Analyzing Third-Party Privacy Policy Violations in Android Health Applications

JAKE GARTH

Bachelor of Engineering (Honours) with the Bachelor of Science

MACQUARIE
University
SYDNEY·AUSTRALIA

Supervisor: Dr Muhammad Ikram

A thesis submitted in fulfilment of
the requirements for the degree of
Bachelor of Engineering (Honours)

School of Engineering
Department of Science and Engineering
Macquarie University
Australia

5 June 2020

Abstract

Mobile devices contain an abundance of private information about their users, making them valuable targets for both legitimate and malicious actors. This private information is often exploited through the use of mobile applications. Mobile applications can collect and share a user's private information through third-party libraries. Users have the option of reading the privacy policy of an application before installing it to determine whether their private information is shared, however, it is not always accurate. This paper presents App Garadyi, a privacy analysis web-platform for Google Play applications. App Garadyi conducts static, dynamic and meta-information analysis to collect important security features and information about a given application. The purpose of App Garadyi is to help users of mobile applications understand the privacy issues present in their applications, and also to aid future research into mobile application privacy. Health applications are particularly prone to privacy risks due to the sensitive nature of the information that they can collect. Because of this, App Garadyi was used to conduct a case study that analysed the privacy and security issues of 5,000 health-related Google Play applications. In particular, this case study investigated privacy policy violations regarding the sharing of user information with third-parties. App Garadyi detected that [x] amount of the applications analysed contained third-party libraries and [x] amount of health applications violated their privacy policy by falsely claiming to not share information with third-parties. These results emphasise the need for Google Play to monitor their applications more strictly.

Acknowledgements

I would like to sincerely acknowledge the effort Dr Muhammad Ikram put in to make this project a success. Without the regular meetings and discussions we had, in conjunction with Dr Muhammad Ikram's expertise in the mobile application security, App Garadyi would not have the value it has now.

I would also like to thank Professor Michael Johnson for giving guidance on how to approach an honours thesis and all the valuable insights that he has provided.

Statement of Candidate

I, Jake Garth, declare that this report, submitted as part of the requirement for the award of Bachelor of Engineering in the School of Engineering, Macquarie University, is entirely my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualification or assessment at any other academic institution.

Glossary

Below is a list of commonly used terms that the reader will need to be familiar with to understand this document. The terms here are either technical terms or terms used to indicate future changes to this document.

Term	Description
[x]	Indicates information that is absent due to this document being an incomplete thesis. This information will be included in the final document.
Note to Marker:	Indicates that the following information is only intended for the marker of Thesis A.
APK	The file type in which Android applications are compiled into. This term is synonymous with "Android application".
API Method / API Method Call	A method within the source code of an application that communicates with a remote server.
Component	Refers to the parts of an application in which a mobile device's system or its user can communicate with.
Case Study	Refers to the case study of 5,000 health applications that this project produced.
Handle	Refers to the unique identifying string of a Google Play application.
Manifest File	A file in every APK that outlines essential information about the application. This information includes permissions, minimum device specifications, components of the application, etc.
Permission	The method in which Android applications (and their third-party libraries) can request user information.
Third-Party Library	Refers to functionality in an application that is not written by the developer of the application.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Mobile devices are incredibly prevalent in the modern world, with over 14 billion mobile phones and tablets forecast to exist in 2020 [1]. Mobile devices contain a plethora of information about their users such as location, photos, text messages, web history, and much more. Mobile applications often exploit this private information about its users, both for malicious intentions and for legitimate reasons. There is additional risk for health-related applications as they often collect private medical information from its users. In addition to this, health applications are very common, with over 50% of American mobile phone users having at least one health related application installed on their mobile device [2]. Users of Google Play applications have the option of looking at the privacy policy of the mobile application to see how their information is used, however, privacy policies can't always be trusted as they don't always accurately reflect the behaviour of the application [3]. This paper aims to investigate the disconnect between the sharing of data that mobile health applications collect and their privacy policies. Specifically, the research question this thesis solves is: "What portion of health-related Google Play applications share private user information with third-party libraries and not declare it in their privacy policy?".

To comprehensively analyse the sharing of private user information with third-party libraries, a publicly available web-service was built. This web-service, named App Garadyi, was used to collect 5,000 mobile health applications from Google Play and perform static, dynamic and meta-information analysis to determine whether or not applications are sharing user information with third-party libraries without declaring it in their privacy policy.

1.1 Motivation

The motivation of answering the research question and building App Garadyi is threefold:

(1) To provide an understanding of how Android Applications in the health category are collecting user's information and sharing this information to third-parties;
(2) To determine what portion of health applications are misleading users about the use of their private information; and
(3) To aid future empirical research into mobile application security by minimising the time that researchers must spend on data collection and manual static and dynamic analysis of mobile applications.

1.2 Project Scope

This section discusses what is and what is not part of this projects scope. This includes the type of applications being analysed, the methods of analysis and the scope of the privacy policy analysis.

1.2.1 Types of Applications Used in Case Study

Google Play has the largest amount of available applications as of Q4 2019 [4], as a result, App Garadyi will conduct its privacy analysis exclusively on Google Play applications. This is opposed to iOS applications and other Android application stores. In addition to this restriction, only applications that have been assigned to the "Health and Fitness" category of Google Play are analysed. Further criteria can be found in the Methodology chapter in this document.

1.2.2 Analysis Methodology

App Garadyi will specifically focus on the static, dynamic and meta-information analysis of mobile applications. Forensic analysis of the mobile device, taint analysis and other forms of analysis are out of the scope of this project.

The information that App Garadyi will collect and use in its privacy analysis is restricted too:

(1) Static Analysis
   - Permissions
   - Third-Party Libraries
   - Certificate
   - VirusTotal Verdict

(2) Dynamic Analysis
   - Network Traffic

(3) Meta-Information
   - Privacy Policy
   - Rating
   - Description
   - Number of Installations
   - Hash of the Application

App Garadyi will then present these results for a given application on its web-page and make them downloadable in a researcher friendly format.

1.2.3 Privacy Policy Analysis

Privacy policy analysis is limited to determining whether it claims to be sharing user data to third-parties or not. It does not include determining what information the privacy policy claims to be sharing or using.

1.3 Deliverable

There are two deliverable items in this thesis:

- Conducting a case study to answer the research question; and
- Developing a publicly available privacy analysis tool for Google Play applications.

1.4 Thesis Structure

This section outlines the structure of the rest of this document. Chapter 2 is the literature review which investigates existing security analysis methodologies of mobile applications and their privacy policies. Chapter 3 is the plan for how App Garadyi and the case study was conducted. Chapter 4 contains the requirements and design of App Garadyi. Chapter 5 is the methodology for building App Garadyi and conducting the case study research into the 5,000 health applications. Chapter 6 contains the results of the of the case study research and the success of App Garadyi. Chapter 7 is a discussion of the results presented in Chapter 6. Chapter 8 contains the conclusion and future works.

CHAPTER 2

Literature Review

In this literature review, we present research on privacy issues most relevant to the research question and the development of App Garadyi. This includes:

(1) Privacy research into health-related mobile applications and their exposure to sensitive information;
(2) Methodologies in which mobile applications have been analysed for security issues; and
(3) Tools built to detect privacy policy violations.

Health-related mobile applications are often trusted with medical and other private information. Thus, it is particularly important to ensure privacy and security of health-related applications in comparison to other genres. This literature review explores the security and privacy issues within health-related mobile applications, and how they can potentially expose user's sensitive information to both malicious and legitimate entities. The ways in which mobile applications can be assessed for privacy concerns is also discussed, providing insights into appropriate ways that App Garadyi can discover the privacy and security features of an application. Finally, we provide a review of how previous privacy policy analysis tools have been built, with the intention of learning about the most relevant practices for detecting privacy policy violations.

2.1 Privacy Research into Health Applications

Mobile applications often collect and share information about its users. Health applications often present additional privacy risk to users as they often collect medical information in addition other private information. In this section, we explore the existing privacy research that has been conducted into mobile health applications and their privacy policies.

2.1.1 Mobile Health Applications Privacy Concerns

Mobile health applications have been shown to share information with third-parties and vendor in unsecured ways. It has been demonstrated that they do not necessarily follow good security practices such as encrypting user information or observe the principle of least privilege.

Mobile health applications have been found to typically share user information with both the vendor of the application and third-parties. [5] found that 80% of health applications would communicate with third-parties for analytics and advertisement purposes, and 90% would communicate with third-parties and/or the applications vendor. A study investigating the privacy of the most installed health applications [6] found that 40% of the applications analysed would send user's private health information to third-party domains and 80% send this information to the vendor. This same study found that half of these applications were using HTTP, not HTTPS, to transmit private health information to both third-parties and the vendor. As a result, these applications are vulnerable to man-in-the-middle attacks. This vulnerability within health related mobile applications has been further emphasised in [7], [8] and [5], where approximately half of health related applications analysed were not encrypting information being transmitted. This is contradictory to more recent research [9] which analysed 24 medicine related applications and found that 94% of detected transmissions were encrypted and only 18% of applications that transmitted messages sent plain-text message.

Furthermore, [6] discovered that the majority of health applications analysed were unjustifiably over-permissioned. Permissions enable applications (and the third-party libraries installed in an application) to access sensitive information about its user. The over-use of

permissions creates security issues for non-medical private information on the mobile device. The most common dangerous permissions detected in this case study included reading and writing to the external storage of the mobile device and also accessing the user's location. This study found that dangerous permissions implemented in the analysed applications could have been avoided with better design or were completely unnecessary. This conclusion was also reached in more recent research [9], which found that 19 out of the 24 medicine-related applications were requesting to read or write to the mobile devices external memory. This access to private information, coupled with the significant amount of data being transmitted, which is often in plain-text, leads to a significant security and privacy risks for health application users.

Ultimately, these studies demonstrate how prevalent sharing private user information is amongst health applications. However, it should be noted that most of these relevant case studies into health applications are approximately half a decade old, meaning that more proximate research should be conducted to assess the privacy risks in mobile health applications.

2.1.2 Mobile Health Application Privacy Policy Research

There is existing research with similar goals to this thesis, that demonstrate there are often inconsistencies between a mobile health application's behaviour and its privacy policy. However, the most recent forms of research in this area tend to be a few years old. Also, the existing research into this topic is on a much smaller scale than this paper has proposed due to their methodologies being manual. But first, we present the essential background information about Google Play privacy policies.

2.1.2.1 Privacy Policy Background

Since 2017, Google Play requires its application developers to provide a privacy policy in the event that their application handles or accesses users private data in any way [6]. This includes the sharing of user data to third-parties. The privacy policy of an application that does access or share user information must explicitly disclose the reasons why it does so.

For example, valid reasons include advertising or tailoring the application to match its user's demographic. Google Play also declares that applications are not allowed to use permissions for the sole purpose of data collection. User information gathered by permissions may only be collected or shared if the application has a functional purpose for doing so. To ensure it is easy for users to see privacy policies, an applications developer is required to upload the privacy policy to the applications page on the Google Play website and also be stored in the application itself [10].

Despite the standard set by Google Play for developers, privacy policies may not be accurate. For example, Care19, a corona-virus contact tracing application was found to be sharing its users location data to a third-party, despite its privacy policy claiming users data was to be kept private [11]. Privacy policy violations can lead to significant fines and legal action, such as when Path, a social networking application, collected user information without the users consent by not correctly mentioning what information the application was collecting in their privacy policy. This led to a fine of $800,000 USD for the company [12].

2.1.2.2 Mobile Health Application Privacy Policy Research

There is limited research into the privacy policies of mobile health applications. However, the existing research, despite it being outdated, indicates it is not uncommon for mobile health applications to violate their privacy policy.

Very little relevant research could be found that investigated the privacy policies of health applications. However, [13] found that there are gross privacy policy violations amongst health applications in both iOS and Android (before Google Play ensured its applications presented privacy policies). It found that only 4% of health applications data-handling behaviour matched their privacy policy exactly. This study also found that 25% of applications privacy policies sent analytical data without informing users. However, this study is limited due to it being before privacy policies were enforced by Google Play and its sample size of 49 applications. Another study assessed privacy policies of prominent health applications in 2017 and 2018 [14]. However, it did not perform static or dynamic analysis to verify the legitimacy of the privacy policy. This study found that the vast majority of health application

privacy policies were claiming the sharing of user information with third-parties. However, this study is limited due to it only analysing a total of 117 prominent health applications. This study excludes less-popular applications, which may explain why these privacy policies were so eager to announce their privacy issues.

It is clear that more research should be done in this area, due to the lack of volume of applications analysed and the lack of recent publications.

2.2 Privacy Analysis Methodologies

This section discusses the most common Android application security research methodologies. Specifically, this section emphasises the importance of static analysis, dynamic analysis, hybrid analysis and meta-information analysis when detecting privacy issues in mobile applications. These four methodologies for analysing Android applications are worth discussing due to their overwhelming prevalence in existing security analysis tools. Explanations of these methodologies are given, and tools are presented that perform security analysis that are most relevant to App Garadyi. Finally, an investigation has been conducted to compare the current online Android application security analysis market, with the intention of demonstrating the differentiating features of App Garadyi.

2.2.1 Static Analysis

Static analysis involves using the source code of a program as an input into an algorithm with the intention of achieving a conclusion about that program. Static analysis of Android applications are typically performed to uncover security and privacy issues within the application [15], this is opposed to detecting non-security related features about the application. As a result of this interest in statically analyzing Android Applications for security purposes, static analysis tools have been created and used successfully for mobile applications previously. The most relevant employments of static analysis include user information sharing, permissions analysis and certificate analysis.

2.2.1.1 User Information Sharing

Static analysis can be used to detect applications that transmit information to external servers. This goal of tracing what information could be sent to external entities was solved by AndroidLeaks [16]. AndroidLeaks successfully does this by mapping sources of private information to an API call method. It does this mapping through dataflow analysis, where the source of the information is traced to a sink. This tool has the ability to find leaks to both third-parties and the vendor.

AndroidLeaks does not detect what third-party libraries are installed in an application despite it being possible. A study has used static analysis [17] to identify tracking libraries in applications and present their security risks and privacy leaks. Identifying each tracker was achieved by inspecting the class hierarchy of a given application as a means of identifying known tracking libraries. In addition to this, the information that the tracker was collecting was unearthed by determining where privacy invasive API calls were located. If these API calls were located within a tracker that was installed in the application, it is clear that the tracker is collecting this information. The goal of [17] was to analyse the tracking libraries in the most popular applications in the Google Play market, however, this methodology could be adopted to identify third-party information sharing more generally.

2.2.1.2 Permissions Analysis

In addition to discovering the privacy concerns surrounding sharing user information, static analysis has also been used to predict if an Android application is malicious through its use of permissions. APK Auditor [18] used a statistical method for detecting malicious Android applications. APK Auditor will assign a score to each possible permission that an application can request. A high score represents that a permission is very common among malicious application. If the sum of an applications permission scores breaches a threshold, the application is deemed to be malicious. FAMOUS [19] extends this notion that permissions can be used to predict whether an application is malicious or benign by recommending the use of machine learning techniques. Similar to APK Auditor, FAMOUS gives a weighted score to each permission based on its frequency in malicious and benign applications. FAMOUS

then used machine learning techniques to build a random forest classifier based on a data-set of malicious and benign applications. Both tools were successful and achieved very high accuracy. Demonstrating that permissions are not only useful for identifying what information an application access but also as a means of predicting the maliciousness of an application.

2.2.1.3 Digital Certificate

Furthermore, static analysis can also be used to verify the digital certificates within an Android application. Every application on Google Play must contain a digital certificate. STAMBA [20], a tool that was created to perform static analysis on banking applications, used OpenSSL [21] to verify that an Android applications certificate was legitimate. Specifically, verifying a digital certificate ensures that the public key used to encrypt data belongs to the correct entity. This is important as it ensures that data being encrypted can only be decrypted by the appropriate entities.

2.2.2 Dynamic Analysis

Dynamic analysis involves the testing of a program during run-time with the intention of achieving a conclusion about the program. Dynamic analysis has been successful in detecting security issues in Android applications. It is often coupled with machine learning algorithms due to the complexity of analysing applications behaviour. Methodologies for using dynamic analysis to detect suspicious network traffic is explored in this subsection due to the relevance it has with App Garadyi.

2.2.2.1 Dynamic Analysis of Network Traffic

Mobile-Sandbox [22] analysed the PCAP files containing network traffic of a mobile device during run time. Mobile-Sandbox assessed whether the hostnames the device communicated with has been reported as malicious and whether data the application is sending is private. This tool also monitors the logs of the device to record method calls in the application and used this information to build a machine learning classifier for malicious and benign applications.

Similarly, GranDroid [23] used dynamic analysis on mobile applications to identify malicious network behaviours. This was conducted by building graphs that represent the method calls, classes and class loaders of the application during run time. Then, key features of the graphs were extracted and were fed through various machine learning algorithms to achieve an accuracy of 93%. Dynamic analysis has also been used to build a machine learning algorithm that will determine if an application is benign or malicious based on the mobile device's HTTP requests during run time [24]. This approach achieved a 99% accuracy when detecting new malicious applications.

It is clear that dynamic analysis is a valuable tool in analysing the security properties of mobile applications, however, it is often necessary to resort to machine learning to automate this process due to the overwhelming amount of information it often produces.

2.2.3 Hybrid Analysis

Hybrid analysis refers to the use of both static and dynamic analysis, often with the intention of countering the flaws of static or dynamic analysis. In this subsection, the flaws of static and dynamic analysis are presented as well as techniques used to overcome these flaws.

2.2.3.1 Limits of Static Analysis

Despite the usefulness of static analysis, there are limitations to exclusively using static analysis for detecting privacy issues. For example, code obfuscation techniques are often used in Android applications for legitimate reasons such as protecting intellectual property and malicious reasons such as hiding malware [25]. Software with code obfuscation has been proven to ensure that malicious code can go undetected in anti-virus tools [26], demonstrating that static analysis alone may not be sufficient to detect malicious software.

2.2.3.2 Limits of Dynamic Analysis

Dynamic analysis techniques that have been applied to detect malware in Android applications typically have better precision when compared with static analysis [27]. However, malicious

code will only be detected if it is being run during testing. This is contrary to static analysis which can analyse the entire source code of an application to detect malicious intent. In addition to this, dynamic analysis is typically much more resource and time consuming than static analysis as it must execute the code on a device.

Furthermore, it has been discovered that many dynamic analysis Android malware detection tools can be made redundant as they use Android emulators, oppose to real mobile devices [28]. Malicious applications can be designed to detect when it is being run on an emulator or a physical mobile device and then ensure it does not conduct malicious activity while being run on an emulator [28]. However, anti-analysis methods, such as emulator detection, can be discovered by Android analysis tools. Droid-AntiRM [29] has been designed to find methods in the source code of an application that detects if the device is an emulator. Droid-AntiRM will then tame this method so it no longer interferes with the dynamic analysis of the application.

2.2.3.3 Mitigating Limitations

Dynamic analysis is necessary to overcome the flaws of static analysis, as it can track what instructions/messages are being sent to and from the software, rendering code obfuscation ineffective at hiding malicious code [26]. Thus, it is often important to include dynamic analysis to ensure the validity of a static analysis tool.

The drawback of dynamic analysis being much more time and resource consuming than static analysis can be mitigated by only conducting dynamic analysis if static analysis does not detect any malicious code [30]. The benefit of this is that if static analysis has already deemed an application to be malicious, there is no need to waste more time and resources. In addition to this, conducting static analysis will assist in countering any emulator detection methods within an application.

To conclude this section, it is important to combine both static and dynamic analysis as a means of complimenting their strengths and weaknesses.

2.2.4 Meta-Information Analysis

In comparison to static and dynamic analysis methodologies, there appears to be a lack of interest in collecting meta-information about a mobile application as a method for detecting malicious intent. By doing this, static and dynamic analysis are ignoring the user experience (e.g. reviews, ratings, and downloads) and information about the developer of the application. However, meta-information has been proven to be successful at detecting malicious intent in mobile applications.

There have been a few application security analysis tools developed that use meta-information analysis to achieve strong results detecting malware in Android Applications. Meta-information, such as the rating, description and developer of an application has been used in ANDROIT. ANDROIT is a tool that predicts whether a given application is malicious and achieved a 93.7% accuracy [31]. ANDROIT was able to achieve such results by create a database of benign and malicious applications that it tested through VirusTotal. Then it trained classification machine learning algorithms on key pieces of meta-information. This meta-information includes country of origin, user rating, user installs, etc. In addition to this meta-information collection, ANDROIT used text mining to collect terms that would often be used by either malicious or benign applications in the meta-information it collected. This information is used in the classification stage of the machine learning process. The success of ANDROIT is not surprising considering that previous research has concluded that machine learning can be applied to meta-information as a means of detecting malicious applications. Similar to ANDROIT, [32] built a database of malicious and benign Android applications using VirusTotal and then used this database to discover if meta-information can be used as an indicator of malicious intent. The results of this study showed that particular meta-information does indeed have significant predictive power in whether an application is malicious or benign. The study found that developer information, certificate information and category of the application are the most powerful meta-information for predicting if an application is malicious.

These studies demonstrate the usefulness of meta-information as a means of predicting whether an application is malicious. However, the limitation of meta-information analysis is

	VirusTotal [33]	Sisik [34]	SandDroid [35]	Exodus [36]	App Garadyi
Permission Analysis	Yes	Yes	Yes	Yes	Yes
Certificate Information	Yes	Yes	Yes	Yes	Yes
Verification of Source Code	No	No	No	No	Yes
Third-Party Libraries	No	No	No	Yes	Yes
Anti-Virus Detection	Yes	No	Yes - Through VirusTotal	No	Yes - Through VirusTotal
Meta-Information Analysis	No	No	No	No	Yes
Developer Information	No	No	No	No	Yes
Provides Network Information	Yes	No	Yes	No	Yes
Privacy Policy Analysis	No	No	No	No	Yes
Results Provided in a Data Ready Format	No	No	Yes	No	Yes

TABLE 2.1: Comparison of Online Android Application Analysis Tools

that it is based on statistics - it can not definitively determine if an application is malicious or benign in the way that static or dynamic analysis can.

2.2.5 The Current Online Android Application Security Analysis Market

There are already online and freely available tools that will perform static and/or dynamic analysis on Android applications [33] [34] [35] [36]. The key differentiating factor between App Garadyi and other online Android application security analysis tools is that App Garadyi makes a verdict on whether an application is violating its user information sharing policy. No other available Android application analysis tools perform this kind of privacy analysis.

A concise representation of the features in the current online APK security analysis market has been outlined in Table 2.1.

2.3 Automated Privacy Policy Tools

Successful privacy policy violation detection tools have been made. Privacy policy violation detection tools and frameworks typically include some form of static analysis to evaluate the behaviour of an application. This information is then compared against the privacy policy of the application using natural language processing techniques. In this section, the methodologies that these tools used are discussed.

2.3.1 Privacy Policy Violation Detection Tools Methodology

Slavin et al. proposed a semi-automated framework that detects privacy policy violations in Android applications [37]. Specifically, this framework identifies if a privacy policy incorrectly claims what information was being collected by the application. This framework would first identify and annotate common API method calls with low-level technical terminology. These API methods would then be mapped to common privacy policy phrases based on their technical terminology. Now that this mapping is complete, static analysis can be conducted on an application to detect whether its API methods are sending information to a remote server. If the static analysis detects information is being sent to a remote server, the framework will search for the corresponding phrase in the privacy policy. If this phrase does not exist in the privacy policy, a violation has been detected. PPChecker [38] improved upon this framework by automating the privacy policy phrase making process through natural language processing techniques. In addition to this, it also uses a database of class names commonly found in third-party libraries installed on mobile applications as a means of detecting information sharing. This framework presented by Slavin et al. as well as PPChecker do not consider the sharing of user-input information. User-input information refers to information that a user manually shares with the application. GUILeak [39] resolves this issue by using static analysis to trace information sent from API method calls to GUI components in the Android application.

Slavin's framework and GUILeak use manually designed natural language processing techniques to analyse a privacy policy. This is time consuming and results in detailed algorithms needing to be written to automate privacy policy analysis. This is contrary to privacy policy research by Zimmeck [40] and PPChecker. Zimmeck's research used supervised machine learning techniques on a corpus of 115 privacy policies to classify privacy policies on what information they claimed to be sharing. This study successfully analysed 9,050 mobile applications with a privacy policy. It determined that 41% of these applications were collecting information and 17% were sharing user information with third-parties without disclosing it in their privacy policy. The success of this study demonstrates that machine learning analysis

of privacy policies is practical. However, by the very nature of machine learning, there are bound to be errors within the results.

In an attempt to stop accidental privacy policy violations, a tool has been developed to automate privacy policy generation. Applying a similar method to privacy policy violation detection tools, AutoPPG [41] uses static analysis to determine what should be written in a privacy policy. Then it uses natural language processing to automatically write a privacy policy. Like PPChecker, AutoPPG analyses the API methods to map them to private information and determines if these API methods are part of third-party libraries. This is determined by whether the name of an API method is the same as a third-party library API method name that has been previously identified. Once AutoPPG has determined what private information is being collected or shared to third-parties, it applies natural language processing techniques to generate privacy policy sentences that accurately reflect the behaviour of the application.

To conclude this subsection, Table 2.2 displays the methodologies of each of these tools. It is clear from the research that static analysis combined with natural language processing techniques or machine learning has proven to be successful at detecting privacy policy violations. However, it is unknown and untested whether dynamic analysis tools can be used effectively in detecting applications that violate their privacy policies. It is also worth noting that none of these tools are readily available online, making it difficult for consumers or developers to identify any privacy policy violations. There are other tools that assist in generating a privacy policy for Android applications, such as [42] [43], however, none analyse the source code. They merely ask developers to manually enter information into a template.

2.4 Conclusion

After assessing the current research into the security and privacy of mobile health applications, it is clear that more research should be conducted. In particular, contrasting the sharing of private user information with mobile health application privacy policies has not been executed recently or on a mass scale. This is important as incorrect privacy policies can present privacy issues to the users and legal consequences for the vendors of an application.

Tool Name	Purpose	Application Analysis Method	Policy Analysis Method
GUILeak [39]	Detects privacy policies that do not disclose use of user input information	Detects API method calls that access user input data	Natural language processing techniques to map user input information to key phrases typically found in privacy policies
Slavin's Framework [37]	Detects privacy policies that do not correctly identify what permission-based information they collect	Detects API method calls and annotates them with technical terminology	Maps annotations to relevant phrases found in privacy policies
PPChecker [38]	Detects imprecise privacy policies, inconsistent privacy policies and incorrect privacy policies	Traces API method calls also compares API method call names with known third-party API call names	Machine learning classifiers are used to identify what information is claimed to be collected
Zimmeck's Research [40]	Conducted an analysis of 18,000 applications to find applications that did not disclose user information collection	Traces API method calls to user information	Supervised machine learning to detect violations of user information collection
TapVerifier [44]	Determines whether privacy policies can be used to detect whether the applications matches its description	Traces API method calls to user information	Uses natural language processing and information extraction techniques

TABLE 2.2: Comparison of Privacy Policy Tools

There are existing static and dynamic analysis techniques that are suitable for detecting information sharing with third-parties. In addition to this, existing privacy policy violation tools tend to use static analysis as a means to detect user information sharing. These tools also use natural language processing or machine learning algorithms to automate the detection of privacy policy violations. As these tools have been successfully implemented previously, it would be wise to follow similar methodologies to answer the research question of this thesis.

CHAPTER 3

Plan

In this chapter, I briefly discuss the planning and progress I have made so far. This chapter will be removed in the Thesis B submission, as this chapter's purpose is to explicitly demonstrate adequate planning and progress for the purposes of Thesis A marking.

My planning has involved writing the essential requirements for App Garadyi and the methodology in which I plan on meeting these requirements. These can be seen in Chapters 4 and 5.

I have made some progress building App Garadyi, which can be seen in my Github repository and a video demonstration. The links to these can be found in Table 3.1.

	Link
Github	https://github.com/JakeGarth/django/
Video	https://youtu.be/tQHWLHgxanA

TABLE 3.1: Links to App Garadyi Progress

I have so far completed building the framework for App Garadyi and the static analysis component of App Garadyi. What "static analysis" means in this context is discussed in the methodology chapter. A gantt chart has been made, and can be seen on the next page, which depicts how I plan on spending the next few months working on my thesis.

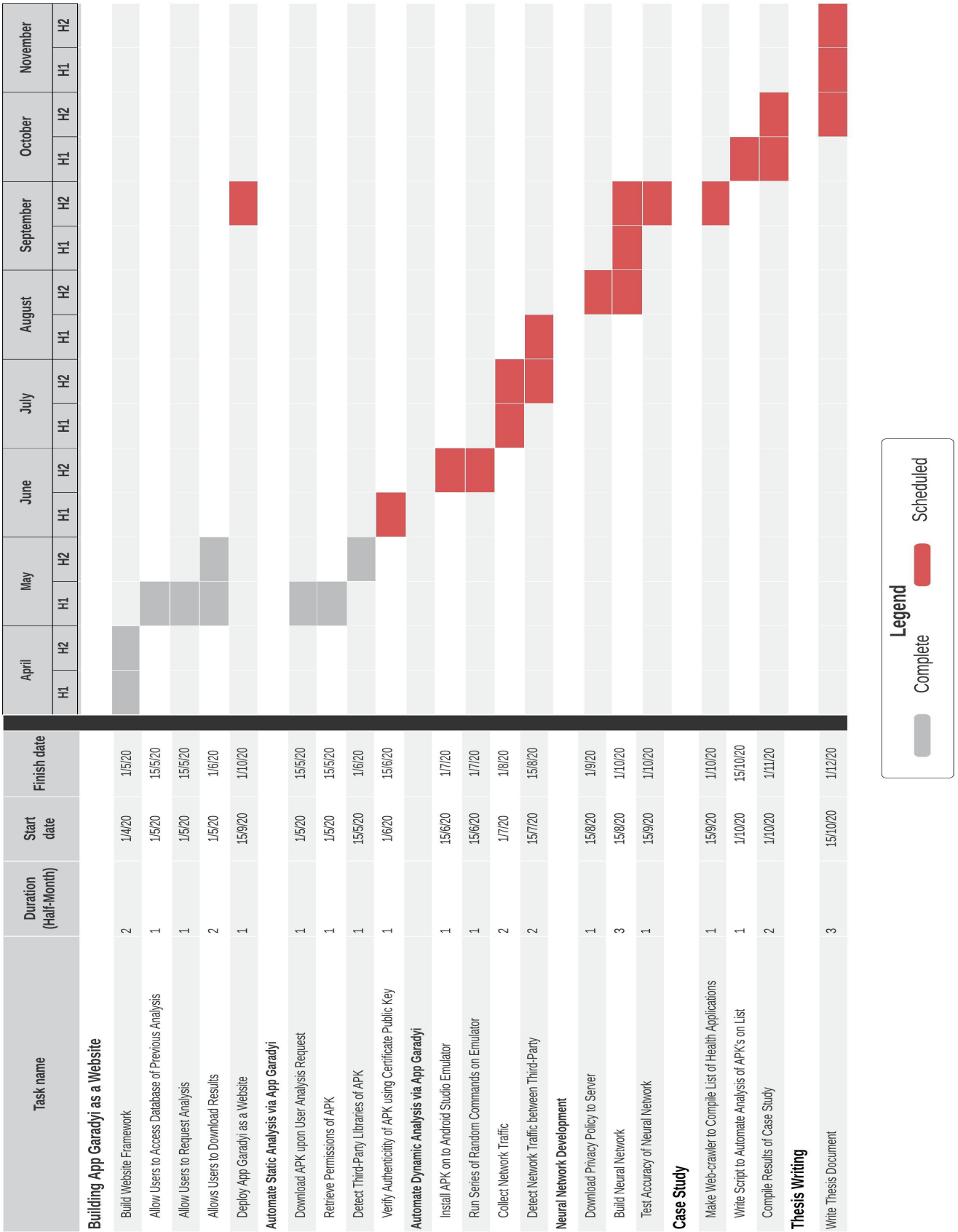

FIGURE 3.1: Thesis Plan

CHAPTER 4

Requirements and Design

The purpose of this chapter is to outline the requirements that App Garadyi will need to fulfill for it to assist in answering the research question of this project. In addition to this, a site map and a sequence diagram are displayed to visually represent App Garadyi. Requirements are written in the following format: A description of the requirement is given and additional context/rationale is given below.

4.1 Functional Requirements

The functional requirements for App Garadyi have been broken up into the following subsections:

- Accessibility and Usability
- Static Analysis
- Dynamic Analysis
- Meta-Information Analysis
- Privacy Policy Analysis
- Detection of Incorrect Privacy Policy

4.1.1 Accessibility and Usability

This subsection covers how users can access App Garadyi and how it can be interacted with from the user's perspective.

FR 1.1. App Garadyi is a publicly available website and can be used to its full functionality via a web-browser.

To ensure that App Garadyi can be used by other researchers, and not just for the single case study presented in this paper, App Garadyi should be made easily accessible. To this end, App Garadyi has been made available as an online web-service.

FR 1.2. Users can analyse a Google Play application by entering the handle of the application into App Garadyi.

The "handle" of a Google Play application is its unique string that identifies the application. It can be found in the URL of the application's Google Play website for an application.

As App Garadyi is limited to only analysing Google Play applications, users will be presented with the option to insert the Google Play handle of the application they want to analyse. Once entered, App Garadyi will conduct its analysis.

FR 1.3. Users can view the results of previously analysed Google Play applications on the App Garadyi website.

App Garadyi will contain a web-page that lists Google Play applications that have been previously analysed. Users will be able to click on a previously performed analysis and be redirected to the corresponding 'results' page.

FR 1.4. Users can download all the analysis results of an application in a JSON formatted text file.

Users will have the option of downloading the results in a JSON formatted text file. All results will be displayed in this file. This includes static analysis, dynamic analysis, meta-information, privacy policy analysis and detection of incorrect privacy policy.

FR 1.5. Each individual analysis of an APK has a primary key

The intention of this requirement is to assist in satisfying requirement FR 1.3. In order for users to be able to access a web-page of a unique analysis of an APK file, that analysis will need to be given a unique key to identify that analysis. Note that the hash of an APK can not be the primary key because there may be multiple instances of analysis testing the same APK.

4.1.2 Static Analysis

Here, the requirements that relate to the static analysis of Google Play applications are discussed.

FR 2.1. Once the Google Play handle of an application has been sent to App Garadyi, the APK file for that application will be downloaded to App Garadyi's storage device.

The intention of this requirement is to make it unnecessary for users of App Garadyi to download an APK on to their device only to re-upload it to App Garadyi.

FR 2.2. App Garadyi will decompile an APK file to human-readable files.

As APK files can not be read in their compiled form, a decompiling tool will need to be used on an APK before conducting static analysis.

FR 2.3. App Garadyi will extract the permissions requested by an application and categorise permission based on their "protection level".

Protection level refers to the level of risk that Android has assigned to a permission. The varying App Garadyi will also label these permissions as "normal", "signature" or "dangerous", based off the protection level they were assigned by the Android developers.

FR 2.4. App Garadyi will detect third-party libraries in the source code of a given Google Play application. App Garadyi will also need to determine what the purpose of the library is. Such as "tracking", "analytics", "advertisement", etc.

APK files do not explicitly state what (if any) third-party libraries it uses. App Garadyi will need to use a deductive method for detecting the use of third-party libraries.

FR 2.5. App Garadyi will verify the authenticity of the files inside the APK.

APK files contain a digital certificate and thus its public key. There is also a signature file in the APK that lists the message digest of every source code file inside the APK with respect to its public key. This public key can be used to verify the authenticity every file inside the APK by calculating the digest of every file and comparing it against the database. This is to prevent "repacking" attacks, where malicious actors will edit the source code of the application and make it available for download.

FR 2.6 App Garadyi will detect whether a given Google Play application is malicious by parsing it through VirusTotal.

VirusTotal is an online anti-virus website that will test whether a file contains malware or not.

4.1.3 Dynamic Analysis

In this subsection, we discuss how App Garadyi will be required to conduct dynamic analysis to detect network traffic going to third-parties.

FR 3.1. App Garadyi will automatically install the APK onto an Android Studio emulator.

Google Play applications can be installed and run to near-exact functionality on the Android Studio emulator. App Garadyi will install the application on to the Android Studio emulator.

> **FR 3.2.** App Garadyi will open the application on the emulator and feed 1,000 random user commands to the emulator while running the application.

The intention of this random activity is to try and trigger network activity that may occur due to interactions with the application.

> **FR 3.3.** App Garadyi will collect all network traffic from the emulator while the application is running and make this information available to the user via download.

This requirement can be met by redirecting network traffic through a proxy and using a tool to record the network traffic.

> **FR 3.4.** App Garadyi will determine if there is communication between the application and third-parties by analysing the network traffic.

App Garadyi will monitor network traffic through a proxy network and capture any traffic that is being transmitted to a known tracking domain.

4.1.4 Meta-Information Analysis

> **FR 4.1.** App Garadyi will collect the meta-information displayed on the Google Play website about a given application.

This meta-information includes the rating, number of installs, developer name, description, links to developer website, privacy policy and contact information.

FR 4.2. App Garadyi will determine the hash of a given APK file before and after running it on an emulator.

This will determine if the APK file has changed and may have had malicious code installed during the run-time of the application.

4.1.5 Privacy Policy Analysis

Here, we discuss the requirements associated with using natural language processing and machine learning to determine whether a privacy policy has claimed it shares user information with third-parties.

FR 5.1. App Garadyi will download the privacy policy HTML file to its database.

Every Google Play application has a link to its privacy policy. App Garadyi will download this privacy policy to prepare it for analysis.

FR 5.2. App Garadyi will determine whether a privacy policy acknowledges the sharing of user information to third-parties or not using machine learning algorithms and a neural network.

This requirement will be met through a supervised machine learning algorithm that has been trained on a corpus of Android privacy policies.

4.1.6 Detecting Privacy Policy Information Sharing Violation

In this subsection, the requirements for detecting whether a privacy policies stance on third-party information sharing is inconsistent with an applications actual behaviour is discussed.

FR 6.1. App Garadyi will determine if a Google Play application incorrectly claims to not share user information with third-parties.

This requirement is an amalgamation of the requirements: FR 2.4, FR 3.4 and FR 5.2. If an APK does appear to be sharing information with third-parties as detected in FR 2.4 or FR 3.4, but, it does not claim its sharing of information to third-parties in its privacy policy as detected in FR 5.2 then App Garadyi will conclude that there is an inconsistency.

4.2 Non-Functional Requirements

In this section, the non-functional requirements of App Garadyi are discussed. These requirements have been split into the following subsections:

- Speed of App Garadyi
- Security of App Garadyi
- Visualization of Results

4.2.1 Speed of App Garadyi

> **NFR 1.1.** App Garadyi will conduct all analysis of an application within 120 seconds after downloading the APK file.

This includes static analysis, dynamic analysis and privacy policy analysis. As the download rate and size of an APK is unpredictable and can not be improved upon, this requirement specifically mentions that this time frame will start once downloading the APK file has complete.

> **NFR 1.2.** Users will be able to access an APK in App Garadyi's database page instantly after analysis has been completed.

This requirement is to ensure that App Garadyi's users can visit the analysis of a requested Google Play application instantly.

4.2.2 Security of App Garadyi

NFR 2.1. Users will be required to register login credentials.

The aim of this is to detect and deter any illegitimate use of App Garadyi (such as spamming).

NFR 2.2. Users login details will be hashed and salted.

This is to prevent any brute-force attacks in the event a malicious actor gained access to the database of logins and passwords.

4.2.3 Visualisation of Results

NFR 3.1. App Garadyi will display the results (such as permissions, privacy policy results, etc) in a visually appealing way.

This requirement is necessary so that users can easily read the results of an analysis.

4.3 Site Map

This section presents the site map of App Garadyi and brief explanations of how users can navigate the site.

The user will access the "Uploads" page to request a Google Play application to be analysed. Once the analysis was successful, the user will be redirected to the newly created and unique results page for that particular analysis.

In the event a user wants to access previously analysed Google Play applications, the user will be able to do so through the "Database" page. The Database page will provide a list of previously conducted analysis and a search tool for users to able to locate previously performed analysis.

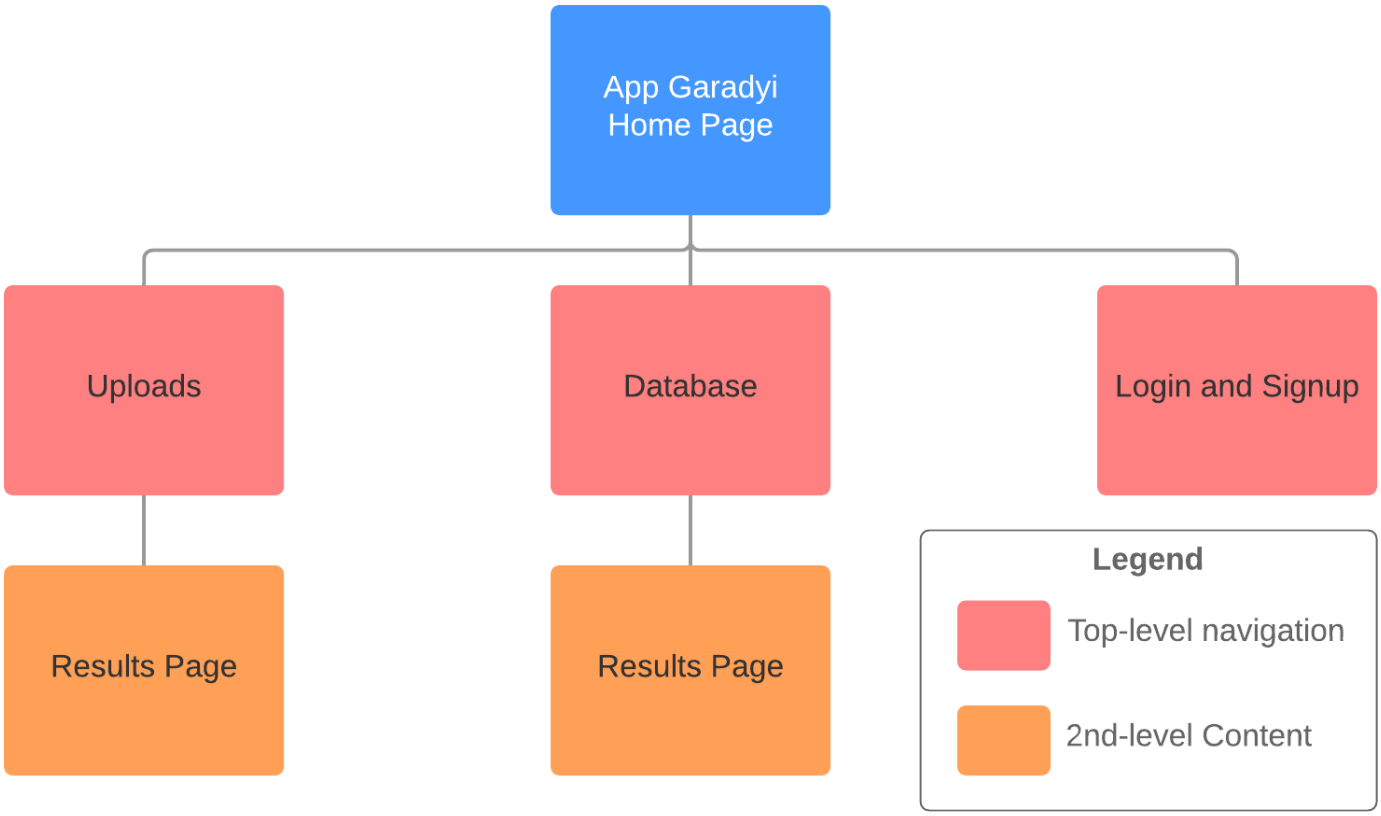

FIGURE 4.1: App Garadyi Site Map

4.4 Sequence Diagram

This sequence diagram demonstrates how App Garadyi will interact with external information sources and the order in which App Garadyi does so. In the event that a source of information is inaccessible (such as the Google Play Store website not responding), then App Garadyi will continue its data collation and analysis without that information. A results page will still be displayed, but, with an acknowledgment of what information is missing.

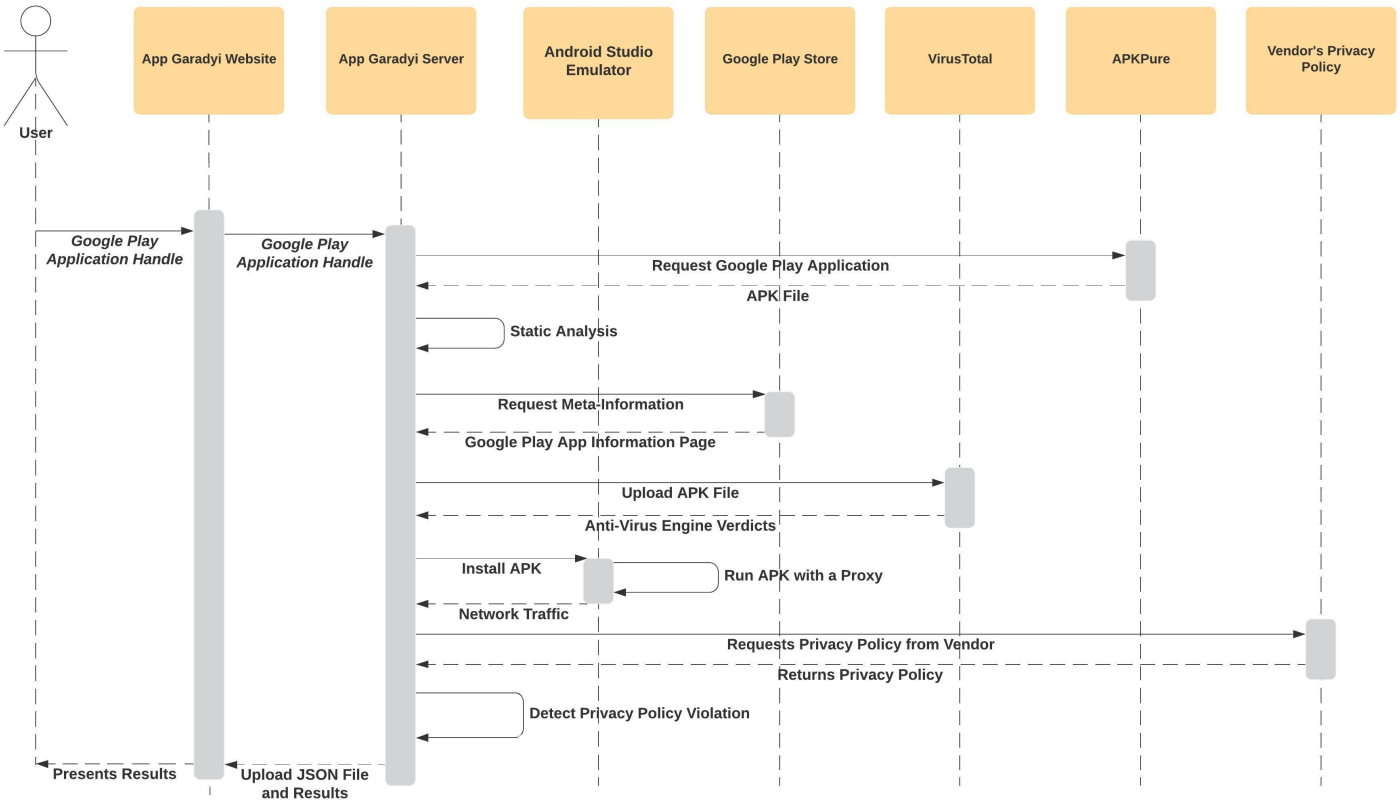

FIGURE 4.2: App Garadyi Sequence Diagram

CHAPTER 5

Methodology

In this chapter, the three components in this project's methodology are discussed. First, the tools used to build App Garadyi are mentioned; second, the ways in which App Garadyi detects security and privacy issues in an application are discussed; and finally, the methodology for the case study is explained.

5.1 Software Tools

In this section, the software that were used to build App Garadyi are discussed. In addition to this, a short description of the purpose of each software tool is given. If relevant, the version number of the tool used is given as well.

5.1.1 Python and Django

Django [45] version 3.0.1 is a web-development framework that lets developers build a full-stack website using Python. Django was used to build App Garadyi and App Garadyi's web interface. Python version 3.8.1 was used.

5.1.2 Apktool

Apktool [46] version 2.4.1 was used to decompile an Android application from an ".apk" file to a folder which contains the file's original contents in plain-text.

5.1.3 Android Studio

Android Studio [47] is an IDE for Android application developers. The importance of Android Studio is that Android Studio allows the testing of Android applications through an emulator. An emulator can be run and given commands through the command line using Android Debug Bridge and MonkeyRunner. Both Android Debug Bridge and MonkeyRunner are part of the Android SDK and come with the installation of Android Studio.

5.1.4 Mitmproxy

Mitmproxy [48] version 5.1 is a HTTPS proxy service which App Garadyi uses to capture and analyse the network traffic of an application.

5.1.5 OpenSSL

OpenSSL [21] is a toolkit that assists in analysing TLS and SSL protocols. App Garadyi applied OpenSSL to convert the digital certificates in Android Applications to readable text.

5.1.6 Jarsigner

Jarsigner [49] is a tool that enables the verification of JAR files through the validation of signatures. App Garadyi uses this tool to verify APK files.

5.1.7 APKPure

App Garadyi uses 'APKPure' [50] to download Android applications. APKPure is a website that enables its users to download exact copies of free Google Play applications.

An existing Python script [51] was used to download APK files from APKPure. This script works by searching for a Google Play application's handle on the APKPure website. The script will then download the closest match that APKPure finds. However, this script has been edited to only download an APK file if the handle the user is searching for matches the handle

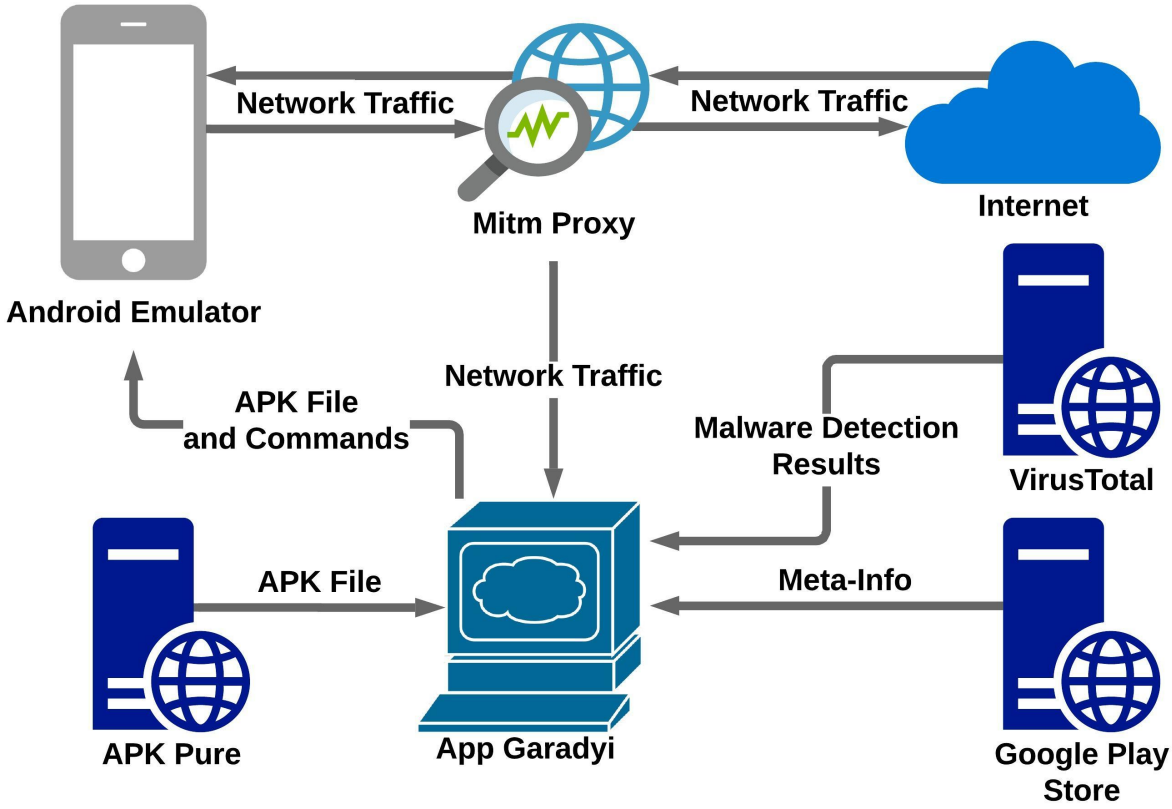

FIGURE 5.1: App Garadyi Data Collection Method

of the APK file. In the event that App Garadyi can not find the Google Play application on the APKPure website, the user of App Garadyi is redirected to an error page.

5.2 Privacy Issue Detection

App Garadyi collects information about an Android application from various sources and uses them to present privacy concerns. This section discusses how App Garadyi retrieves this information and the methodologies it uses to identify an applications security issues. First, this section presents methodologies related to the use of third-party libraries and then explains the additional security analysis that App Garadyi produces.

5.2.1 Detecting Third-Party Libraries Using Static Analysis

Third-party libraries are not explicitly identified in an APK file. However, previous research has been conducted to reveal how to identify third-party libraries that have been installed in APK files. When third-party libraries are installed into Android Applications, they create unique sub-directories inside the APK file. App Garadyi uses a database that connects the names of sub-directories in an APK to a third-party library. If the name of a sub-directory matches that of a third-party library, the application is deemed to have been using that third-party library.

This database originated from [17] and was expanded upon to analyse mobile ad-blocking applications [52] and used to analyse VPN applications [53]. This database can be found here [54].

5.2.2 Detecting Third-Party Network Traffic

Mobile applications with third-party libraries will often send user information to these third-parties. This information sharing can be detected by analysing the network traffic of the mobile device while the application is running.

App Garadyi installs a given application on to Android Studio's emulator. App Garadyi then proceeds to begin monitoring the network traffic of the mobile emulator through Mitmproxy. 1,000 random user commands are sent to the mobile device while running the application as a means of triggering network traffic. Once the 1,000 commands have been sent, the application is uninstalled from the mobile emulator and the network traffic is stored in App Garadyi's database.

App Garadyi uses Easy List's EasyPrivacy list [55] to identify any network traffic between the mobile device and third-parties. This is a list containing HTTP/S requests that are known to be information collectors. Using this list, App Garadyi compares the network traffic headers with the list to determine whether any of an applications network traffic is sending information to third-parties.

5.2.3 Analyzing a Privacy Policy Using Neural Networks

Note to marker: This section is partially incomplete. There will be more details to come when I build the neural network, such as hyper-parameters.

In this subsection, we discuss the methodology for detecting whether a Google Play application's privacy policy claims to share user information with third-parties or not. App Garadyi does this through a neural network. The methodology in how this neural network was developed, including tools that were used to build the neural network and how the neural network was trained are discussed. Finally, this subsection explains how App Garadyi can automatically retrieve a privacy policy for analysis.

5.2.3.1 Tools Used to Build the Neural Network

It is worth noting the tools used to build the neural network. App Garadyi uses the Fastai Python library [56], an extension of PyTorch [57], to assist in the development of the neural network. These libraries will essentially enable the neural network development process to be written in very few lines of code. These tools will automate pre-processing (e.g. tokenization and numericalization), as well as the training of a neural network on a corpus of privacy policies.

5.2.3.2 Training a Neural Network

Before our neural network can analyse a privacy policy, we apply the concept of transfer learning to help the neural network interpret the English language. App Garadyi's neural network builds off of Wiki 103's language model [58], which has the ability to predict the next word in a sentence, and uses this pre-built language model to create a classifier that will detect whether a privacy policy claims to share user information to third-parties.

We specifically build a recurrent neural network due to its success in being applied to classification problems and text-based problems. In this study, we use a corpus of 350 Android application privacy policies [59] to train our neural network to distinguish between a

privacy policy that shares user information to third-parties with one that does not. This corpus contains the text of privacy policies as well as annotations regarding third-party information sharing.

An 80:20 split between training and testing privacy policies was used. This equates to 280 training privacy policies and 70 testing privacy policies. During training, we specifically avoid "fine-tuning" the hidden layers within the neural network to avoid over-fitting as the privacy-policy corpus is relatively small.

5.2.3.3 Downloading Privacy Policies for Analysis

Every Google Play application that accesses user information provides a link to its privacy policy on its Google Play page. App Garadyi will download the privacy policy of a given application and extract the text using the Beautiful Soup Python Library. The Beautiful Soup Library is able to extract all the text displayed on a web-page using its "get_text()" function.

5.2.4 Privacy Policy Violation Detection Algorithm

If a third-party library was detected via network traffic or static analysis in an application, and that application's privacy policy did not claim to share user information with third-party libraries, App Garadyi identifies a privacy policy violation. This algorithm can be best visualised through Figure 5.2.

5.2.5 Permissions

Third-party libraries inherit the permissions of the application they are installed on. As a result, if an application has access to private user information through permissions, the third-party library also has this private user information. Naturally, when determining the privacy risks of an Android application, it is important to identify the permissions of the application.

Every Android application has a manifest file, which contains the total list of permissions that the application uses. App Garadyi will collect the total list of permissions used in the

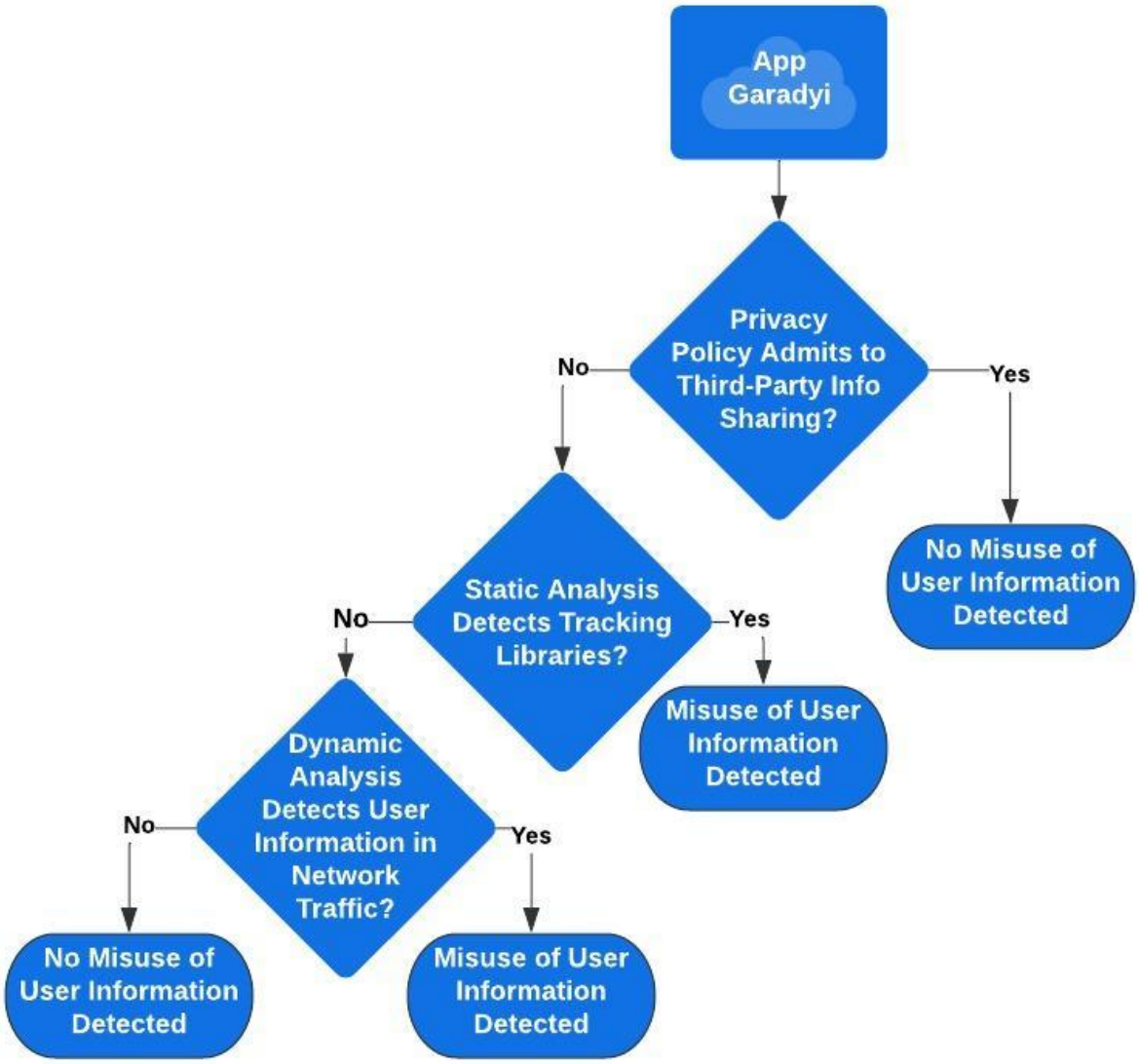

FIGURE 5.2: Privacy Policy Violation Dataflow Graph

application from the manifest file. App Garadyi will also collect the corresponding protection level of each permission. The protection level refers to the impact that granting this permission will affect user's privacy and whether the user needs to explicitly grant the permission. The protection level for an application is determined by Android.

Protection Level	Privacy Risk	Needs Granting by User	Example
Normal	Low	No	ACCESS_WIFI_STATE
Signature	Medium	No	BATTERY_STATS
Dangerous	High	Yes	READ_CONTACTS

TABLE 5.1: Permission Protection Levels

App Garadyi has a database of permissions and their corresponding protection level which it uses to determine the protection level of a permission. This was created using a custom-made Python web scraper that pulled data from the Android permissions website [60].

5.2.6 Verifying the Authenticity of an APK

The digital certificate of an Android application can be found inside the application. The data within a digital certificate file is converted to a text format using OpenSSL, so that it can be read in plain-text. Once converted into text, the file presents information about the certificate such as the public key, the issuer, the dates it is valid and more.

In addition to the digital certificate, a signature file is also contained within the APK. This signature file contains the message digests of all the source code files within the APK. App Garadyi uses the public key to hash every file listed in the APK and compare it against the message digests in the signature file. App Garadyi performs this process automatically by using Jarsigner. App Garadyi will indicate to its users if any of the source code has changed.

5.2.7 Google Play Meta-Information

Google Play will display an application's number of installs, size, date it was last updated, version number, minimum Android version, developer, developer contact details, link to developer privacy policy, description and potentially more. App Garadyi will collect all of the meta-information that is listed on the Google Play website.

App Garadyi collects this information by going to an application's Google Play URL and running a web-scraper that will collect the various meta-information. The web-scraper was built using the "Beautiful Soup" [61] Python library to retrieve the relevant meta-information.

App Garadyi will present to its users this meta-information and emphasise some potential concerns. These concerns include a very low number of installs and a very low rating.

5.2.8 VirusTotal Malware Detection

In this subsection, we discuss how VirusTotal can be used to detect malicious Android applications and malicious network traffic.

5.2.8.1 Malicious Application Code

VirusTotal [33] is a free online tool that runs up to 70 antivirus engines on a given file. Each antivirus engine will provide a verdict on whether an APK file is malicious, and if so, it provides the reason. App Garadyi uses VirusTotal's Python API [62] to retrieve the malware detection results. VirusTotal's Python API will provide App Garadyi with the results in a JSON text file, which App Garadyi will extract the results from for each antivirus engine.

VirusTotal has been used in many Android application security analysis research papers as a means of detecting malicious applications. Typically, these research papers set thresholds when conducting empirical analysis in order to arrive at a conclusion at whether an application is malicious or not. Some research has set the bar at having a threshold as high as 50% of VirusTotal engines indicating a positive result, with the intention of removing any doubt that a given application was malicious or not [30]. With this in mind, App Garadyi applies a benchmark of 25% positive detections of malware before it concludes that the Android application is malicious. If this benchmark of 25% is breached for an application being analysed by App Garadyi, the application is flagged as malware.

5.2.8.2 Malicious Network Traffic

By using Mitmproxy to capture all network traffic from the emulator while an application is running, we can collect the domain names of all communications from an application. Mitmproxy has a Python library that enables this. Similar to analysing APK files, these domain names can be analysed by VirusTotal's anti-virus engines to determine whether a

domain is malicious. Using VirusTotal's Python API, we can request the analysis and gather the results of this analysis. These results are presented to the user of App Garadyi, and if more than 25% of the anti-virus engines deem a domain to be malicious, App Garadyi will flag the Google Play application being analysed as malicious.

5.3 Case Study

In this section, the methodology for conducting the case study are discussed. Specifically, this section discusses the criteria used to determine whether an application was suitable for this case study and the algorithm used to find them

5.3.1 Application Criteria

Table 5.2 represents the criteria for an application to be part of the 5,000 health-applications case study.

Criteria ID	Criteria Description
Criteria 1	The application must be in the "Health and Fitness" category in the Google Play store
Criteria 2	The application's privacy policy has to be in English
Criteria 3	The application must have at least 10,000 installs
Criteria 4	The application must be free

TABLE 5.2: Case Study Criteria

Criteria 1 is necessary to ensure that the application is indeed a health application. Criteria 2 is essential as the neural network only works on English privacy policies. Criteria 3 aims to make the results as relevant as possible - analysing applications with a higher number of installs will be more effective reflection of the mobile health industry. Criteria 4 exists because it would be very expensive to purchase thousands of mobile applications.

5.3.2 Application List Compilation

A Python script was written to compile a list of applications that meet the criteria. This Python script will begin at one application and collect 5 applications that are listed as "similar" to that

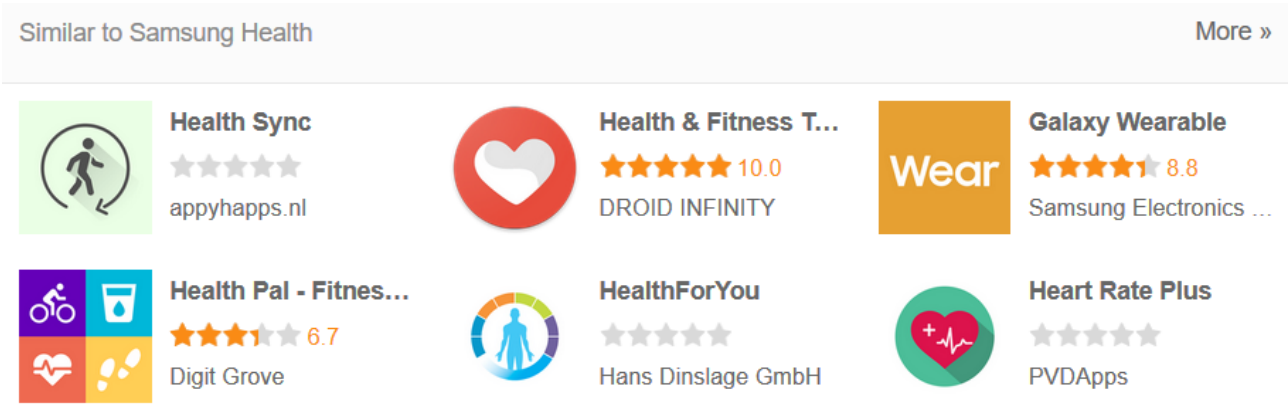

FIGURE 5.3: Applications Listed as Similar to "Samsung Health"

application on the APKPure web-site. The algorithm will dismiss any applications that are already within the list of collected applications or do not meet the criteria in Table 5.2. Then, the algorithm will recursively perform the same behaviour until 5,000 applications have been collected.

CHAPTER 6

Results

Note to marker: This chapter is just a template for me to work from in Thesis B. No results have been collected yet.

This chapter discusses the results achieved by App Garadyi. This includes the accuracy of App Garadyi at detecting privacy policy violations regarding the sharing of user information to third-parties and also the results of the case study of health applications in the Google Play Store.

6.1 Empirical Analysis of Health Applications

In this section, we state the salient statistical features regarding the case study. This includes what portion of health applications violate their own privacy policy by sharing information with third-party libraries, determining the frequency of the different types of third-party libraries and determining the frequency of permissions used in health applications.

6.1.1 Privacy Policy Violations Results

In this subsection, we breakdown the statistics surrounding privacy policy violation due to the sharing of user information to third-party libraries.

6.1.2 Frequency of Third-party Libraries Results

Here, we present the data about the frequency of the various categories of third-party libraries within health applications. These categories include analytics, advertising, utility, development aid and UI components.

6.1.3 Permissions Results

As third-party libraries automatically inherit access to permissions requested by the application they are installed on, the permissions of the applications in the case study are worth investigation. In this subsection, we present the most common permissions and how they relate to the various third-party libraries and privacy policy violations in health applications.

6.2 App Garadyi Privacy Policy Detection Results

In this project, a random set of 5,000 Google Play applications in the Health category were collected. This set of applications were analyzed by App Garadyi to give a verdict on whether there is a contradiction in the privacy policy of the application regarding user data. After manual inspection of 500 applications, it was found that App Garadyi had a precision of [x%] and recall of [y%] when detecting if an application was sending users private information to a third-party and not acknowledging it in its privacy policy.

CHAPTER 7

Discussion

Note to marker: This chapter is just a placeholder

CHAPTER 8

Conclusion

Note to marker: This chapter is just a placeholder

8.1 Future Work

Bibliography

[1] S. O'Dea, *Number of mobile devices worldwide 2019-2023*, Feb. 2020. [Online]. Available: https://www.statista.com/statistics/245501/multiple-mobile-device-ownership-worldwide/.

[2] P. Krebs and D. T. Duncan, 'Health app use among us mobile phone owners: A national survey', *JMIR mHealth and uHealth*, vol. 3, no. 4, Apr. 2015. DOI: 10.2196/mhealth.4924.

[3] R. Slavin, X. Wang, M. B. Hosseini, J. Hester, R. Krishnan, J. Bhatia, T. D. Breaux and J. Niu, 'Toward a framework for detecting privacy policy violations in android application code', *Proceedings of the 38th International Conference on Software Engineering - ICSE 16*, 2016. DOI: 10.1145/2884781.2884855.

[4] J. Clement. (2020). App stores: Number of apps in leading app stores 2019, [Online]. Available: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.

[5] A. Mense, S. Steger, M. Sulek, D. Jukic-Sunaric and A. Mészáros, 'Analyzing privacy risks of mhealth applications', *Stud Health Technol Inform*, vol. 221, pp. 41–45, 2016.

[6] A. Papageorgiou, M. Strigkos, E. Politou, E. Alepis, A. Solanas and C. Patsakis, 'Security and privacy analysis of mobile health applications: The alarming state of practice', *IEEE Access*, vol. 6, pp. 9390–9403, 2018. DOI: 10.1109/access.2018.2799522.

[7] D. He, M. Naveed, C. A. Gunter and K. Nahrstedt, 'Security concerns in android mhealth apps', eng, *AMIA ... Annual Symposium proceedings. AMIA Symposium*, vol. 2014, p. 645, 2014, ISSN: 1942-597X.

[8] T. Brüggemann, J. Hansen, T. Dehling and A. Sunyaev, 'An information privacy risk index for mhealth apps', *Privacy Technologies and Policy Lecture Notes in Computer Science*, pp. 190–201, 2016. DOI: 10.1007/978-3-319-44760-5_12.

[9] Q. Grundy, K. Chiu, F. Held, A. Continella, L. Bero and R. Holz, 'Data sharing practices of medicines related apps and the mobile ecosystem: Traffic, content, and network analysis', *bmj*, vol. 364, p. l920, 2019.

[10] *User data | privacy, security, and deception - developer policy center*. [Online]. Available: https://play.google.com/about/privacy-security-deception/user-data/.

[11] *Jumbo privacy review: North dakota's contact tracing app*. [Online]. Available: https://blog.jumboprivacy.com/jumbo-privacy-review-north-dakota-s-contact-tracing-app.html.

[12] *Path social networking app settles ftc charges it deceived consumers and improperly collected personal information from users' mobile address books*, Feb. 2013. [Online]. Available: https://www.ftc.gov/news-events/press-releases/2013/02/path-social-networking-app-settles-ftc-charges-it-deceived.

[13] K. Huckvale, J. T. Prieto, M. Tilney, P.-J. Benghozi and J. Car, 'Unaddressed privacy risks in accredited health and wellness apps: A cross-sectional systematic assessment', *BMC Medicine*, vol. 13, no. 1, 2015. DOI: 10.1186/s12916-015-0444-y.

[14] L. Parker, V. Halter, T. Karliychuk and Q. Grundy, 'How private is your mental health app data? an empirical study of mental health app privacy policies and practices', *International journal of law and psychiatry*, vol. 64, pp. 198–204, 2019.

[15] L. Li, T. F. Bissyandé, M. Papadakis, S. Rasthofer, A. Bartel, D. Octeau, J. Klein and L. Traon, 'Static analysis of android apps: A systematic literature review', *Information and Software Technology*, vol. 88, pp. 67–95, 2017. DOI: 10.1016/j.infsof.2017.04.001.

[16] C. Gibler, J. Crussell, J. Erickson and H. Chen, 'Androidleaks: Automatically detecting potential privacy leaks in android applications on a large scale', *Trust and Trustworthy*

Computing Lecture Notes in Computer Science, pp. 291–307, 2012. DOI: 10.1007/978-3-642-30921-2_17.

[17] S. Seneviratne, H. Kolamunna and A. Seneviratne, 'A measurement study of tracking in paid mobile applications', in *Proceedings of the 8th ACM Conference on Security Privacy in Wireless and Mobile Networks*, ser. WiSec '15, New York, New York: Association for Computing Machinery, 2015, ISBN: 9781450336239. DOI: 10.1145/2766498.2766523. [Online]. Available: https://doi.org/10.1145/2766498.2766523.

[18] K. A. Talha, D. I. Alper and C. Aydin, 'Apk auditor: Permission-based android malware detection system', *Digital Investigation*, vol. 13, pp. 1–14, 2015. DOI: 10.1016/j.diin.2015.01.001.

[19] A. Kumar, K. Kuppusamy and G. Aghila, 'Famous: Forensic analysis of mobile devices using scoring of application permissions', *Future Generation Computer Systems*, vol. 83, pp. 158–172, 2018. DOI: 10.1016/j.future.2018.02.001.

[20] S. Bojjagani and V. Sastry, 'Stamba: Security testing for android mobile banking apps', *Advances in Signal Processing and Intelligent Recognition Systems. Advances in Intelligent Systems and Computing, vol 425. Springer, Cham*, 2016. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-319-28658-7_57.

[21] I. OpenSSL Foundation, *Openssl*. [Online]. Available: https://www.openssl.org/.

[22] M. Spreitzenbarth, T. Schreck, F. Echtler, D. Arp and J. Hoffmann, 'Mobile-sandbox: Combining static and dynamic analysis with machine-learning techniques', *International Journal of Information Security*, vol. 14, no. 2, pp. 141–153, 2015.

[23] Z. Li, J. Sun, Q. Yan, W. Srisa-an and S. Bachala, 'Grandroid: Graph-based detection of malicious network behaviors in android applications', in *International Conference on Security and Privacy in Communication Systems*, Springer, 2018, pp. 264–280.

[24] H. Zhang, D. Yao and N. Ramakrishnan, 'Causality-based sensemaking of network traffic for android application security', in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, 2016, pp. 47–58.

[25] S. Dong, M. Li, W. Diao, X. Liu, J. Liu, Z. Li, F. Xu, K. Chen, X. Wang, K. Zhang and et al., 'Understanding android obfuscation techniques: A large-scale investigation in the wild', *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Security and Privacy in Communication Networks*, pp. 172–192, 2018. DOI: 10.1007/978-3-030-01701-9_10.

[26] A. Moser, C. Kruegel and E. Kirda, 'Limits of static analysis for malware detection', *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, 2007. DOI: 10.1109/acsac.2007.21.

[27] M. Y. Wong and D. Lie, 'Intellidroid: A targeted input generator for the dynamic analysis of android malware', *Proceedings 2016 Network and Distributed System Security Symposium*, 2016. DOI: 10.14722/ndss.2016.23118.

[28] T. Petsas, G. Voyatzis, E. Athanasopoulos, M. Polychronakis and S. Ioannidis, 'Rage against the virtual machine: Hindering dynamic analysis of android malware', in *Proceedings of the Seventh European Workshop on System Security*, ser. EuroSec '14, Amsterdam, The Netherlands: Association for Computing Machinery, 2014, ISBN: 9781450327152. DOI: 10.1145/2592791.2592796. [Online]. Available: https://doi.org/10.1145/2592791.2592796.

[29] X. Wang, S. Zhu, D. Zhou and Y. Yang, 'Droid-antirm: Taming control flow anti-analysis to support automated dynamic analysis of android malware', in *Proceedings of the 33rd Annual Computer Security Applications Conference*, ser. ACSAC 2017, Orlando, FL, USA: Association for Computing Machinery, 2017, pp. 350–361, ISBN: 9781450353458. DOI: 10.1145/3134600.3134601. [Online]. Available: https://doi.org/10.1145/3134600.3134601.

[30] J. Zhang, X. Zhuang and Y. Chen, 'Android malware detection combined with static and dynamic analysis', *Proceedings of the 2019 the 9th International Conference on Communication and Network Security*, 2019. DOI: 10.1145/3371676.3371685.

[31] A. Martín, A. Calleja, H. D. Menéndez, J. Tapiador and D. Camacho, 'Adroit: Android malware detection using meta-information', in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, Dec. 2016, pp. 1–8. DOI: 10.1109/SSCI.2016.7849904.

[32] A. Munoz, I. Martin, A. Guzman and J. A. Hernandez, 'Android malware detection from google play meta-data: Selection of important features', in *2015 IEEE Conference on Communications and Network Security (CNS)*, IEEE, 2015, pp. 701–702.

[33] *Virustotal*. [Online]. Available: https://www.virustotal.com/.

[34] *Sixo online apk analyzer*. [Online]. Available: https://www.sisik.eu/apk-tool.

[35] Hu.Wenjun, *Sanddroid*. [Online]. Available: http://sanddroid.xjtu.edu.cn/.

[36] [Online]. Available: https://reports.exodus-privacy.eu.org/en/.

[37] R. Slavin, X. Wang, M. B. Hosseini, J. Hester, R. Krishnan, J. Bhatia, T. D. Breaux and J. Niu, 'Toward a framework for detecting privacy policy violations in android application code', in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 25–36.

[38] L. Yu, X. Luo, J. Chen, H. Zhou, T. Zhang, H. Chang and H. K. N. Leung, 'Ppchecker: Towards accessing the trustworthiness of android apps' privacy policies', *IEEE Transactions on Software Engineering*, pp. 1–1, 2018.

[39] X. Wang, X. Qin, M. B. Hosseini, R. Slavin, T. D. Breaux and J. Niu, 'Guileak', *Proceedings of the 40th International Conference on Software Engineering*, 2018. DOI: 10.1145/3180155.3180196.

[40] S. Zimmeck, Z. Wang, L. Zou, R. Iyengar, B. Liu, F. Schaub, S. Wilson, N. Sadeh, S. M. Bellovin, J. Reidenberg and et al., 'Automated analysis of privacy requirements for mobile apps', *Proceedings 2017 Network and Distributed System Security Symposium*, 2017. DOI: 10.14722/ndss.2017.23034.

[41] L. Yu, T. Zhang, X. Luo and L. Xue, 'Autoppg: Towards automatic generation of privacy policy for android applications', in *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*, ser. SPSM '15, Denver, Colorado, USA: Association for Computing Machinery, 2015, pp. 39–50, ISBN: 9781450338196. DOI: 10.1145/2808117.2808125. [Online]. Available: https://doi.org/10.1145/2808117.2808125.

[42] S. Nishant, *App privacy policy generator*. [Online]. Available: https://app-privacy-policy-generator.firebaseapp.com/.

[43] *Privacy policy for android apps*. [Online]. Available: https://www.iubenda.com/en/help/11552-privacy-policy-for-android-apps.

[44] L. Yu, X. Luo, C. Qian, S. Wang and H. K. N. Leung, 'Enhancing the description-to-behavior fidelity in android apps with privacy policy', *IEEE Transactions on Software Engineering*, vol. 44, no. 9, pp. 834–854, Jan. 2018. DOI: 10.1109/tse.2017.2730198.

[45] *Django*. [Online]. Available: https://www.djangoproject.com/.

[46] *Apktool*. [Online]. Available: https://ibotpeaches.github.io/Apktool/.

[47] *Android studio*. [Online]. Available: https://developer.android.com/studio/.

[48] *Mitmproxy - https proxy*. [Online]. Available: https://mitmproxy.org/.

[49] Jan. 2016. [Online]. Available: https://docs.oracle.com/javase/8/docs/technotes/tools/windows/jarsigner.html.

[50] *Apk pure*. [Online]. Available: https://apkpure.com/.

[51] D. Sulaiman, *Download apk*, 2019. [Online]. Available: https://gist.github.com/dawand/7b4308d568c6b955b645dd7e707e5cf1.

[52] M. Ikram and M. A. Kaafar, 'A first look at mobile ad-blocking apps', in *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*, IEEE, 2017, pp. 1–8.

[53] M. Ikram, N. Vallina-Rodriguez, S. Seneviratne, M. A. Kaafar and V. Paxson, 'An analysis of the privacy and security risks of android vpn permission-enabled apps', *Proceedings of the 2016 ACM on Internet Measurement Conference - IMC 16*, 2016. DOI: 10.1145/2987443.2987471.

[54] *Tpl database*. [Online]. Available: https://github.com/JakeGarth/django/blob/master/Smali%20List/apps_libraries.txt.

[55] *Easylist*. [Online]. Available: https://easylist.to/easylist/easyprivacy.txt.

[56] J. Howard *et al.*, *Fastai*, https://github.com/fastai/fastai, 2018.

[57] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, 'Automatic differentiation in pytorch', 2017.

[58] S. Merity, C. Xiong, J. Bradbury and R. Socher, 'Pointer sentinel mixture models', *arXiv preprint arXiv:1609.07843*, 2016.

[59] S. Zimmeck, P. Story, D. Smullen, A. Ravichander, Z. Wang, J. Reidenberg, N. C. Russell and N. Sadeh, 'Maps: Scaling privacy compliance analysis to a million apps', *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, pp. 66–86, Jan. 2019. DOI: 10.2478/popets-2019-0037.

[60] D. Sulaiman, *Download apk*, 2019. [Online]. Available: https://developer.android.com/reference/android/Manifest.permission.

[61] *Beautiful soup documentation*. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/.

[62] *Virustotal python api*. [Online]. Available: https://developers.virustotal.com/reference.