Thomas Brüggemann

Master Thesis
im Fach Information Systems

Automated Information Privacy Risk Assessment of Android Health Applications

Themensteller: Prof. Dr. Ali Sunyaev

Vorgelegt in der Masterprüfung
im Studiengang Information Systems
der Wirtschafts- und Sozialwissenschaftlichen Fakultät
der Universität zu Köln

Köln, September 2016

Contents

Index of Abbreviations

API	Application Programming Interface
APK	Android Application Package
DEX	Dalvik Executable
DRM	Digital Rights Management
JAR	Java Archive
JVM	Java Virtual Machine
mHealth	Mobile Health

1. Problem Statement

The market for mobile phone and tablet applications (apps) has grown extensively since recent years.[1] It is increasingly easier for companies or even single developers to create unique apps that reach millions of users around the planet via digital app stores. This market growth affected mobile health (mHealth) apps as well. More and more mHealth apps are available that support the users in resolving their health-related issues and that try to remedy health-related information deficiencies.

But receiving personal health-related information yields information privacy risks to users. Users are asked to expose personal health-related information, e.g. information on disease symptoms or medical appointments in order to receive a tailored app that fits their needs.[2] It remains however unclear how and where the vulnerable user information is sent, processed and stored.[3]

The information about these privacy related practices of app providers and their offered apps should be stated in the privacy policy document provided by the app provider.[4] Processing these privacy policies requires a higher level of education and time to read through large bodies of text, in order to find the relevant information. Additionally, the important information is hidden in legal language or is insufficiently addressed, if at all.[5] Aside from data usage beyond the control of the users, it is also challenging to assess what kind of private information an app asks for, prior to the app usage. Users have to download the apps of interest and try them out, before it becomes clear what health-related information is processed by the app and in which way. This leads to low comparability between apps. When users are looking for specific functionality in an mHealth app, it is challenging to find the app that offers the desired functionality at an acceptable information privacy risk. Even if users would pursue the task of finding and comparing mHealth apps of similar functionality, the high volume of apps available in the app stores[6] makes

[1] See for this and the following sentence Enck et al. (2011), p. 1.

[2] See Chen et al. (2012), p. 2.

[3] See He et al. (2014), p. 652.

[4] This paragraph follows Dehling, Gao, Sunyaev (2014), p. 11.

[5] See **Pollach2007** p. 104.

[6] See Enck et al. (2011), p. 1.

it laborious to review all of them by hand. One way to assess information privacy risks of the large amount of mHealth apps is to automate the review process of each individual app. The assessment automation can be done by downloading and analyzing the source code of each app and by tracing data leaks. Static code analysis is used in the field of informatics to analyze application source code and detect faults or vulnerabilities.[7] It is yet unclear how and to what degree the concepts of static code analysis and information privacy risk assessment can be combined in order to automate app assessment. A static code analysis could, in theory, be used to automatically assess some of the information privacy risks that mHealth apps pose. Previous research has not shown how and to what degree the combination of static code analysis and information privacy risks assessment is feasible in the field of mHealth app information privacy risk assessment and therefore the aim of this study is to explore the possibilities of static code analysis for information privacy risk assessment. This leads to the research question: How and to what degree can the information privacy risks of mHealth apps be automatically assessed? The ’degree’ refers to the amount and the level of detail that information privacy risk factors can be automatically assessed.

The automated process furthermore can help to drastically reduce the effort of reviewing each individual app and can enhance the information experience users receive while looking for mHealth apps. Additionally, it exposes new possibilities for research in the information privacy risks area. The research could be conducted on providing solutions and best practices for further enhancing the information privacy risks communication of apps.

2. Related Work

mHealth apps have been examined in various research studies that aim at providing insights for developers as well as users into how private information is processed. Privacy issues are the most impactful user complaint while using mobile apps.[8] This encourages research to address information privacy risks.

Research focus has been put on the technical side of information privacy breach. It

[7] See Baca, Carlsson, Lundberg (2008), p. 79.

[8] See Khalid et al. (2015), p. 5.

has been analyzed, to what degree the data storage in internal Android log files or on the memory card within a phone or tablet poses a threat to users information privacy.[9] Technical evaluation of mobile apps even goes further into the topics of decompilation to analyze device identification or geolocation data leaks.[10] Decompilation reveals to be a feasible assessment technique for information privacy risks and data leaks.

In informatics and software development contexts, static code analysis has been used to analyze source code and provide feedback on coding styles to the users while programming or ”to find defects in programs”[11]. Static code analysis provides a fast way to analyze source code[12], which makes it suitable to automate the assessment of large datasets. A further benefit of using static code analysis to retrieve information from software is that the software does not need to be executed during the analyzation process. This additionally supports the development of fast performing assessment tools that are suitable for application on large datasets of source code since there is no need to wait for the application runtime to execute the software.

Our study will use the benefits of static code analysis and apply them to the assessment of mHealth information privacy risks. It is unclear if static code analysis is a viable tool to analyze and identify information privacy risk factors. We will use the comprehensive privacy-risk-relevant information privacy practices that **Dehling2016** identified[13] and try to implement static code analysis strategies to identify those risks automatically. This will be a vital addition to current research, since there is yet no holistic approach to apply static code analysis to information privacy risks detection that takes an ample amount of information privacy risk factors into account.

3. Objectives

The main objective of this study is to ascertain how and to what degree the assessment of information privacy risk factors for mHealth apps can be automated. In order to reach

---

9 For the previous two sentences, see He et al. (2014), p. 645-646.

10 See Mcclurg (2012), p. 1, 5., Enck et al. (2011), p. 1. and Mitchell et al. (2013), p.6-7.

11 Bardas et al. (2010), p. 1.

12 See Bardas et al. (2010), p. 5.

13 See **Dehling2016** p. 8-17.

this objective, the following sub-objectives have to be met.

The first sub-objective is to extract information privacy risk factors from the information privacy practices that **Dehling2016** identified and that are relevant for automated information privacy risk assessment. As a second sub-objective we will develop strategies to identify the information privacy risk factors within the source code of mHealth apps via static code analysis. This is necessary since it is yet unclear how and to what degree the static code analysis can help to identify information privacy risk factors of mHealth apps. Finally we will evaluate how well the automated information privacy risk assessment tool can identify information privacy risk factors in comparison to two human reviewers. In order to fully ascertain the degree static code analysis can identify information privacy risk factors, a manual review of the results of the static code analysis is necessary.

4. Definitions

Certain terms are used in the remainder of this thesis that have to be defined:

4.1 Decompilation

Compilers transfer human readable programming code into machine code and therefore help humans write software applications in understandable text form.[14] Decompilers retrieve the human readable programming code back from the compiled machine code. The compilation process of an application is non-reversible. Therefore, decompiling is a reverse engineering technique that outputs source code, similar to the original source code of the application, but with the same functionality.

4.2 Mobile Health Apps

Mobile health (mHealth) apps are smartphone or tablet applications that support users by enabling them to gather health-related information and support the consumer in dealing with medical or health-related issues.[15]

[14] This section follows Nolan (2012), p. 1-2.

[15] See Dehling, Gao, Schneider, et al. (2015), p. 1.

4.3 Information Privacy Risk Factor

An information privacy risk factor is an indicator of a potential threat or a circumstance that increases the risk of an information privacy breach. Information privacy is defined as the fact that people are able to control if, how and where information and knowledge about themselves is gathered, stored and processed.[16] Additionally society and social structure must establish an information regulation architecture that allows people to enforce their information privacy rights.[17] Threats to information privacy in digital services can happen at application level (e.g. by sharing users' personal information with third-parties), as well as at communication level (e.g. by using an unencrypted data connection).[18]

4.4 Static Code Analysis

Static code analysis refers to the analysis of application source code without actually executing the application. This technique is widely used to detect vulnerabilities or to validate the source code during development in the source code editor software.[19]

5. Methods

5.1 Automating the Information Privacy Risk Assessment

5.1.1 Android Decompilation

In order to automate the information privacy risk assessment of mHealth apps via static code analysis, it is necessary to gain access to the source code of the apps. For this study we focus on Android apps, since Apple's digital rights management (DRM) system encrypts the binary app file in a way that makes recovering the source code difficult. There are existing approaches to decompiling the Apple app binary back into its source code. These approaches involve unlocking and jailbreaking[20] an Apple iPhone or iPad, which

---

16 See **Fischer1998** p. 421-422.

17 **Solove2002** p. 1115.

18 See **Fischer1998** p. 423-427.

19 See Bardas et al. (2010), p. 2-3.

20 Jailbreaking refers to the action of unlocking the iOS device firmware to gain unrestricted access to the bootloader. See Kweller (2010), p. 1.

is a violation of the Apple terms of service and therefore forbidden.[21]

The Google PlayStore on the other hand hosts Android applications in APK containers that are non-encrypted and allow for decompilation back into the original source files. In order to automate the download process of APK files, we make use of an undocumented Google API that reveals access to the APK files from the Google PlayStore. The API can be queried by sending an Android device id, pretending to be a requesting Android device. It is used by the Google PlayStore internally and the result of the query is the binary APK file.

To get an overview of the APK files available on the Google PlayStore, we use the repository of app listings by W. Xu, Liu (2015) that was extracted from the Google Play-Store from the categories 'Health & Fitness' and 'Medical'. Due to the obstacles of gaining access to the Apple iOS binary files, we restrict the dataset to the available Android apps and conduct our automated information privacy risk assessment on these apps. The W. Xu, Liu (2015) dataset contains 5,379 Android apps. We exclude paid apps from our study. Downloading paid apps would charge the credit card of the authors since we use the same API that the Google PlayStore itself uses to purchase apps on Android devices. Downloading all 2,199 paid apps would result in a purchase value of 19,904.24 US-dollars. Therefore, we reduce the dataset to the 3,180 free apps, which is still 59.1% of the apps from the initial dataset. We will download as many of these 3,180 apps as possible. Attempting to download 3,180 apps in a short period of time triggers Googles security mechanisms, since a normal user could not download that many apps in the same timeframe from the PlayStore. We will apply a careful downloading technique and allow for pauses in between app downloads to not over-use the Google API.

As soon as the APK files are available offline, the decompiling phase of the study will begin. Our decompiling process consists of four steps.

The first step is to recover a java archive (JAR) file from the APK file. JAR files contain a collection of .class files. These .class files hold Java bytecode that can be interpreted by the Java virtual machine (JVM) at application runtime.[22] In order to extract the Java

[21] See Kweller (2010), p. 1.

[22] The previous two sentences follow Enck et al. (2011), p. 2.

bytecode .class files, we use the command line tool *dex2jar*[23]. The abbreviation DEX stands for Dalvik Executable and refers to the binary collection of compiled Java classes within the APK package file.[24].

Step two includes the actual decompiling phase of the Java bytecode back into .java files. A .java file contains exactly one Java *class* in human-readable Java code. Enck et al. (2011) developed their own Java decompiling toolchain in order to assess security issues of mHealth apps, since *dex2jar* was not functioning at the time they conducted their study.[25] Within their toolchain, they used a tool called *Soot*. *Soot* optimizes the decompiled code to improve the readability by humans. We will use *FernFlower* as the decompiling tool for our study, since it evidentially outperformed *Soot* in a further evaluation by Enck et al. (2011).[26] The result of this second step will be a directory full of .java files that represents the source code of the APK apps.

The decompiling process delivers source code files that lack any formatting. For the case of manually validating the output of the automated information privacy risk assessment, we will have to take a look into the source code files. To make manual code inspections easier to read, a tool called *astyle*[27] will be used. *astyle* sets appropriate levels on indenting to the source code, to improve the reading experience.

In the last step of our decompilation process, a tool called *apktool*[28] is used to extract all resource files from the APK file. Resource files could be images or XML files that are not compiled into application code.[29] The XML files are of interest for the automated information privacy risk assessment, because Android text input controls can be defined in an external XML file, rather than in the source code itself.

[23] Pan (2010).

[24] See L. Xu (2013), p. 6.

[25] See Enck et al. (2011), p. 16.

[26] See Enck et al. (2011), p. 6.

[27] Davidson, Pattee (2006).

[28] Tumbleson, Wiśniewski (2010).

[29] See L. Xu (2013), p. 5.

5.1.2 Static Code Analysis

In order to analyze potential information privacy risks, we are going to use a static code analysis tool. This tool will scan and parse the Java source code files and make them processable for further analysis. The static code analysis tool will not be able to identify new information privacy risk factors by itself, but rather has to be individually programmed to scan the source code for privacy risk factors.

Dehling2016 compiled a list of information privacy practices that current research literature proposes to be contained in privacy policies and implemented by app providers.[30] The information privacy practices are listed hierarchically in the categories ’data handling’, ’information collection’, ’meta information’, ’privacy controls’ and ’rationale’ of information privacy practices. We argue that, if literature proposes to include certain information privacy practices in privacy policies, these information privacy practices might indicate an information privacy risk. While not all of the risk-bearing **Dehling2016** information privacy practices will be eligible for automated static code analysis, we will classify the relevant practices and include them into our static code analysis assessment.

To name one example strategy of identifying a information privacy practices that bears an information privacy risk, we will take a look at ’SharingWithAnalystContent’. ’SharingWithAnalystContent’ is located in the information privacy practices hierarchy under ’Content’, ’InformationSharingContent’ and expresses the information privacy risk of personal user data being shared with analytic software services, e.g. click tracking services or app usage services. To identify such an information privacy risk within the source code of an app, we will compile a list of common Android analytics libraries and create a static code analysis strategy to scan our app dataset source codes for these libraries. For each individual risk-assessment strategy we will seek for patterns that can identify even more features than than we originally searched for. In the example of ’SharingWithAnalystContent’ this means seeking for a pattern in the source code that can identify Android analytics libraries beyond the predefined list of common libraries. A pattern for analytics libraries could be based on the naming of libraries, e.g. libraries that contain the phrase ”analytics”. Another pattern could be based on common method names other analytics libraries used, e.g. ’.track()’ or ’.trackView()’.

[30] This section follows **Dehling2016** p. 8-17.

5.2 Evaluation

First, we want to evaluate the degree of information privacy risk assessment automation that has been accomplished. The evaluation will be based on the feasibility and obstacles detected in implementing the automated information privacy risk assessment and a performance comparison against human assessment. We will outline which information privacy risk factors can and can not be detected automatically, as well as the reasons for this. Furthermore, we will evaluate for which information privacy risk factors the automated static code analysis is superior to using manual assessment techniques.

For evaluation of the accuracy of automated static code analysis of the apps, we will look at a selection of 15 apps in greater detail and manually review the results of the static code analysis. The manual assessment of the test app sample will be conducted by two researchers individually on the same set of 15 test apps. We will assess if the static code analysis managed to identify all of the prevailing information privacy risks that a human can identify within the app and if not, what the reasons for failure are. This will give a sense of how well the static code analysis performs in identifying the information privacy risks in comparison to a human reviewer.

6. Structure

In this section we would like to introduce the structure of the final thesis chapters.

1 Introduction	The introduction provides an overview of the problem statement, the objectives, the methods used and the structure of the thesis.
1.1 Problem Statement	Description of the research cycle of the master thesis to further stress the practice problem and its relevance, as well as the research problem and its relevance.
1.2 Objectives	This section describes the primary objective of the thesis and the subsequent secondary objectives, that we hope to reach.
1.3 Structure	The formal structure of the thesis, used to provide the reader with an overview of the thesis.

2 Combining Source Code Analysis with Information Privacy Risk Assessment	Overview of previous research and related work that has covered the topic of analyzing mHealth information privacy risks or using automated techniques on gathering privacy risk information as well as the usage of static code analysis.
2.1 Information Privacy Risk Assessment	Overview of literature that already covered the topics of information privacy risks assessent.
2.2 Static Code Analysis	Overview of literature concerning static code analysis as an analysis tool in the field of informatics.
2.3 Relevant Information Privacy Risk Factors	Introducing the **Dehling2016** information privacy practices and filtering out the ones that can pose an information privacy risk.
3 Implementation and Evaluation of an Automated Information Privacy Risk Assessment Tool	This chapter describes the main implementation phase of the automated information privacy risk assessment tool.
3.1 Implementation of an Automated Information Privacy Risk Assessment Tool	Details on the implementation of the automated mHealth information privacy risk assessment tool itself.
3.1.1 Download Phase	Description of the APK file discovery and download phase of the tool. Free Android apps are available for everybody with an Android smartphone. We want to feed mHealth apps into our automated assessment system and therefore need to download the APK files automatically.
3.1.2 Decompilation Phase	This section provides information on the decompilation of the APK files and the methods used to regain the source code of Android apps. This is a crucial step, since the source code is our main resource to perform the static code analysis on.

3.1.3 Static Code Analysis Phase	Information on the different approaches to gather information on individual information privacy risk factors, explained in chapter 3.1. We will try to identify via the static code analysis, if the mHealth app requires a login to tailor the user experience or not. Furthermore, we want to find out if personal health data is collected and if so, what kind of data. Via source code analysis we aim at identifying the targets, where data is sent and if advertisement or click-analytics services are used.
3.2 Evaluation of an Automated Information Privacy Risk Assessment Tool	Information on how we manually review the outcome of the static code analysis by checking a random set of apps from the initial dataset.
4 Feasibility of Automated Information Privacy Risk Assessment	How many apps could be automatically assessed? How many factors could be automatically identified? Was it possible to compute information on all of the information privacy risk factors from chapter 2.3?
4.1 The Automated Information Privacy Risk Assessment of Free Android mHealth Apps	Explaining the outcome of the implementation step.
4.1.1 Download Phase	Results on how many mHealth apps could be downloaded and how the download phase went.
4.1.2 Decompilation Phase	Results on how many mHealth could be decompiled and at what degree of quality.
4.1.3 Static Code Analysis Phase	Results on the outcome of the static code analysis

4.2 Evaluation of the Automated Information Privacy Risk Assessment Tool	Results of how well the automated static code analysis performed in identifying information privacy risk factors by manually surveying a set of apps and comparing the information privacy risk factors to the static code analysis results.
5 Discussion	
5.1 Principle Findings	Information on the feasibility of implementing an automated mHealth information privacy risk assessment tool, the challenges and obstacles.
5.2 Contributions	Outlining the contributions to research and practice this thesis has.
5.3 Limitations	Explanations in what way the research approach and execution was limited or constrained during this thesis.
5.4 Future Research	This section will provide hints to future researchers on how to continue or extend the current study in this thesis and further develop the implemented tool and its implications.
5.5 Conclusion	A critical reflection on the thesis. Were all the objectives reached and are the methods used coherent?

7. Expected Results

The results of this master thesis will be the implementation of the automated information privacy risk assessment tool. Part of the expected results will be the answer to the question, if static code analysis is a viable approach to assess information privacy risk factors of mHealth apps.

We expect the implementation process of the automated information privacy risk assessment to be at least partially feasible and implemented. Automating the downloading process of APK app files from the Google PlayStore is possible to a certain extent, and will be provided. The decompilation process is expected to gain at least a code recovery rate of above 90%.[31] Therefore, we expect the majority of APK files to be assessable. We expect the static code analysis for information privacy risk factors to be successful for the majority of factors and will provide insights into the possible degree of detail.

8. Problems

So far, there are no problems.

[31] See Enck et al. (2011), p. 5-6.

References

Baca, Carlsson, Lundberg (2008)

Dejan Baca, Bengt Carlsson, Lars Lundberg: Evaluating the cost reduction of static code analysis for software security. In: Proceedings of the third ACM SIGPLAN workshop on Programming languages and analysis for security - PLAS '08. 2008, p. 79

Bardas et al. (2010)

Alexandru G Bardas et al.: Static code analysis. In: Journal of Information Systems & Operations Management. No. 2, Vol. 4, 2010, pp. 99–107

Brüggemann et al. (2016)

Thomas Brüggemann, Joel Hansen, Tobias Dehling, Ali Sunyaev: "An Information Privacy Risk Index for mHealth Apps". Manuscript. 2016

Chen et al. (2012)

Connie Chen, David Haddad, Joshua Selsky, Julia E Hoffman, Richard L Kravitz, Deborah E Estrin, Ida Sim: Making sense of mobile health data: an open architecture to improve individual- and population-level health. In: Journal of medical Internet research. No. 4, Vol. 14, 2012, e112

Davidson, Pattee (2006)

Tal Davidson, Jim Pattee: Artistic Style. https://web.archive.org/web/20160228181142/http://astyle.sourceforge.net/, visited cached site on 02/28/2016

Dehling, Gao, Schneider, et al. (2015)

Tobias Dehling, Fangjian Gao, Stephan Schneider, Ali Sunyaev: Exploring the Far Side of Mobile Health: Information Security and Privacy of Mobile Health Apps on iOS and Android. In: JMIR mHealth and uHealth. No. 1, Vol. 3, 2015, e8

Dehling, Gao, Sunyaev (2014)

Tobias Dehling, Fangjian Gao, Ali Sunyaev: Assessment Instrument for Privacy Policy Content: Design and Evaluation of PPC. In: WISP 2014 Proceedings. 2014,

Enck et al. (2011)

William Enck, Damien Octeau, Patrick McDaniel, Swarat Chaudhuri: A Study of Android Application Security. In: Proceedings of the 20th USENIX Conference on Security. No. August, Vol. SEC'11, 2011, pp. 21–21

He et al. (2014)

Dongjing He, Muhammad Naveed, Carl A Gunter, Klara Nahrstedt: Security Concerns in Android mHealth Apps. In: AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium. Vol. 2014, 2014, pp. 645–54

Khalid et al. (2015)

Hammad Khalid, Emad Shihab, Meiyappan Nagappan, Ahmed E. Hassan: What Do Mobile App Users Complain About? In: IEEE Software. No. 3, Vol. 32, 2015, pp. 70–77

Kim et al. (2012)

Jinyung Kim, Yongho Yoon, Kwangkeun Yi, Junbum Shin: Scandal: Static Analyzer for Detecting Privacy Leaks in Android Applications. In: IEEE Workshop on Mobile Security Technologies (MoST). 2012,

Kweller (2010)

Abby Kweller: Jailbreaking and Unlocking the iPhone: The legal implications. In: Polymer Contents. No. 8, Vol. 27, 2010, pp. 480–523

Mcclurg (2012)

Jedidiah Mcclurg: Android Privacy Leak Detection via Dynamic Taint Analysis. In: . 2012,

Mitchell et al. (2013)

Stacy Mitchell, Scott Ridley, Christy Tharenos, Upkar Varshney, Ron Vetter, Ulku

Yaylacicegi: Investigating privacy and security challenges of mhealth applications. In: 19th Americas Conference on Information Systems, AMCIS 2013 - Hyperconnected World: Anything, Anywhere, Anytime. Vol. 3, 2013, pp. 2166–2174

Nolan (2012)

Godfrey Nolan: Decompiling Android. 1. Edition, New York 2012

Pan (2010)

Bob Pan: dex2jar. https://web.archive.org/web/20160222112240/https://github.com/pxb1988/dex2jar, visited cached site on 02/22/2016

Tumbleson, Wiśniewski (2010)

Connor Tumbleson, Ryszard Wiśniewski: Apktool. https://web.archive.org/web/20160222172049/https://github.com/iBotPeaches/Apktool, visited cached site on 02/22/2016

L. Xu (2013)

Liang Xu: “Techniques and Tools for Analyzing and Understanding Android Applications”. PhD thesis. 2013

W. Xu, Liu (2015)

Wenlong Xu, Yin Liu: mHealthApps: A Repository and Database of Mobile Health Apps. In: JMIR mHealth and uHealth. No. 1, Vol. 3, 2015, e28