Developing a Process to Test Privacy in Mobile Apps: Executive Summary

Carnegie Mellon University
Privacy Engineering Capstone Project, Spring 2018
Team Members: Sharada Boda
Sponsor: Philips
Advisor: Norman Sadeh

Regulations such as General Data Protection Regulation (GDPR) have made privacy testing paramount. The goal of the project was to develop a repeatable process to test mobile apps for privacy violations or aid in raising pertinent privacy questions. Considering the numerous dimensions of GDPR, the scope was limited to the following areas 'Purpose Identification', 'Third-party library Identification and their access', Consent and Portability. The scope was also limited to analyzing the data on mobile apps. The developed process does not cover the behavior of the data on the server.

The process combined automated tools and manual analysis. Several tools were used to analyze the apps and develop the process. The results from various tools and the code were manually analyzed. In most stages, the tools served to aid manual analysis. Research into various aspects showed that no single tool sufficiently analyzes the code. Rather, a combination of tools with manual analysis is required for comprehensive testing. The process while helping in identifying a few potential privacy violations, serves more to aid in helping privacy managers/officers to narrow-down possible areas of violation and work with developers to eliminate or resolve issues.

There are two types of analysis that can be conducted – static analysis and dynamic analysis. Static analysis involves analyzing the code or the apk file without running the app. The limitation of static analysis is that since it analyzes the entire code, the results may be too comprehensive to analyze easily. Most static analysis approaches also do not cover reflection APIs or obfuscated code. Reflection APIs are typically used to access code written in another language. Dynamic analysis involves analyzing the code while its running. The limitation of dynamic analysis is that all code flows must be covered. However, the results from dynamic analysis are easier to analyze as they are contextual and limited in scope. The tools used in the process made use of both static and dynamic analysis.

Prior to determining privacy violations, the crucial task of identifying personal data had to be first completed as GDPR is relevant only to personal data. Personal data in mobile apps can be generated using specific APIs in the code such as access to location, contacts, camera etc. This data collection most often requires the developers to update the Manifest file with corresponding permissions which results in the user getting a popup on his device requesting permission. To identify this data, the CMU MAPCS tool and the manifest file were analyzed (with an alternative of using the tool Inspeckage). Personal data can also be processed when a user enters data on a form. This method of collection does not result in the device generating permission popups and consent must be carefully collected for this data. Examples of this type of collection include the user entering the pregnancy due date, medical history, etc. This data was identified by interacting with the app and noting all the form entries.

The code for an Android mobile app often contains third-party libraries. These libraries usually consist of several functionalities. Some of these functionalities process personal data. However, the main app code may never call these functions and personal data may never be processed. In these cases, some of the tools generate false positives. Eliminating false positives was a vital step after using any tool. Verification of the results was usually a manual task.

GDPR also places special emphasis on data sent to third-parties. Therefore, the third-parties and the data that they are processing had to be identified. AdAppBrain tool was used to identify majority of the third-party libraries. These libraries included Ad Networks, Social Networks and developer libraries. The tool only identified the libraries but not what data the libraries were processing. The CMU MAPCS tool identified a subset of personal data that are being accessed by the third-party libraries. As explained earlier, the tools at times generate false positives. Therefore, the code was manually inspected to determine the location of API call and map the method calls to the actual app code if called from a third-party library. At times, due to the vast nature of the third-party libraries, this was not possible and in these circumstances verification from the developers was required.

Exported components in Android mobile apps provide additional access to third-party libraries. The Inspeckage was used to identify these components. However, the manifest file could also be used to identify exported components. The manifest file was further inspected to analyze the permissions that are provided to these exported components.
After identifying the personal data, third-party libraries and their access to personal data; and exported components, the behavior of this data was analyzed. This part of the process was mostly manual.

The identification process explained so far made use of manual analysis tools that utilized static analysis. The next part of the process which is analyzing the behavior of the data mostly involved manual analysis combined with dynamic analysis. The purpose of personal data, consent and portability was determined by manually interacting with the app and analyzing the code. In the case of third-party libraries, their purpose was determined by reading their description on the official website and their privacy policy, if available. The network traffic was inspected using the tool Fiddler. Any alternative proxy could also be used to inspect the network traffic. The network traffic was inspected to verify the use of encryption mechanisms when required and identify the data sent (especially third-party libraries).

In the final stage, the privacy policy text was analyzed. The CMU MAPCS tool identifies possible privacy policy omissions by mapping the APIs used in the code with the content written in the privacy policy. Some of these results contained false positives and were eliminated by manually inspecting the code. The CMU MAPCS tool while extendable to cover various other parameters, inspected a limited set of APIs at the time of conducting this project. The other data identified in the app as part of the previous two stages of the process were verified against the policy text.

The process overall consisted of three stages. The first stage involved identification of personal data processed in the app, third-party libraries and their access to personal data; and exported components. The second stage consisted of analyzing the purpose, consent and portability of the identified data. The final stage consisted of analyzing the policy text to ensure that they are no violations (e.g. processing the data when the policy denies processing) and no omissions (e.g. missing purposes).

The process used four tools – AdAppBrain, CMU MAPCS tool, Inspeckage and Fiddler. Two apps were tested using this process. As part of further work, use of FlowDroid was suggested to generate call graphics to ease manual analysis. Storage of personal data verification could also be further analyzed using Inspeckage.