Reversing an Android sample which uses Flutter

cryptax.medium.com/reversing-an-android-sample-which-uses-flutter-23c3ff04b847

@cryptax May 12, 2022

@cryptax

May 12

.

6 min read

Flutter is a framework able to build **multi-platform** apps (e.g. iOS and Android) from a *single* code base. The **same source code** is able to generate an iOS app, and/or an Android app, which is extremely convenient from a developer's perspective. In the case of Flutter, the source code is written in , and the apps are *natively* compiled. For reverse engineering, this is an issue, because (1) we don't have good Dart decompilers, and (2) native apps are usually more difficult to reverse than Dalvik ones.

Thanks to @Hexe and @U+039b for their assistance & enthusiasm on this work.

A few days ago, I got *enticed* into reversing a possibly malicious Android sample, which was posing as an app for the French national health system ("Ameli"). Its hash is `171b326ba772e0c15558679ab3bfe88a55d99b70978a4c0c6b60f66c025585eb`.

There are 2 different parts to this blog post:

1. . Read this part if you want to hear how I struggled to reverse the app.
2. . Read this part to learn if the app is malicious or not.

Reverse engineering Flutter-based Android apps

How do I detect the app uses Flutter?

There are two cases.

- If the app is in , you are lucky. Unzip the APK and look for the code in `./assets/flutter_assets/kernel_blob.bin` [1]
- If the app is in (which is the case for the suspicious sample), you will find `libflutter.so` in `./lib/` subdirectories.

I have added **Flutter detection to** . In addition, DroidLysis detects use of **Andromo**. Andromo is a "*No-Code iOS and Android native apps development platform [..] backed by Google Flutter*". This is probably how the app was written: the developer used Andromo, which uses Flutter.

```
Kit properties /  Detected 3rd party SDKs
andromo              : True (Andromo - No code Android app
appcompat            : True (AndroidX Support Library)
chromium             : True (Support library boundary)
flutter              : True (Flutter or its plugins)
googleads            : True
```

Where is the Dart code?

Don't waste your time reversing the dalvik bytecode. I personally wasted time decrypting a Base64 AES encrypted dynamically loaded DEX… and landed inside Google Ads…

In Flutter release apps, the app payload is located in `./lib/<platform>/libapp.so` [2].

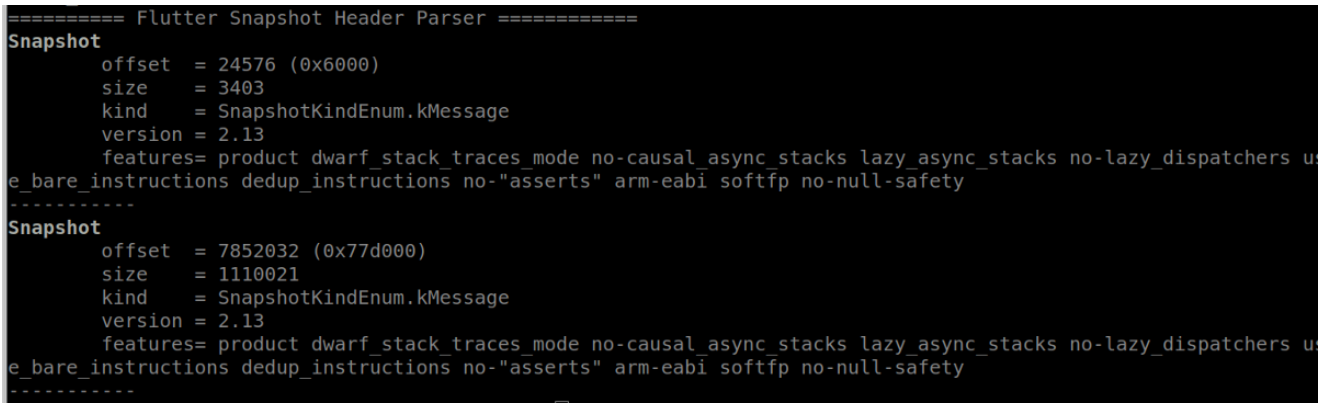
Dart code runs in an "isolate", which is a structure which contains a heap, references to objects etc. Isolates are *isolated* 😉 and can't access each other, except one special isolate, the "VM" isolate that everybody can access [3].

When we inspect `libapp.so`, we see a VM "snapshot" and an isolate snapshot. Snapshots are the serialized state of an isolate, frozen at a given moment.

```
Symbol table '.dynsym' contains 6 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     0: 00000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 00001000    32 FUNC    GLOBAL DEFAULT    1 _kDartSnapshotBuildId
     2: 00002000 13368 FUNC    GLOBAL DEFAULT    3 _kDartVmSnapshotInstructi
     3: 00006000 23024 FUNC    GLOBAL DEFAULT    4 _kDartVmSnapshotData
     4: 0000c000 0x770310 FUNC    GLOBAL DEFAULT    5 _kDartIsolateSnapshotInst
     5: 0077d000 0x207100 FUNC    GLOBAL DEFAULT    6 _kDartIsolateSnapshotData
```

Output of : readelf -s libapp.so

So, more precisely, the Dart code of the app is `_kDartIsolateSnapshotInstruction` (and Data). The format of snapshots is explained in [2,4] and I have written a Python tool to parse snapshot headers.



Parsing the snapshots of the app. The first one is the VM isolate. Both use version 2.13.

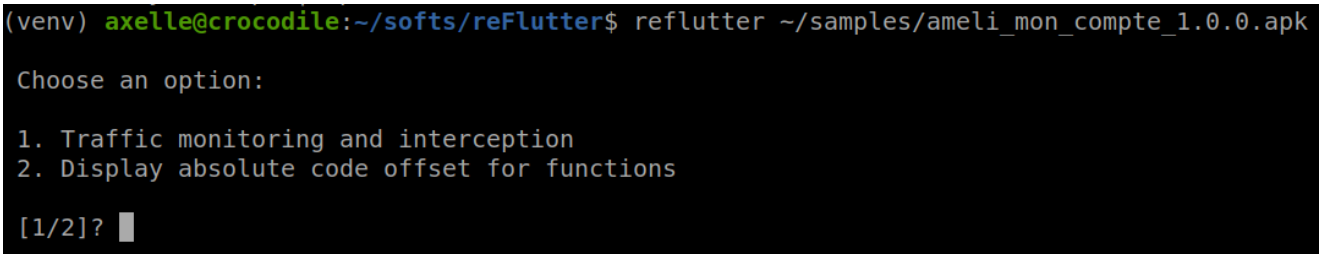
Tools to reverse Dart

There are basically 3 tools to reverse Dart:

- [5]: this is a Python toolkit to parse `libapp.so`. It works for Flutter 2.5. Example of use . Unfortunately, we have.
- [6]: this tool is meant to parse `libapp.so` and dump all classes of the isolate snapshots. Exactly what I am looking for, except it works for Flutter 2.10. There's. It isn't finished yet. I tried to fix errors for my sample, by quickly moving out of issues it encountered, but I got no interesting decompiled output in the end (meaning my "quick fix" is too quick, and there's more to be done to get it to work).
- [7]: this framework operates differently. The idea is to patch the sample and use a patched version of the Flutter library. Then, to write Frida hooks and dynamically analyze calls to the patched library.

To reFlutter … or not

Patching the application is easy: run `reFlutter` to generate the patched application (select option 2), then align it (`zipalign`) and sign it (`apksigner`).



Patching the sample with reFlutter — select option 2 for dynamic analysis of the sample
Then, run the application and get a dump in `/data/data/<PACKAGE>/dump.dart`. This is where I had read the project description too fast. I thought I'd get the decompiled dart code. No. We get code offsets to use in Frida hooks.

For example, the dump provides the following offset for `get:zra`.

```
Function 'get:zra': getter const. null {

Code Offset: _kDartIsolateSnapshotInstructions + 0x000000000000c1a4

       }
```

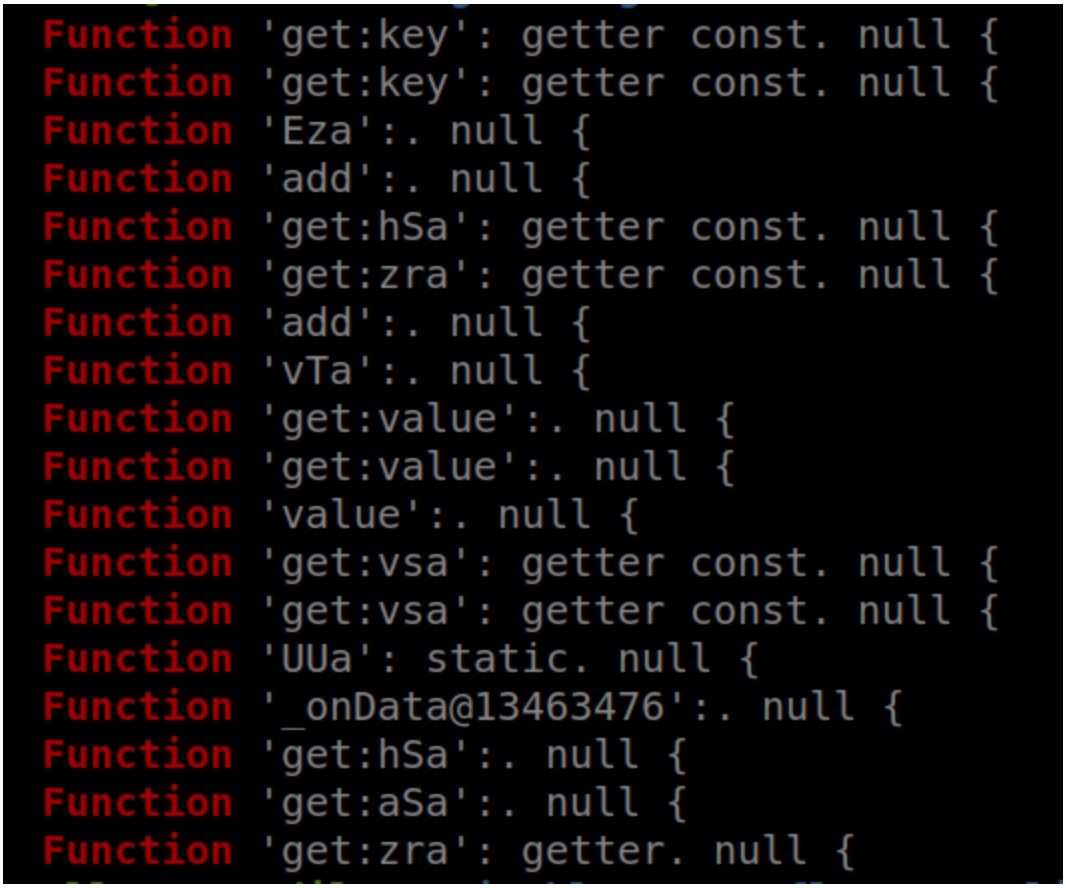
To hook it, I'll need to customize the reFlutter Frida hook with the correct offset:

```
function hookFunc() {    // _kDartIsolateSnapshotInstructions (c000) + code offset
(c1a4)    var dumpOffset = '0x181a4'     var argBufferSize = 150    var address =
Module.findBaseAddress('libapp.so')     console.log('\n\nbaseAddress: ' +
address.toString())...
```

This is not very handy: I don't know which are the interesting functions, so I'd need to hook them all. Actually, the issue is I don't want *dynamic* analysis at this stage, but *static* analysis ("static analysis ruleZ" is my favorite sentence!). My bad, I should have understood this before trying to use reFlutter! :-(Finally, to my understanding, reFlutter is similar (or the same?) as [8] which recompiles the Flutter engine located in `libflutter.so` to print debug messages.

Analyzing the reFlutter dump

Even if the reFlutter dump does not contain what I am after (Dart decompiled code), it nevertheless contains *valuable information*: the **list of all libraries, classes, objects and functions.**



This is the list of methods of the internal Dart:_http library
Unfortunately, if we rule out all standard Dart libraries (_http, core, ffi, developer, io…), we are left with … obfuscated library names (`aAf` , `AAf` , `aof` …) and obfuscated function names.

```
$ grep -A 3 "Library:'cuf'" dump.dart.2 | grep Function
  Function 'Wkd': static. ({required int PVb, required vJ nq}) => bool {
  Function '_Ukd@620222615': static. null {
```

Obfuscated functions names of library “cuf”
I assume this is because the app was built using Andromo. In theory, if we were lucky, the class/function names could help us understand what the sample does.

Malware analyst’s angle

So, the fact is **we don’t have any good tool to decompile Dart** from `libapp.so` **+ the sample has some form of obfuscation**. We’re not totally done yet: `strings` outputs interesting strings.

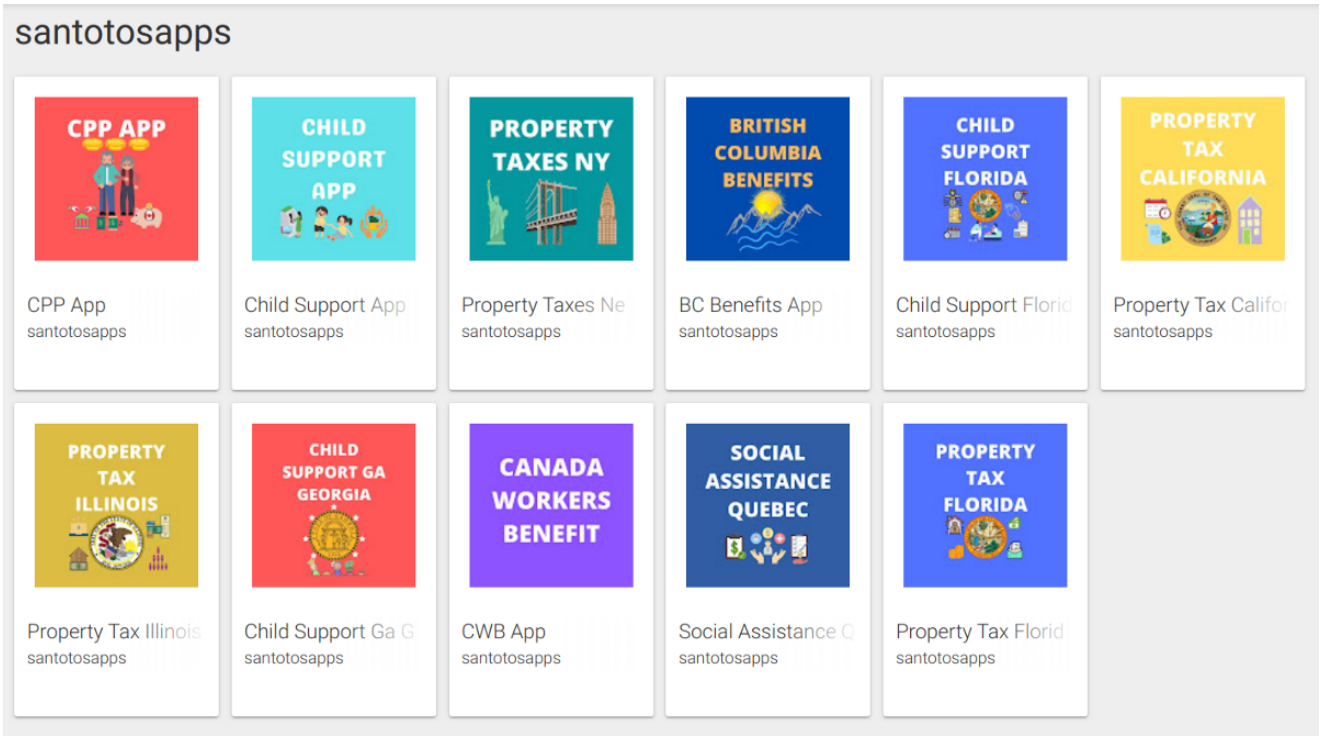
```
$https://amelimoncompte.blogspot.com/8
Whttps://amelimoncompte.blogspot.com/2021/10/certificat-et-code-daffiliation.html
https://stackoverflow.com/questions/tagged/flutter
rxWhttps://amelimoncompte.blogspot.com/2021/10/debloquer-mon-compte-ameli.html
  https://api.flutter.dev/flutter/material/Scaffold/of.html
  https://api.flutter.dev/flutter/dart-ui/ChannelBuffers-class.html
CFhttps://builder.andromo.com/hub/6cb42c3646917ed5db25d9241cd1e7fa/
Hhttps://amelimoncompte.blogspot.com/2021/06/ameli-compte.html
```

Notice the URLs going to amelimoncompte[.]blogpost[.]com
The sample seems **nothing more than a front-end** to website https://amelimoncompte[.]blogpost[.]com. We have URLs such as https://amelimoncompte[.]blogspot[.]com/2021/06/se-connecter.html (“se connecter” = login) which might have us fear the website tries to phish national health credentials. At least today, *this is not the case*: a link redirects to https://lescertificats[.]net/ameli-mon-compte/ which in turn redirects to the official website of French national health system.

What’s the goal?

So, the sample is a somewhat **useless but non malicious** front-end to understand or head to the French national health system. What’s the point in creating such an app?!

Another URL in the sample links to the developer of the apps: https://builder[.]andromo[.]com/hub/6cb42c3646917ed5db25d9241cd1e7fa/ points to “santotosapps”. This **developer creates several other similar apps** on the Play Store.



Applications developed by “santotosapps” in Google Play Store. Notice how each app look alike: a large rectangular icon with simple upper case font.
Those apps also look like front-end to official websites. They don’t seem malicious (yet), but use is unclear. So, once again, what’s the point?

The only reason I can see is advertisement. Each of these apps come packaged with Google Ads. Perhaps they will generate some income to the developer if enough people download them?

Conclusion: it’s not malicious — from a malware analyst’s point of view — but my personal advice is to keep away from such apps.

— the Crypto Girl

References

- [1]
- [2]
- [3]
- [4] and part 2:
- [5]
- [6]
- [7]
- [8]

My personal additions:

- in Android apps within DroidLysis:
- :