the analysis provided by the prior sections, the user agreement and privacy statement of Google's search application which uses Voice Search and see if this is still in line with the new regulation introduced by the EU. All citations and other external information provided in this chapter is gathered from the latest privacy regulations from Google effective since May 25, 2018[4] and should be in compliance with the GDPR. One particular aspect of the privacy regulation is that most content in Google's privacy policy do not mention audio data in specific. At the section "Information we collect" there is no mention directed to audio data. The only topic that could match the use of this kind of data is called "Local Information", with its accurate description: "We may collect and store information (including personal information) locally on your device using mechanisms such as browser web storage (including HTML 5) and application data caches."[6]. All other probable notions of audio data are all mentioned as "personal information" in Googles privacy document. This generalization of the collected data is not a transparent description and may be opposing with the GDPR.
One specific mention can be found in the additional terms and services of the Google application[5], which says:

[4]Source: https://policies.google.com/privacy/update — *visited on May 2018, last checked on June 2018*

[5]Source: https://support.google.com/websearch — *visited on April 2018, last checked on June 2018*

What's saved in your Voice & Audio Activity
Google records your voice and other audio, plus a few seconds before, when you use audio activations like:

- Saying commands like "Ok Google"
- Tapping the microphone icon

Your audio is saved to your account only when you're signed in and Voice & Audio Activity is turned on. Audio can be saved even when your device is offline.
Note: Not all apps support saving audio to your account.
How Voice & Audio Activity improves your experience
To help you get better results using your voice, Google uses your Voice & Audio Activity to:

- Learn the sound of your voice
- Learn how you say words and phrases
- Recognize when you say "Ok Google"
- Improve speech recognition across Google products that use your voice"

In the first line, the notion "plus a few seconds before..." complies with some of our results in Chapter 4. There is indeed confirmation from Google of the extra audio data that is processed. This is required to mention to the user for giving his consent, looking at the GDPR (article 7).
To test Google voice search compliance with our focus points of the Data Protection Regulation, we need to check two things. The first one is if the user is giving full consent freely to the audio activities of the application, which is explained in article 7[6] and partially covered above. The second focus point is based on data-minimization. Is the collected data "adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed"[5] (GDPR, article 5c). If we take a look at the latter, not much is said of audio data processing in Google's privacy statement. The only specific notion is:
"Voice Search allows you to provide a voice query to a Google search client application on a device instead of typing that query. It uses pattern recognition to transcribe spoken words to written text. We send the utterances to Google servers in order to recognize what was said by you. For each voice query made to Voice Search, we store the language, the country and our system's guess of what was said. We keep utterances to improve our services, including to train the system to better recognize the correct search query if

you have given your consent to such data use." [6]
Here is explained that Google saves the language, country, result of the speech recognition and utterances to improve their services. Which services are improved specifically and how they are improved is only explained partially through some examples. One of the examples with more specific information of how Google uses pattern recognition, is found in the images section:
"If you get a little more advanced, the same pattern recognition technology that powers face detection can help a computer to understand characteristics of the face it has detected. For example, there might be certain patterns that suggest a face is smiling or has its eyes closed. Information like this can be used to help with features like Google Photos' suggestions of movies and other effects created from your photos and videos." [3]
Here, Google informs the users that face detection is also used in a more 'advanced' way, to analyze emotions and other characteristics of one's face. A voice also consist of aspects, such as someones tonality, articulation and volume, that contain much more information about a person than just the interpretation of his search query. If Google applies the same 'advanced' way to its audio data, which is certainly possible with todays audio- and neural network technology, thousands of personal meta-characteristics can be extracted from every voice search query.
This is just a speculation, but gets strengthened by a patent of Google published in 2006[1] where they can extract emotion from natural language in a dynamic way. If there is any implementation of the technology in this more than a decade old patent, it should be properly noted in their privacy agreement according to the GDPR. Let's make the assumption that Google is using their technology in an 'advanced' way to extract more personal data from one's voice, using todays privacy statement (which does not contain any notion of gathering personal data such as emotion). Than there is contradiction with article 5 of the GDPR on two points. First, transparency (article 5a[5]) is lacking. There is no declaration that they use such technology and why this is useful for the users to improve their services. Second, there exists no explanation why this is relevant and limited to what is necessary for achieving Google's purposes, and so it lacks of data-minimization (article 5c[5]). In a lawful perspective this is troubling, but still needs confirmation because this is only a strong presumption and not a fact.
In general, Google's privacy agreement has some vague aspects that do not meet with the level of transparency, described in the GDPR. A lot of it though, is acceptable due to the utilization of *Safeguards and derogations relating to processing for archiving in the public interest, scientific or historical research purposes or statistical purposes*[7] (article 89). Statements like "Learn how you say words and phrases", "Improve speech recognition across

[6]Source: https://policies.google.com/technologies/pattern-recognition

Google products" and "Statistical research to improve and personalize the user experience" can all be seen as research that meets article 89(1). In this article, such data can be processed if it does not permit the "identification of the data subject". This way it is very hard to estimate if there is any real violation of the GDPR.
Concluding, particular information of the processing of users audio data is very limited in Googles privacy statement. There is some special consent for audio data which is required for compliance with article 7. There is a notion that advanced techniques are present for other data such as images and videos, but these are missing for voice and other audio data. If Google uses such technology this may not meet the level of transparency and data-minimization, taking article 5 into account. Because of article 89 it is hard to estimate this matter because additional data processing can be seen and noted as "public interest, scientific or historical research purposes or statistical purposes" by a company. For reaching clarity in this matter, more legal expertize is required.

Chapter 5

Conclusions

In this paper we discussed if privacy issues arise in voice controlled search, taking Google's realization of this technology, *Google Voice Search*, as an example. To give a complete picture we discussed what happens with the generated data on the server side, by looking at academical research of technological challenges and algorithms which make speech recognition possible. Multiple Android-oriented tools were combined to glimpse "under the hood" of a device, to see that the microphone is always active, and capable of recording and sending personal audio data, when the screen is on. This, combined with some strengthened presumptions of using advanced pattern recognition techniques on audio data, raises possible privacy conflicts with article 5 of the GDPR, which is enforced in May 2018.
These are some peculiar discoveries but to make them results, more research is needed. For future work, the network traffic experiment described in chapter 3.2 can be executed with some minor security tweaks to see which audio data is sent from an Android device to Google's servers. This is some valuable missing information in our results. More investigation, low-level Android knowledge and expertise in privacy law can deepen our findings in Chapter 4.2 and 4.3.
Our results may not be final, but our discoveries raise concern of the capabilities of the companies that provide us with their technological services. Companies that use the latest technology and have a huge amount of data can extract a ton of personal characteristics and information from one's voice, which they can use for their own purposes. Most users, even with the updated privacy agreements, are not aware of any such activity happening. With this obliviousness present today, giving consent to "research and improvement of services" still seems unfair and not transparent to us. That is why one final question still remains; is this actually OK, Google?

Bibliography

[1] Ian M Bennett and Palo Alto. US8214214B2 Google patent - Emotion detection device & method for use in distributed systems. *Google Patents (patents.google.com/patent/US8214214B2/)*, 1(19), 2006.

[2] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, and Brian Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *IEEE Signal Processing Magazine*, (November):82–97, 2012.

[3] Candace Kamm, Marilyn Walker, and Lawrence Rabiner. The role of speech processing in human-computer intelligent communication. *Speech Communication*, 23(4):263–278, 1997.

[4] Bert-jaap Koops. Tilburg Law School Legal Studies Research Paper Series - The Internet and its Opportunities for Cybercrime. (09), 2011.

[5] European Parliament and Council of the European Union. Article 5 "Principles relating to processing of personal data". *The EU general data protection regulation 2016/679 (GDPR)*, page 1–88, 2016.

[6] European Parliament and Council of the European Union. Article 7 "Conditions for consent". *The EU general data protection regulation 2016/679 (GDPR)*, page 1–88, 2016.

[7] European Parliament and Council of the European Union. Article 89 "Safeguards and derogations relating to processing for archiving, purposes in the public interest, scientific or historical research purposes or statistical, purposes". *The EU general data protection regulation 2016/679 (GDPR)*, page 1–88, 2016.

[8] Michael Picheny, Yorktown Heights, and Robert Dalrymple. Innovative approaches for large vocabulary name recognition Yuqing Gao , Bhuvana Ramabhadran , Julian Chen , Hakan Erdo g IBM Thomas J . Watson Research Center In vocabulary.

[9] C. Platzer and S. Dustdar. A vector space search engine for Web services. *Third European Conference on Web Services (ECOWS'05)*, 2005.

[10] Joseph Polifroni and Marilyn Walker. An analysis of automatic content selection algorithms for spoken dialogue system summaries. *2006 IEEE ACL Spoken Language Technology Workshop, SLT 2006, Proceedings*, (January):186–189, 2006.

[11] Johan Schalkwyk, Doug Beeferman, Beaufays Francoise, Bill Byrne, Ciprian Chelba, and M Cohen. Google Search by Voice : A case study. *Work*, pages 1–35, 2010.

[12] Ye Yi Wang, Dong Yu, Yun Cheng Ju, and Alex Acero. An introduction to voice search. *IEEE Signal Processing Magazine*, 25(3):29–38, 2008.

Appendix A

Appendix

Here all additional information is displayed. Most of it are log output with key information used in Chapter 4. The in section 4.3 addressed GDPR articles are included here.

A.1 Bash commands prior to man-in-the-middle attack

```
# configure access point
sudo iw dev wlp5s0 interface add wlan0 type managed
sudo hostapd -d /etc/hostapd/hostapd.conf

# in other terminal window
sudo ifconfig wlan0 192.168.0.1 netmask 255.255.255.0
sudo iptables -t nat -A POSTROUTING -o wlp5s0 -j MASQUERADE
sudo udhcpd -f

# to configure mitmproxy
sudo iptables -t nat -A PREROUTING -p tcp --dport 443 -j
    REDIRECT --to-ports 8443
sudo mitmproxy -T -p 8443 -w /tmp/mitm
```

A.2 Network experiment error

This error appeared in the HTML responses captured in the network experiment from Chapter 3. This error is sliced in two parts and shortened (noted as ,.....,) because it was to large to fit in a listing.

```
<< Certificate verification error for b'www.google.nl': hostname
    "b'www.google.nl'" doesn't match either of 'google.com',
    '*.2mdn.net', '*.android.com', '*.appengine.google.com', '*.
    au.doubleclick.net', '*.cc-dt.com', '*.cloud.google.com', '*.
    db833953.google.cn', '*.de.doubleclick.net', '*.doubleclick.
    com', '*.doubleclick.net', '*.fls.doubleclick.net', '*.fr.
    doubleclick.net', '*.g.co', '*.gcp.gvt2.com', '*.google-
    analytics.com', '*.google.ac', '*.google.ad', '*.google.ae',
```

```
'*.google.af', '*.google.ag', '*.google.ai', '*.google.al',
'*.google.am', '*.google.as', '*.google.at', '*.google.az',
'*.google.ba', '*.google.be', '*.google.bf', '*.google.bg',
'*.google.bi', '*.google.bj', '*.google.bs', '*.google.bt
',.....', 'google.kg', 'google.ki', 'google.kz', 'google.la',
'google.li', 'google.lk', 'google.lt', 'google.lu', 'google.
lv', 'google.md', 'google.me', 'google.mg', 'google.mk', '
google.ml', 'google.mn', 'google.ms', 'google.mu', 'google.mv
', 'google.mw', 'google.ne', 'google.ne.jp', 'google.net', '
google.ng', 'google.nl', 'google.no', 'google.nr', 'google.nu
', 'google.off.ai', 'google.pk', 'google.pl', 'google.pn', '
google.ps', 'google.pt', 'google.ro', 'google.rs', 'google.ru
', 'google.rw', 'google.sc', 'google.se', 'google.sh', '
google.si', 'google.sk', 'google.sm', 'google.sn', 'google.so
', 'google.sr', 'google.st', 'google.td', 'google.tel', '
google.tg', 'google.tk', 'google.tl', 'google.tm', 'google.tn
', 'google.to', 'google.tt', 'google.ua', 'google.us', '
google.uz', 'google.vg', 'google.vu', 'google.ws', '
googlecommerce.com', 'gstatic.com', 'source.android.google.cn
', 'urchin.com', 'www.goo.gl', 'youtu.be', 'youtube.com', '
youtubeeducation.com', 'yt.be'
```

A.3 ADB logcat output

Hotword recognizer and microphone

```
HotwordRecognitionRnr: Stopping hotword detection.
04-08 00:27:51.478 13243 27025 I GsaVoiceInteractionSrv:
    HotwordCallback#onError
04-08 00:27:51.478 13243 27025 I HotwordRecognitionRnr: Catched
    InterruptedException
04-08 00:27:51.478 13243 19349 I GsaVoiceInteractionSrv: #
    onClosed - Hotword is not running
04-08 00:27:51.479 13243 19349 I HwDetectorWithState: #
    performNextHotwordAction action: 3
04-08 00:27:51.484 12370 16797 I MicroRecognitionRunner:
    Starting detection.
04-08 00:27:51.486 12370 16817 I MicrophoneInputStream:
    mic_starting  SR : 16000 CC : 16 SO : 1999
04-08 00:27:51.488  3022 16907 I AudioFlinger: AudioFlinger's
    thread 0xf3f831c0 ready to run
04-08 00:27:51.499 12370 16817 I MicrophoneInputStream:
    mic_started  SR : 16000 CC : 16 SO : 1999
```

Hotword error stacktrace phrasecompiler

```
04-08 01:17:14.098 12370 12370 I AudioRouter: Route changed:
    ROUTE_NO_BLUETOOTH->ROUTE_NO_AUDIO, CONNECTION_TYPE_NONE->
    CONNECTION_TYPE_NONE
04-08 01:17:14.109 12370 12370 I MicroDetector: Keeping mic open
    : false
```

```
04-08 01:17:14.110 12370 17982 I DeviceStateChecker:
    DeviceStateChecker cancelled
04-08 01:17:14.110 12370 17966 I AudioController:
    internalShutdown
04-08 01:17:14.111 12370 16904 I MicroRecognitionRunner:
    Stopping hotword detection.
04-08 01:17:14.113 12370 17963 I MicrophoneInputStream:
    mic_close  SR : 16000 CC : 16 SO : 1999
04-08 01:17:14.139 12370 12370 I MicroDetectionState: Keep
    running hotword in interactor process
04-08 01:17:14.140 12370 17958 I MicroRecognitionRunner:
    Detection finished

phrase_compiler.cc:437 Phrase not compiled:
written_phrase {
  weight: 1
  written_word: "Gesproken"
  written_word: "zoekopdr."
  original_phrase: "Gesproken zoekopdr."
}
spoken_phrase {
  spoken_forms {
    variant: "gesproken"
  }
  spoken_forms {
    variant: "zoekopdr."
  }
}

Unknown grapheme: '.'


[ 04-08 01:25:23.341 12370:18156 W/native    ]
phrase_compiler.cc:437 Phrase not compiled:
written_phrase {
  weight: 1
  written_word: "Google+"
  original_phrase: "Google+"
}
spoken_phrase {
  spoken_forms {
    variant: "google+"
    variant: "google positive"
    variant: "google plus"
  }
}
```

Screen on and off with hotword recognizer

```
At startup: 04-10 21:42:14.823  3473  4355 I ActivityManager:
    Start proc 4491:com.google.android.googlequicksearchbox:
    search/u0a75 for service com.google.android.
    googlequicksearchbox/com.google.android.hotword.service.
    HotwordService
```

```
04-10 21:42:17.675  4253  5497 I HotwordRecognitionRnr: Starting
    hotword detection.
04-10 21:42:22.770  4253  4253 I HotwordRecognitionRnr: Stopping
    hotword detection.
04-10 21:42:24.808  4253  5495 I HotwordRecognitionRnr: Starting
    hotword detection.
```

When app did not receive query

```
pending download info for pending_hotword_model_download_info is
    null
pending download info for
   pending_xgoogle_hotword_model_download_info is null
pending download info for pending_hotword_model_download_info is
    null
pending download info for
   pending_xgoogle_hotword_model_download_info is null
pending download info for pending_hotword_model_download_info is
    null
pending download info for
   pending_xgoogle_hotword_model_download_info is null
```

A.4 GDPR related articles

Article 5[5]

Principles relating to processing of personal data

1. Personal data shall be:

(a)

processed lawfully, fairly and in a transparent manner in relation to the data subject ('lawfulness, fairness and transparency');

(b)

collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes; further processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes shall, in accordance with Article 89(1), not be considered to be incompatible with the initial purposes ('purpose limitation');

(c)

adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed ('data minimisation');

(d)

accurate and, where necessary, kept up to date; every reasonable step must be taken to ensure that personal data that are inaccurate, having regard to the purposes for which they are processed, are erased or rectified without delay ('accuracy');

(e)

kept in a form which permits identification of data subjects for no longer than is necessary for the purposes for which the personal data are processed; personal data may be stored for longer periods insofar as the personal data will be processed solely for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes in accordance with Article 89(1) subject to implementation of the appropriate technical and organisational measures required by this Regulation in order to safeguard the rights and freedoms of the data subject ('storage limitation');

(f)

processed in a manner that ensures appropriate security of the personal data, including protection against unauthorised or unlawful processing and against accidental loss, destruction or damage, using appropriate technical or organisational measures ('integrity and confidentiality').

2. The controller shall be responsible for, and be able to demonstrate compliance with, paragraph 1 ('accountability').

Article 7[6]

Conditions for consent

1. Where processing is based on consent, the controller shall be able to demonstrate that the data subject has consented to processing of his or her personal data.

2. If the data subject's consent is given in the context of a written declaration which also concerns other matters, the request for consent shall be presented in a manner which is clearly distinguishable from the other matters, in an intelligible and easily accessible form, using clear and plain language. Any part of such a declaration which constitutes an infringement of this Regulation shall not be binding.

3. The data subject shall have the right to withdraw his or her consent at any time. The withdrawal of consent shall not affect the lawfulness of processing based on consent before its withdrawal. Prior to giving consent, the data subject shall be informed thereof. It shall be as easy to withdraw as to give consent.

4. When assessing whether consent is freely given, utmost account shall be taken of whether, inter alia, the performance of a contract, including the provision of a service, is conditional on consent to the processing of personal data that is not necessary for the performance of that contract.

Article 89 (1)[7]

Safeguards and derogations relating to processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes

1. Processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes, shall be subject to appropriate safeguards, in accordance with this Regulation, for the rights and freedoms of the data subject. Those safeguards shall ensure that technical and organisational measures are in place in particular in order to ensure respect for the principle of data minimisation. Those measures may include pseudonymisation provided that those purposes can be fulfilled in that manner. Where those purposes can be fulfilled by further processing which does not permit or no longer permits the identification of data subjects, those purposes shall be fulfilled in that manner.