# Investigating the Effectiveness of Obfuscation Against Android Application Reverse Engineering

Rowena Harrison
Supervisor: Prof. Keith Mayes

## ABSTRACT

**One particular threat facing Android is from repackaged applications; that is legitimate applications that have been reverse engineered, modified to include malicious code, repackaged, and then distributed for download.**
**Software developers wary of the reverse engineering process will obfuscate their code, trying to make it harder to read the code and understand its function.**
**This project investigates the effectiveness of obfuscation to counter Android application reverse engineering in practice by creating an app and testing the obfuscation offered by the tool ProGuard.**

## PROBLEM

Android app developers want a robust method to make the reverse engineering process harder so apps are not cloned and repackaged.

## SOLUTION

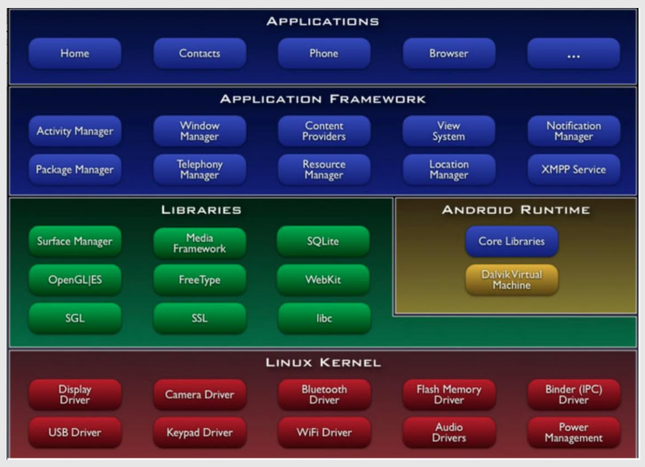
ProGuard obfuscation is an effective way to hinder the reverse engineering process, but it is far from foolproof.

## Project Objectives

- To create an Android application, with functionality to encrypt and store personal information.
- To reverse engineer this app using the tools ApkTool and dex2jar and compare their outputs to the application source code through heuristic analysis.
- To create an obfuscated version of the Android application and reverse engineer this obfuscated app with ApkTool and dex2jar.
- To compare the output of the tools on the obfuscated app to the output of the tools on the unobfuscated app through heuristic analysis, in order to determine the effectiveness of obfuscation to combat reverse engineering.

## Android

- The Android OS is made up of several layers; a Linux kernel, open source libraries, a runtime environment, an application framework with a Java interface and an application layer.
- The Android runtime environment's primary constituent is the Dalvik Virtual Machine (VM), which allows applications to be run on Android devices.

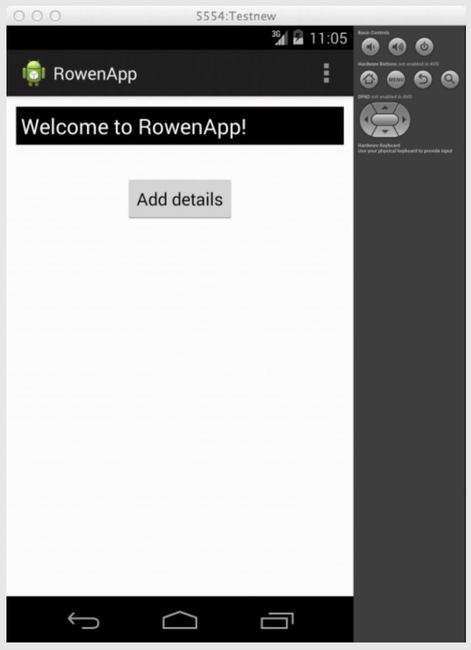



## Android Apps

- Android apps are written in the Java programming language and are stored as an Android package, or .apk file.
- The main files of an app include: the Android Manifest, app resources and the source files.
- Apps can be written using the Android Software Development Kit (SDK), which provides the relevant libraries and API's used to develop Android applications, along with an emulator.

## RowenApp

- The app created in this project, named RowenApp, is designed to encrypt and store personal banking information of the user or users in a database.
- The app can then find and retrieve user information when requested.




## ProGuard

- ProGuard, a free tool which is available as a stand alone product and within the Android SDK, does four things: shrinking, obfuscation, optimisation and pre-verification.
- ProGuard was applied to RowenApp to create the obfuscated app, RowenAppobsf.

## Reverse Engineering Tools

- Two tools were used in this project, ApkTool and dex2jar.
- ApkTool is able to disassemble Android applications into smali/baksmali, allow the user of the tool to debug and modify the code, and then repackage the application. The smali files which are the output of ApkTool can be read by various editors.
- dex2jar converts the Dalvik bytecode back into Java bytecode. This means the source code can be viewed in the Java format it was written in, using a graphical interface, JD-GUI.

## Results

- The outputs of ApkTool and dex2jar for RowenApp were compared, looking at features such as: syntax, variable names, context and conditional statements.
- The outputs from RowenApp were compared to those of RowenAppobsf, and found that class, method and variable names had been obfuscated an replaced with meaningless letters, as seen below.
- The context and purpose of the app was still able to be discovered by examining the code further.

```
byte[] arrayOfByte1 = c.a(locald.f());
byte[] arrayOfByte2 = c.a(locald.g());
char[] arrayOfChar = a(str1);
```

## Conclusions

- Both tools successfully reverse engineered both apps. Using the two tools together can provide the necessary means to provide a reasonable understanding of the app.
- The reverse engineered output of RowenAppobsf provided more of a challenge than that of RowenApp; requiring a more extensive analysis and took several more hours to understand the nature of the app.
- ProGuard was effective in hindering the reverse engineering process; however since the purpose of the app was still able to be found, this tool is not infallible.
- More test apps, containing all types of computing logic and app functionality, could be produced in order to test obfuscation tools' effectiveness against reverse engineering.
- This would give insight into how these tools obfuscate different pieces of code, and provide input for their improvement.