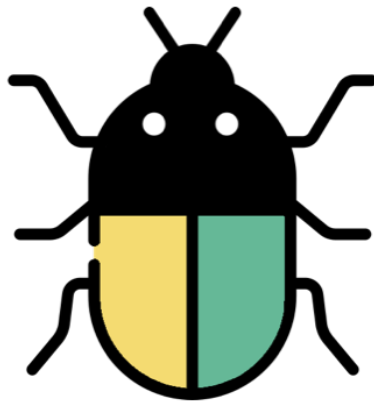


Beetlebug: A very insecure Android App

For Developers, Pentesters and
Mobile Application Security Enthusiasts

Beetlebug



Beetlebug v1.0

Author & Developer: Hafiz Aziz

Email: contact@hafiz.ng

Beetlebug v1.0

CTF Challenges

1. **Hardcoded Secrets**
2. **Insecure Data Storage**
3. **Vulnerable Activities, Services & Content Providers**
4. **Insecure WebViews**
5. **Sensitive Information Disclosure**
6. **Fingerprint Bypass**
7. **Firebase Database Misconfiguration**
8. **Jailbreak Detection**
9. **SQL Injection**
10. **Input Validation (XSS)**

Disclaimer

This walkthrough will contain spoilers. Therefore, I urge readers to attempt the CTF before reading this writeup. You will learn more by first attempting it yourself and only reference this writeup when you find yourself struggling with a challenge.

How to Play

Beetlebug is a beginner-friendly Capture the Flag Android application that aims to inspire interest in Mobile Application Security. To complete the challenges, the CTF player will have to find all the flags within the application and other resources. Each CTF flag is preceded with a 0x followed by 7-digit alphanumeric characters (e.g., 0x342!834).

Beetlebug tracks the user's progress, flag completion state, and gives the user hints to assist in solving the challenges.

Tools

To complete the CTF challenges, you will need these tools to get you started. Please make reference to Lab Setup Guide included in the root folder of [Beetlebug Github](#) repository to guide you in setting up your environment.

Must have

[Android Debug Bridge \(adb\)](#) - This is command line tool that lets you communicate with the device or emulator. It has all sorts of commands that you can use to communicate with the device like installing and debugging apps, providing access to a Unix shell on the device. We can find the adb tool in the platform-tools directory of the android sdk installation directory.

[SQLite3](#) - Android uses the SQLite database engine, a self-contained, transactional database engine that requires no separate server process. This tool allows you to browse table contents, run SQL commands, and perform other useful functions on SQLite databases. We can find the tool in the platform-tools directory of the android sdk installation directory.

[Jadx](#) - A command line and graphics user interface tool for producing Java source code from Android .dex and .apk files.

Nice to have

[DB Browser for SQLite](#) – is a visual, open source tool to create, design, and edit database files compatible with SQLite.

[Drozer](#) – Drozer allows you to search for security vulnerabilities in apps and devices by assuming the role of an app and interacting other apps' IPC endpoints and the underlying OS. It makes it easy for a tester to create test cases and check for possible vulnerabilities in the components of an application.

[MobSF](#) – Mobile Security Framework (MobSF) is an automated, all-in-one mobile application (Android/iOS/Windows) security assessment framework used for performing static and dynamic analysis.

Decompiling Android Apps

Why should I decompile an app

There can be several reasons why you might want to decompile an app. I will mention some here:

1. Simply because you are curious and want to see what the source code looks like (at least roughly, because decompiling won't give you exactly the same source code)
2. To see if there are any security problems in your app, like keys written directly as strings in the source code
3. To find a way to utilize undocumented functions
4. To find some secrets

Let us now move to decompiling the Beetlebug APK file. We are going to use Jadx, the command line and GUI tool to produce Java Source code from Android DEX and APK files.

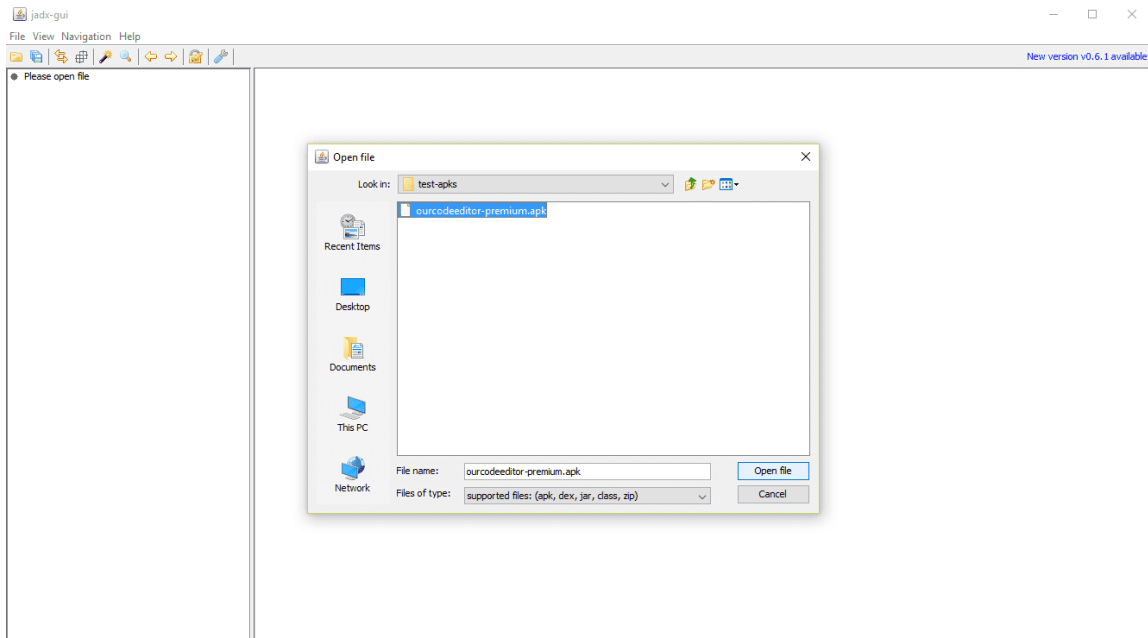
Tools used

- JADX
- MobSF

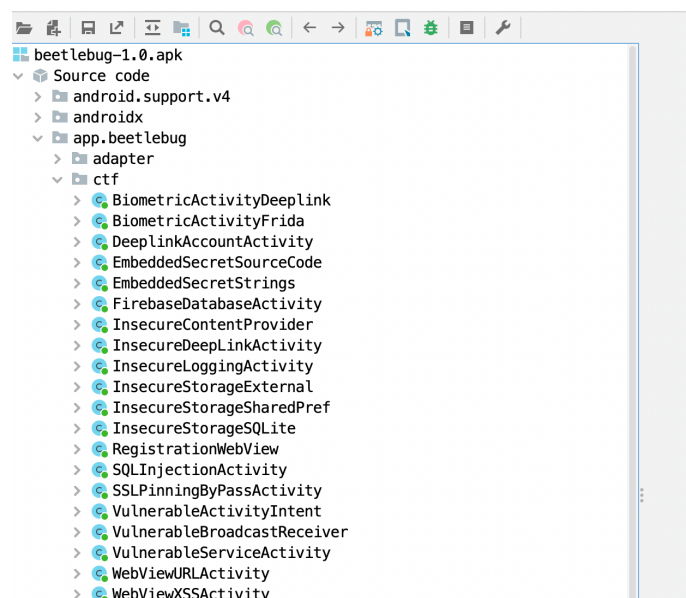
How to decompile APK using JADX

1. Pull the .apk file from your device. You can use a tool like the [APK Backup and Extractor](#) tool or the `adb pull` command of the `adb` tool.

2. jadx is generally used with a graphical user interface, start the `jadx-gui.bat` file (inside the `jadx-folder/bin` folder) with administrator rights. As first you will get a window that allows you to choose the APK file that you want to decompile:

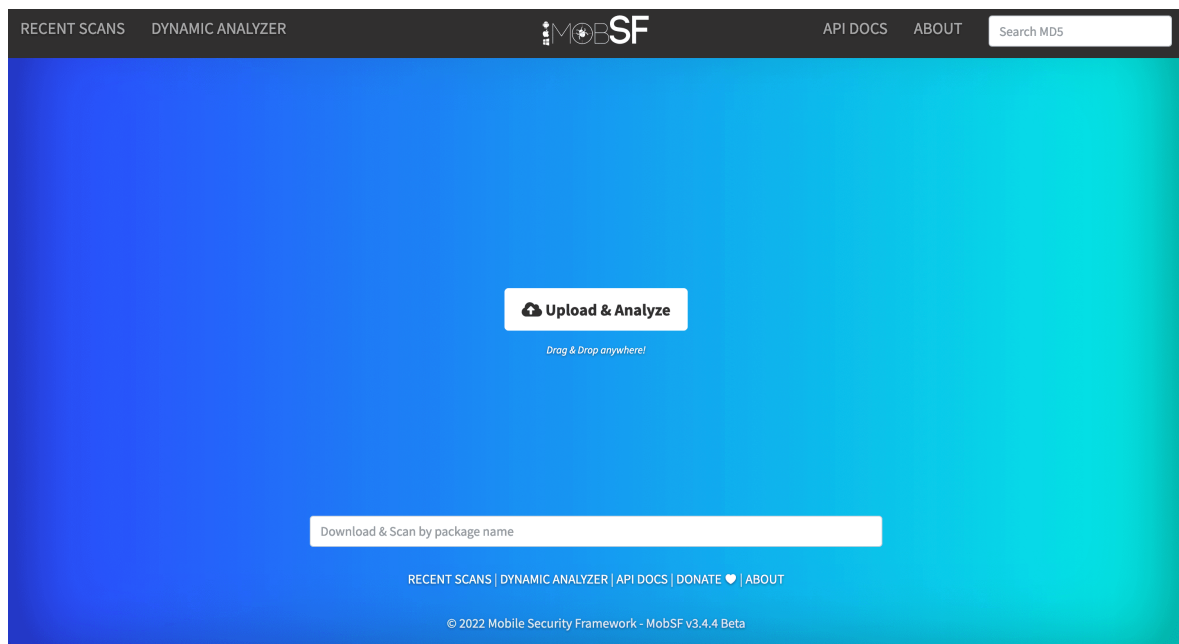


3. Once you've selected the file, jadx will decompile it and will provide an explorer IDE-style window at the left side of the application that lists all the Java packages and files of the APK.




Reverse Engineering with MobSF

1. Navigate to the MobSF directory and `./run.sh`.
2. Point your web browser to <http://127.0.0.1:8000>. By default, MobSF listens at port 8000.



MobSF Static Analyzer interface

3. Drag your APK file to the MobSF web interface to begin analysis.
4. The process will automatically run, then present the results after completion as shown below:



APP SCORES

Average CVSS 6.6

Security Score 70/100

Trackers Detection 0/427

FILE INFORMATION

File Name base.apk

Size 9.63MB

MD5 c02e9f5ed5c60450d2fae43891369efd

SHA1 91671c526a3ee6dd10e790634417382f278a7c9b

SHA256 30c266a27ced4cc6b1aa712e943381372f9d3bc3b126c3e688b562d8db4981d3

APP INFORMATION

App Name Beetlebug

Package Name app.beetlebug

Main Activity app.beetlebug.Walkthrough

Target SDK 30 Min SDK 23 Max SDK

Android Version Name 1.0 Android Version Code 1

29

ACTIVITIES

View

2

SERVICES

View

1

RECEIVERS

View

3

PROVIDERS

View

Exported Activities

10

Exported Services

1

Exported Receivers

1

Exported Providers

2

- From the output, you can also see the **Androidmanifest.xml** file and decompiled Java source code of the application.

SCAN OPTIONS

Rescan

Start Dynamic Analysis

DECOMPILED CODE

View AndroidManifest.xml View Source View Smali

Download Java Code Download Smali Code Download APK

To be continued...