The complexity of reversing Flutter applications

Axelle Apvrille, Fortinet

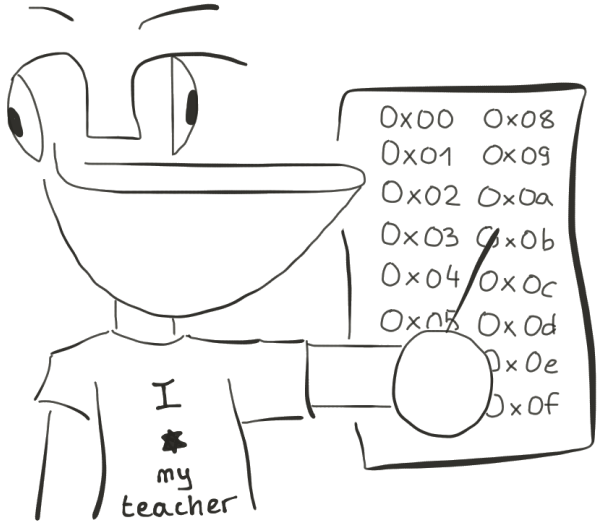
Nullcon, March 2024

Who am I?

Axelle Apvrille

Principal Security Researcher at **Fortinet**, @cryptax
Lead organizer of **Ph0wn CTF**
I analyze **Android malware** and **IoT** malware

Goal of this talk



Understand how to reverse Flutter applications with a special focus on Android malware

sub-goal: solve GreHack CTF 2023 Dart challenge

Dart is an **object**-oriented programming language with a **C-style** syntax

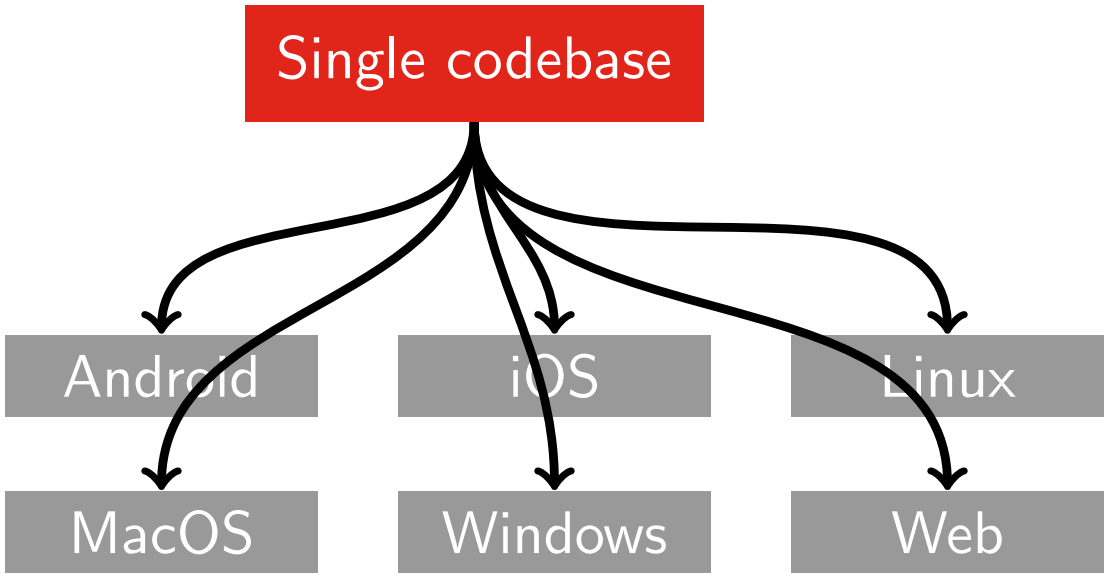
```
class Hello {
  void sayHello() {
    print("Hello Nullcon!");
  }
}

void main() {
  var hello = Hello();
  hello.sayHello();
}
```

Dart: 4 output formats

Output format	Size	Time
Self contained	5,919,728	0,006s
AOT snapshot	873,440 **14%**	0,012s **2x**
JIT snapshot	4,924,048 **83%**	0,333s **55x**
Kernel snapshot	1968 **0.03%**	0,411s **68x**

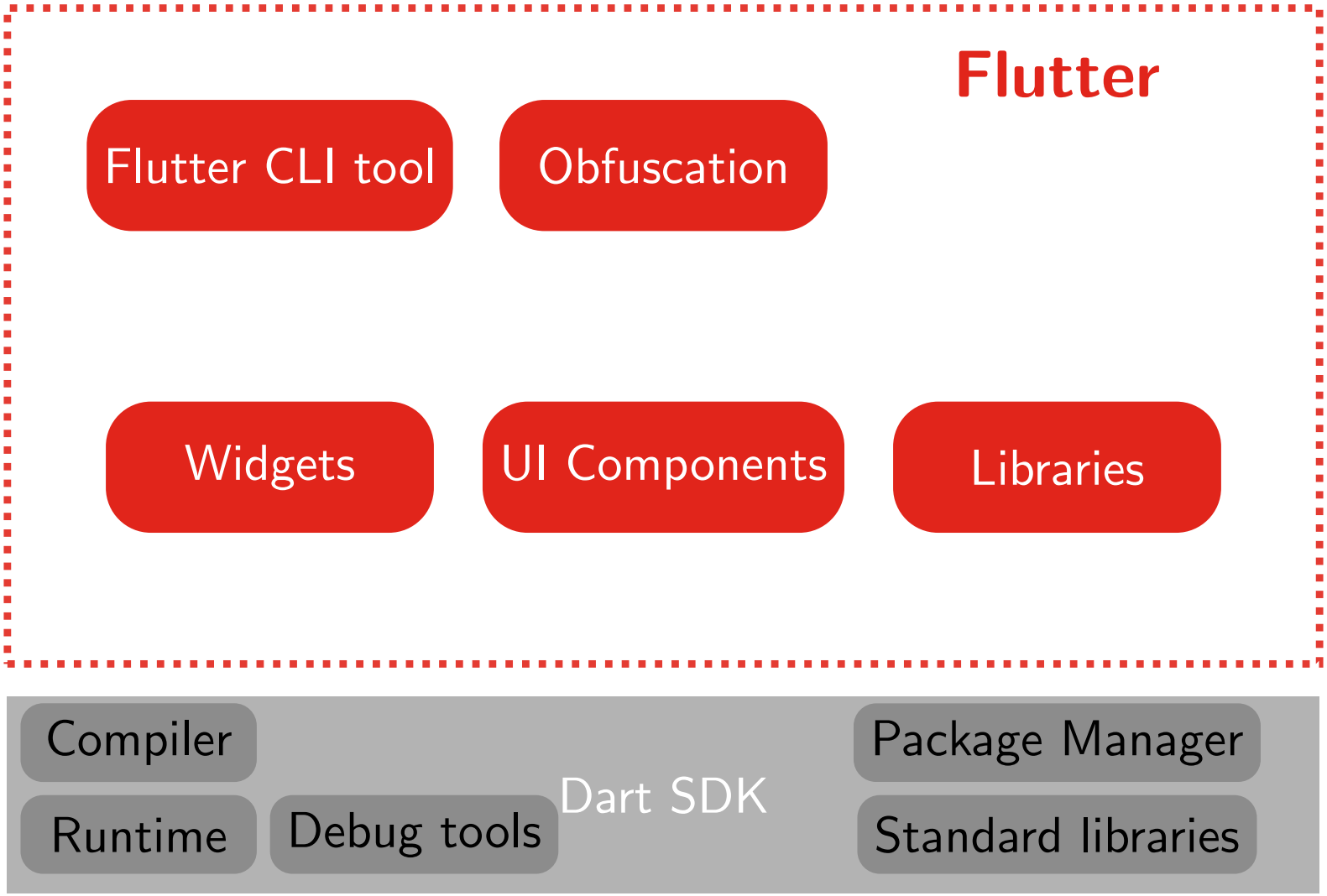
```
dart compile exe|aot-snapshot|jit-snapshot|kernel file.dart
```

Dart can be natively compiled for multiple platforms



Dart	**Java**
Native machine code Android and iOS: apps bundled with a Dart VM runtime	Byte code Android and iOS: JVM for mobile exists but primarily for dev and testing.

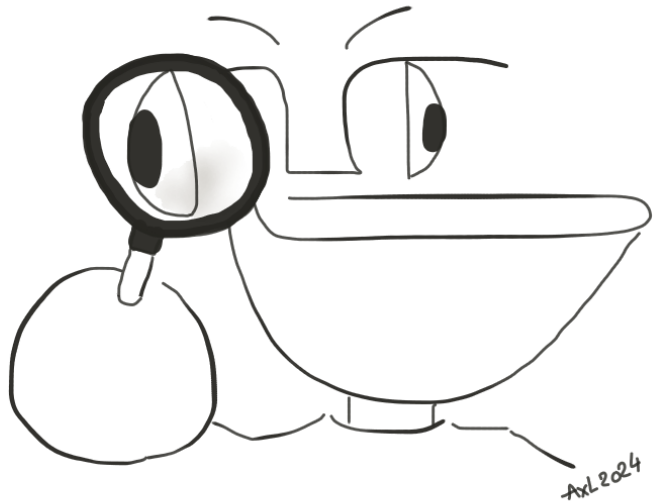
Flutter uses the Dart language and SDK



Flutter output types

Output type	**Speed**	**Comments**
Kernel snapshot	Slow	Flutter **Debug** builds. Portable. Easy to reverse
~~JIT snapshot~~		*Not used in Flutter*
AOT snapshot	Fast	Flutter **Release** builds. **Compiled Natively**. Difficult to reverse
~~Self contained~~		*Not used in Flutter*

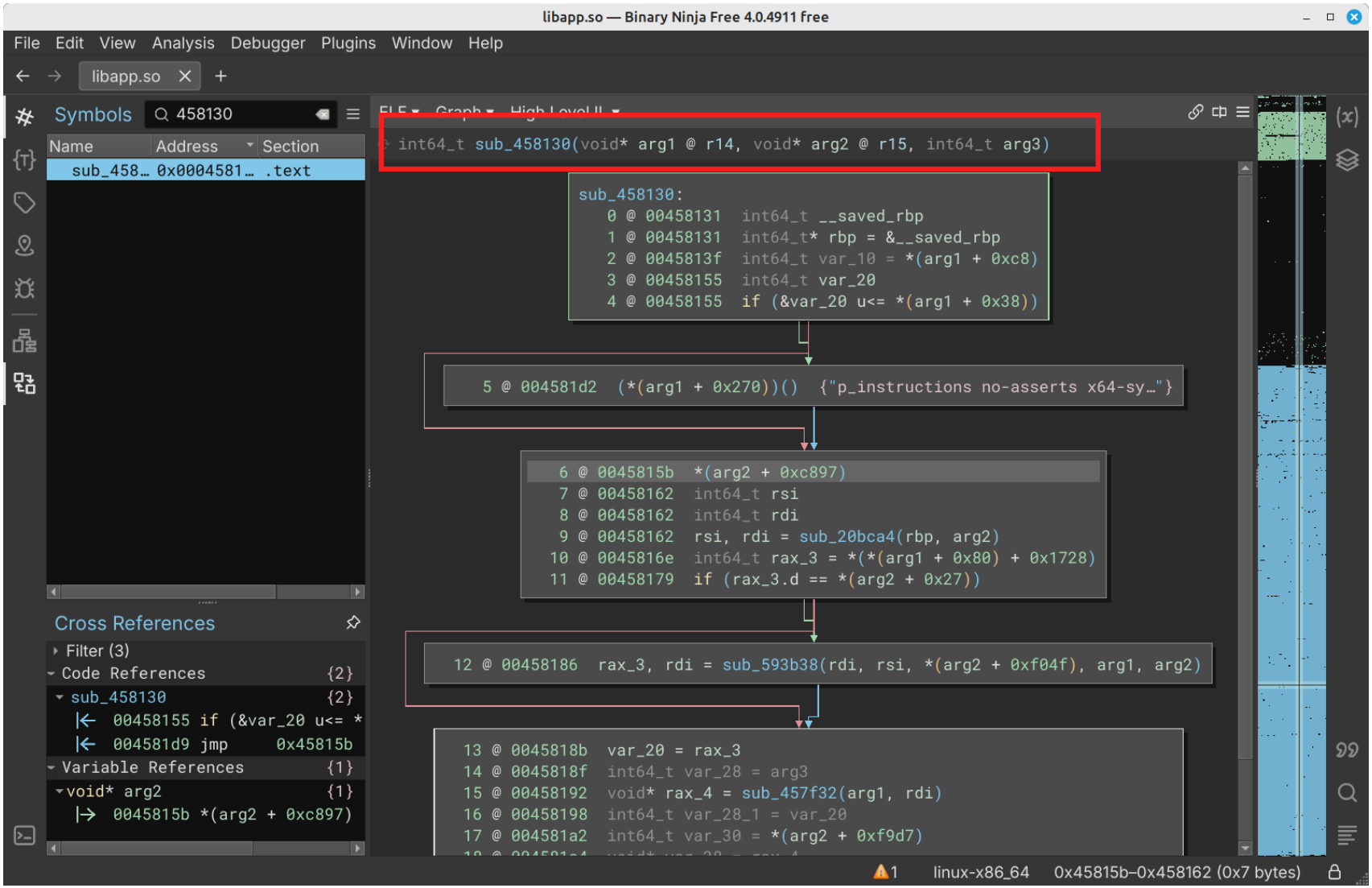
Focus



1. Understand how to reverse **Flutter** applications for Android, especially malware. **Release** applications $\rightarrow$ **Dart AOT snapshot**.
2. Solve **GreHack CTF 2023 Dart challenge** $\rightarrow$ It's a **Dart AOT snapshot**

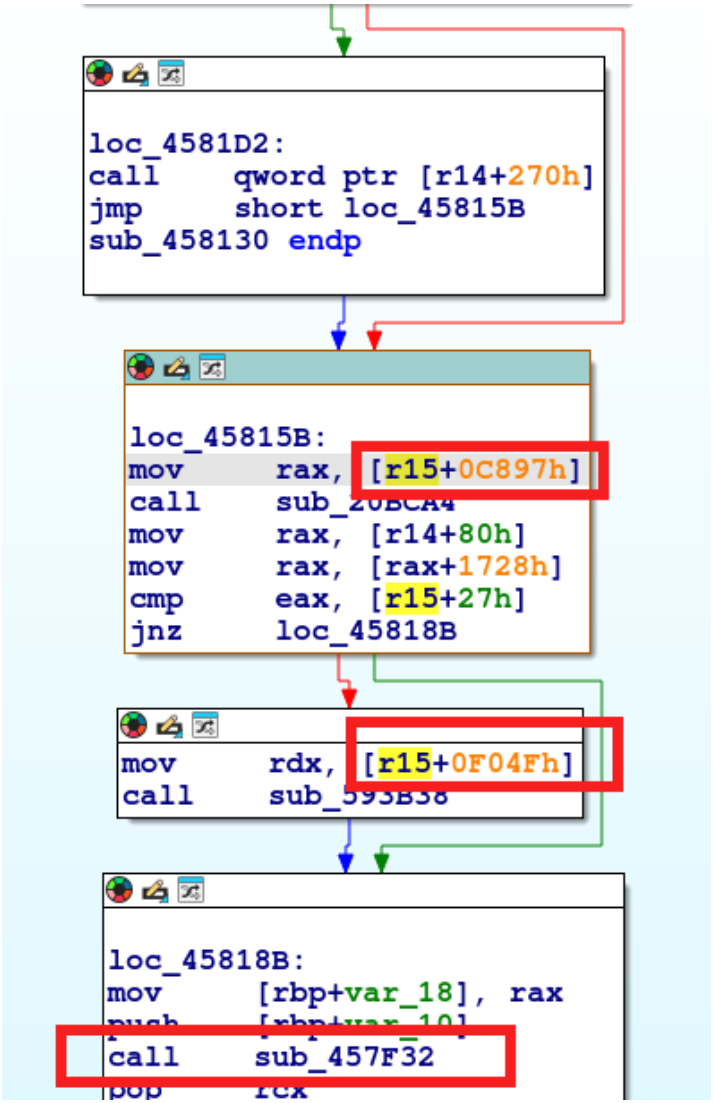
Let's focus on **Dart AOT snapshots**

Disassemblers do not support AOT snapshots



No function name, wrong arguments for the function

Disassemblers do not support AOT snapshots



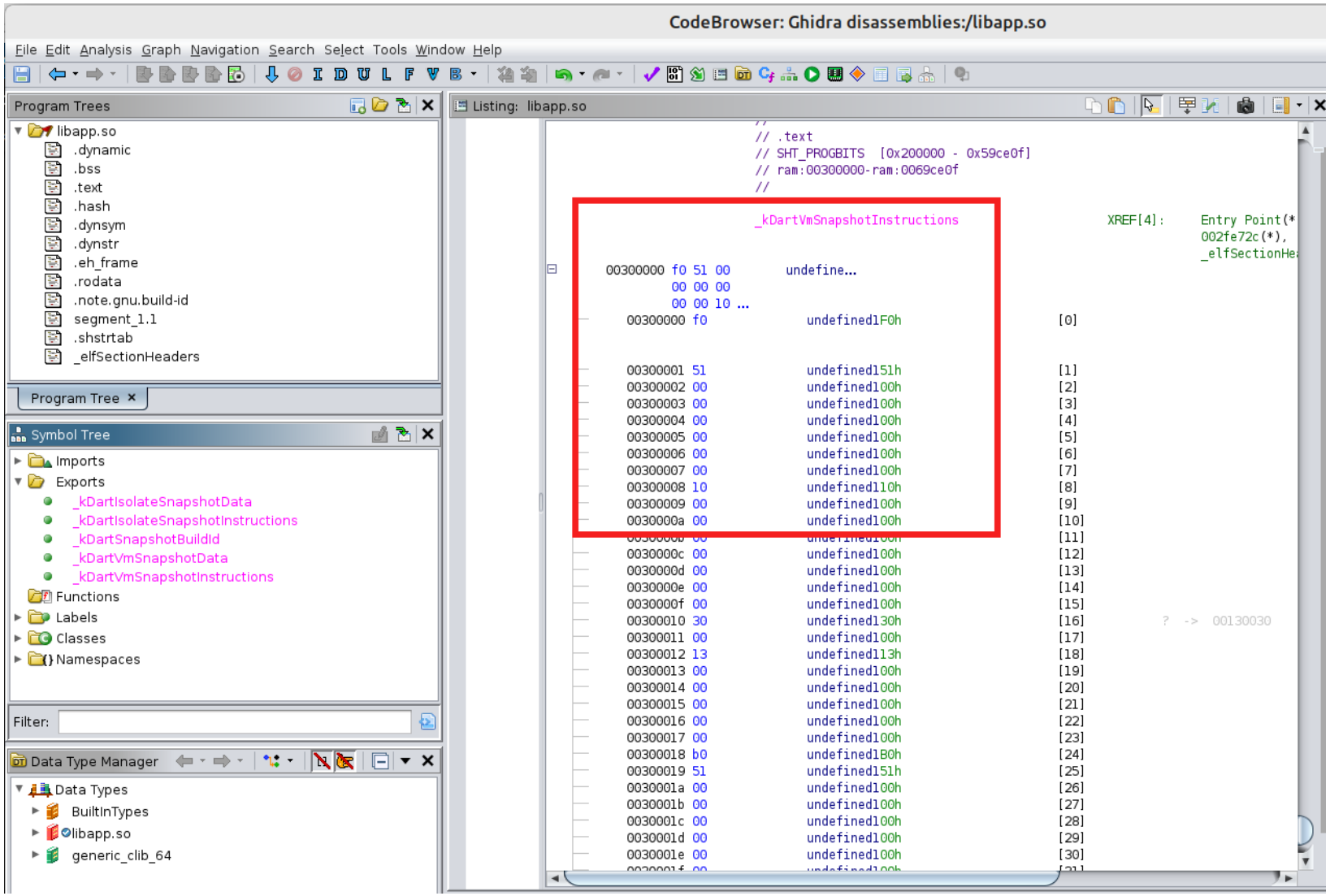
No strings, no literals, no function names

Disassemblers do not support AOT snapshots

```
0x00458131      4889e5          mov rbp, rsp
0x00458134      4883ec18        sub rsp, 0x18
0x00458138      498b86c80000.   mov rax, qword [r14 + 0xc8]
0x0045813f      488945f8        mov qword [var_8h], rax
0x00458143      33c0            xor eax, eax
0x00458145      4863c0          movsxd rax, eax
0x00458148      488b4c8510      mov rcx, qword [rbp + rax*4 + 0
0x0045814d      48894df0        mov qword [var_10h], rcx
0x00458151      493b6638        cmp rsp, qword [r14 + 0x38]
0x00458155      0f8677000000    jbe 0x4581d2
; CODE XREF from fcn.00458130 @ 0x4581d9(x)
0x0045815b      498b8797c800.   mov rax, qword [r15 + 0xc897]
0x00458162      e83d3bdbff      call fcn.0020bca4
0x00458167      498b86800000.   mov rax, qword [r14 + 0x80]
0x0045816e      488b80281700.   mov rax, qword [rax + 0x1728]
0x00458175      413b4727        cmp eax, dword [r15 + 0x27]
0x00458179      0f850c000000    jne 0x45818b
0x0045817f      498b974ff000.   mov rdx, qword [r15 + 0xf04f]
0x00458186      e8adb91300      call fcn.00593b38
; CODE XREF from fcn.00458130 @ 0x458179(x)
0x0045818b      488945e8        mov qword [var_18h], rax
```

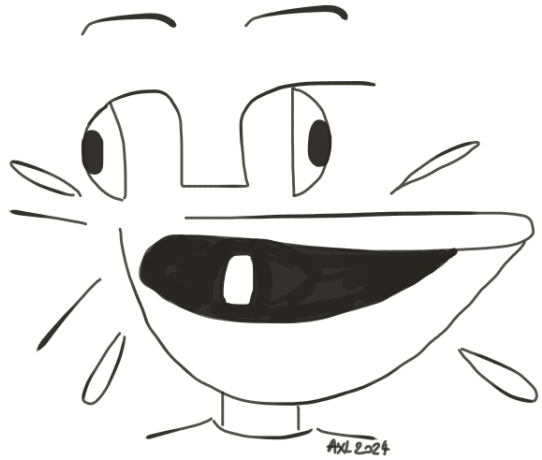
No strings, no literals, no function names

Disassemblers do not support AOT snapshots



Bad entry point, Completely lost

Dart assembly defines its own registers!



	x86_64	**Aarch32**	**Aarch64**
Stack Pointer (SP)	R4	R14	**X15** (custom)
Object Pool (PP)	**R15**	**R5**	**X27**
VM Thread (THR)	**R14**	**R10**	**X26**

Documentation is ... the code

https://github.com/dart-lang/sdk/

```
enum Register {
...
  R5  = 5,  // PP
  R6  = 6,  // CODE_REG
  R7  = 7,  // FP on iOS, DISPATCH_TABLE_REG on non-iOS (AOT only)
  R8  = 8,
  R9  = 9,
  R10 = 10,  // THR
  R11 = 11,  // FP on non-iOS, DISPATCH_TABLE_REG on iOS (AOT only)
  R12 = 12,  // IP aka TMP
  R13 = 13,  // SP
  R14 = 14,  // LR
  R15 = 15,  // PC
```

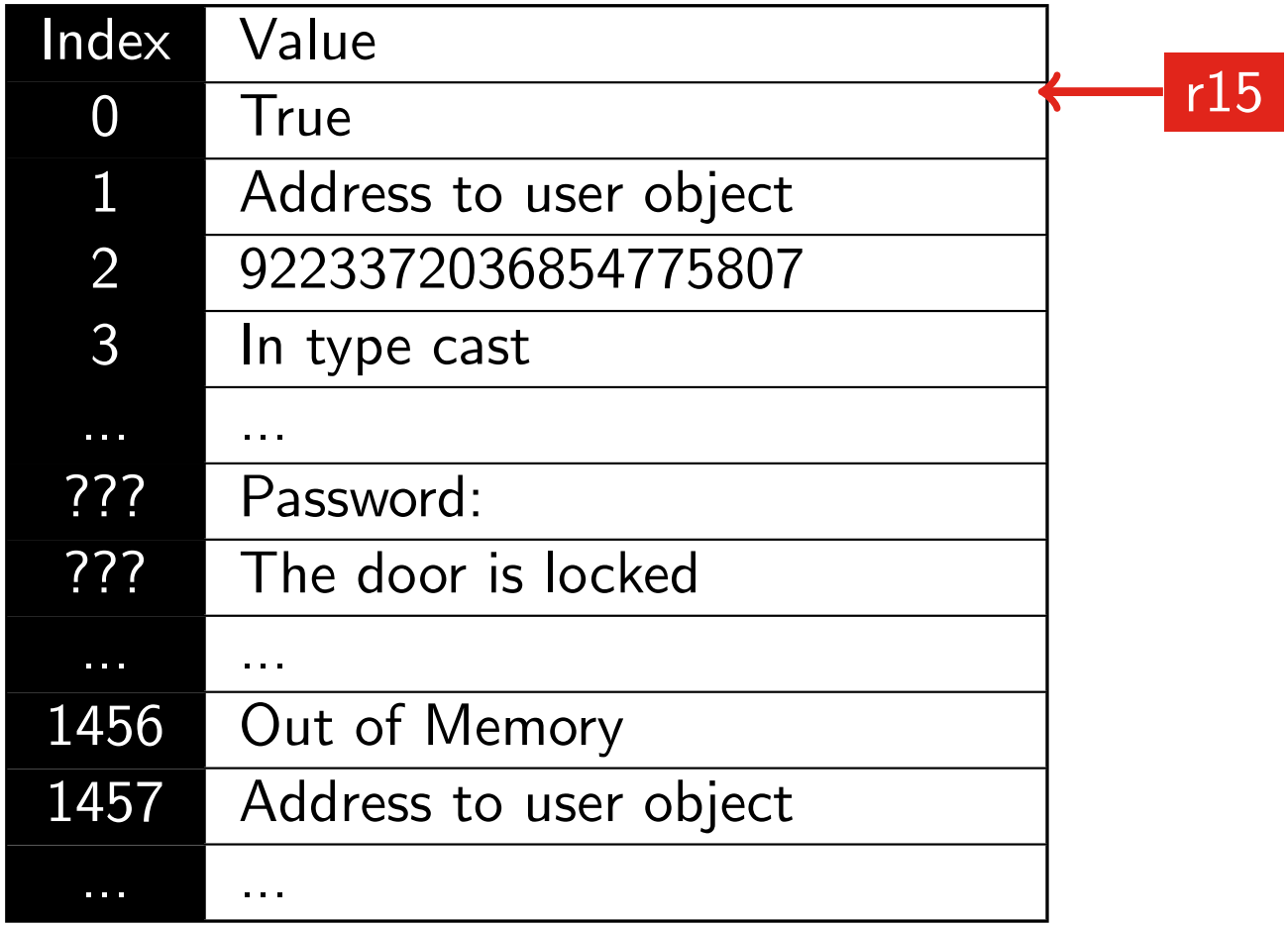
https://github.com/dart-lang/sdk/blob/main/runtime/vm/constants_arm.h

Example of Function Prologue for Aarch64

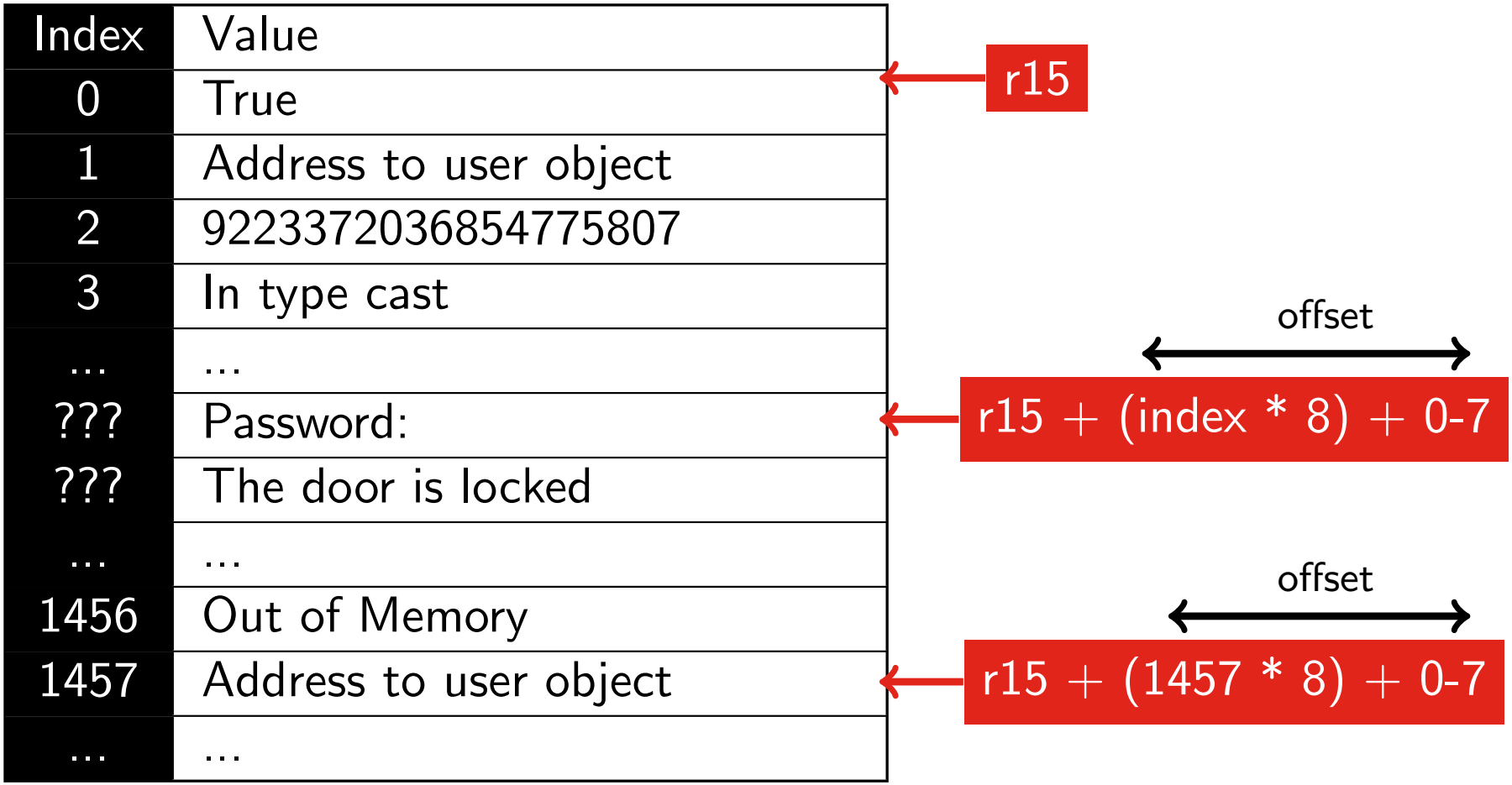
```
; push frame pointer and link register on the stack
STP         X29, X30, [ X15 , #FFFFFFF0h]!
; update frame pointer
MOV         X29, X15
; allocate 16 bytes on the stack
SUB         X15, X15, #10h
; stack overflow check
LDR         X16, [ X26 , #38h]
CMP         X15, X16
B.LS        loc_3D75DC
```

- **X15**: custom stack pointer for AAarch64
- **X26**: holds a pointer to the current thread

Dart Object Pool



Dart Object Pool

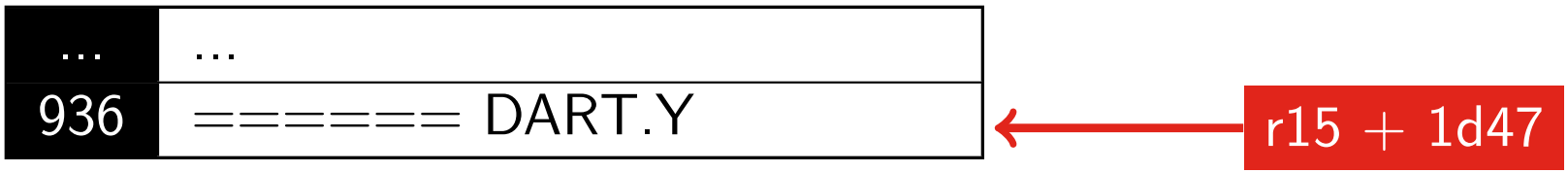


offset = (**index** * 8) + (0-7)

index = offset // 8

Examples of access to the Object Pool

```
; x86-64
mov       r11, qword ptr ds:[r15+1D47h]
```



Big indexes are computed - example on Aarch64

```
; loads object pool + 0x0F038
ADD       X17, X27, #Fh, LSL #12
LDR       X17, [X17, #38h]
```

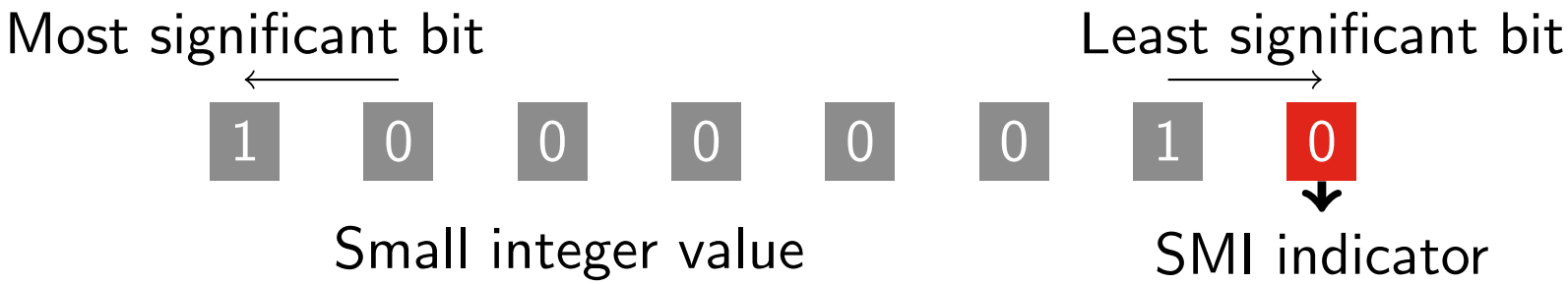
Loads object pool (X27) + 0xF000 (0xF LSL 12) + 0x38 = 0x0F038

Dart's representation of integers: SMI/MINT

```
mov qword [rbp - 0x18], rax
mov r11d, 0x75e ; decimal value = 943
mov qword [rax + 0x17], r11
mov r11d, 0x760 ; 944
mov qword [rax + 0x1f], r11
mov r11d, 0x422 ; 529
```

Dart has 2 different representations of integers:

1. **Small Integers** (SMI). They fit on 31 bits. **Least significant bit set to 0**.
2. **Medium Integers** (Mint). Bigger.



https://cryptax.medium.com/reversing-flutter-apps-darts-small-integers-b922d7fae7d9

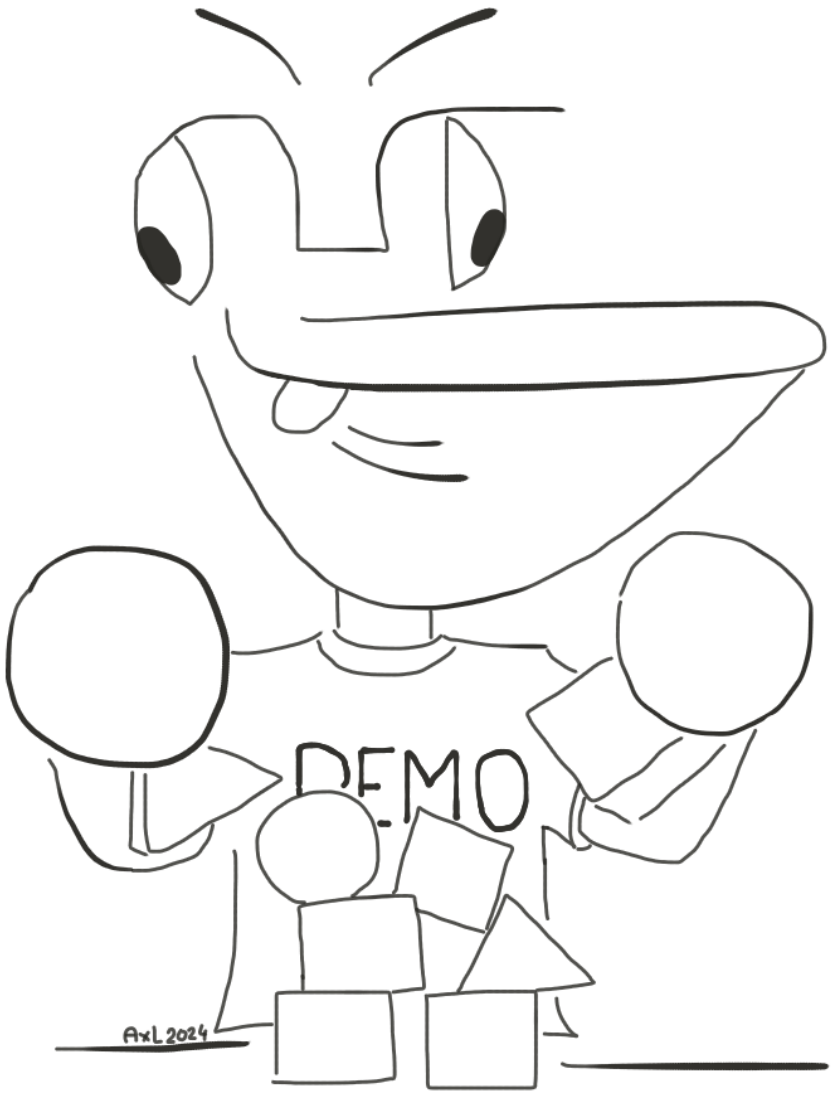
DART.Y CTF challenge

Pico bought a connected fridge at DART.Y. It locks up his favorite caviar from predators, except Pico is hungry and can't remember the password to open his fridge...

```
====== DART.Y - Your Secure & Smart Fridge ======
Password:
```

- Dart AOT snapshot, not stripped
- Flag format is `GH23{.........}`
- The challenge was renamed in GreHack CTF 2023

Demo



When you enter the wrong password

```
Content          | Index
---------------- | ------------
deli             | 943
ph0wn{           | 944
{                | 529
pico             | 945
le               | 946
croco            | 947
GH23{            | 948
caviar           | 949
```

```
champagne        | 950
drink            | 951
chocolate        | 952
yacht            | 953
_                | 555
@                | 231
++               | 954
+                | 535
...
The door is locked
====================
```

It's an Object Pool!

Disassembling the AOT snapshot

Fortunately, it's *not* stripped

```
[0x00058000]> afl~main
0x000afb90    6    201 main
0x000b297c    3     33 sym.main_1
```

We don't have the entire Object Pool but we can guess some

```
mov r11, qword [r15 + 0x1d47] <-- guess: ====== DART.Y ...
mov qword [rsp], r11
call sym.printToConsole
call sym.stdout
mov qword [var_8h], rax
mov r11, qword [r15 + 0x1d4f] <-- guess: Password:
mov qword [rsp], r11
call sym._StdSink.write
call sym.stdin                <-- wait for user input
```

Create Flag

```
mov qword [rsp], rax
call sym.Stdin.readLineSync
mov qword [var_bp_8h], rax
call sym.createFlag          <-- Oooooooooooh! createFlag
```

In createFlag

```
[0x000afb90]> s sym.createFlag
[0x000afc5c]> pif
...
mov r11, qword [r15 + 0x1d6f] <-- Guess: Content | Index
mov qword [rsp], r11
call sym.printToConsole
mov r11, qword [r15 + 0x1d77] <-- Guess: ----- | -----
mov qword [rsp], r11
call sym.printToConsole
```

Many objects are loaded from the Object Pool

```
call sym.stub__iso_stub_AllocateArrayStub
mov qword [var_8h], rax
mov r11, qword [r15 + 0x1d7f]
mov qword [rax + 0x17], r11
mov r11, qword [r15 + 0x1d87]
mov qword [rax + 0x1f], r11
mov r11, qword [r15 + 0x108f]
mov qword [rax + 0x27], r11
mov r11, qword [r15 + 0x1d8f]
mov qword [rax + 0x2f], r11
mov r11, qword [r15 + 0x1d97]
mov qword [rax + 0x37], r11
mov r11, qword [r15 + 0x1d9f]
mov qword [rax + 0x3f], r11
mov r11, qword [r15 + 0x1da7]
mov qword [rax + 0x47], r11
mov r11, qword [r15 + 0x1daf]
mov qword [rax + 0x4f], r11
mov r11, qword [r15 + 0x1db7]
mov qword [rax + 0x57], r11
...
```

Index	Value
0x1d7f // 8	deli
0x1d87 // 8	ph0wn{
0x108f // 8	{

$$0x1d7f//8 = 943$$
$$0x1d87//8 = 944$$

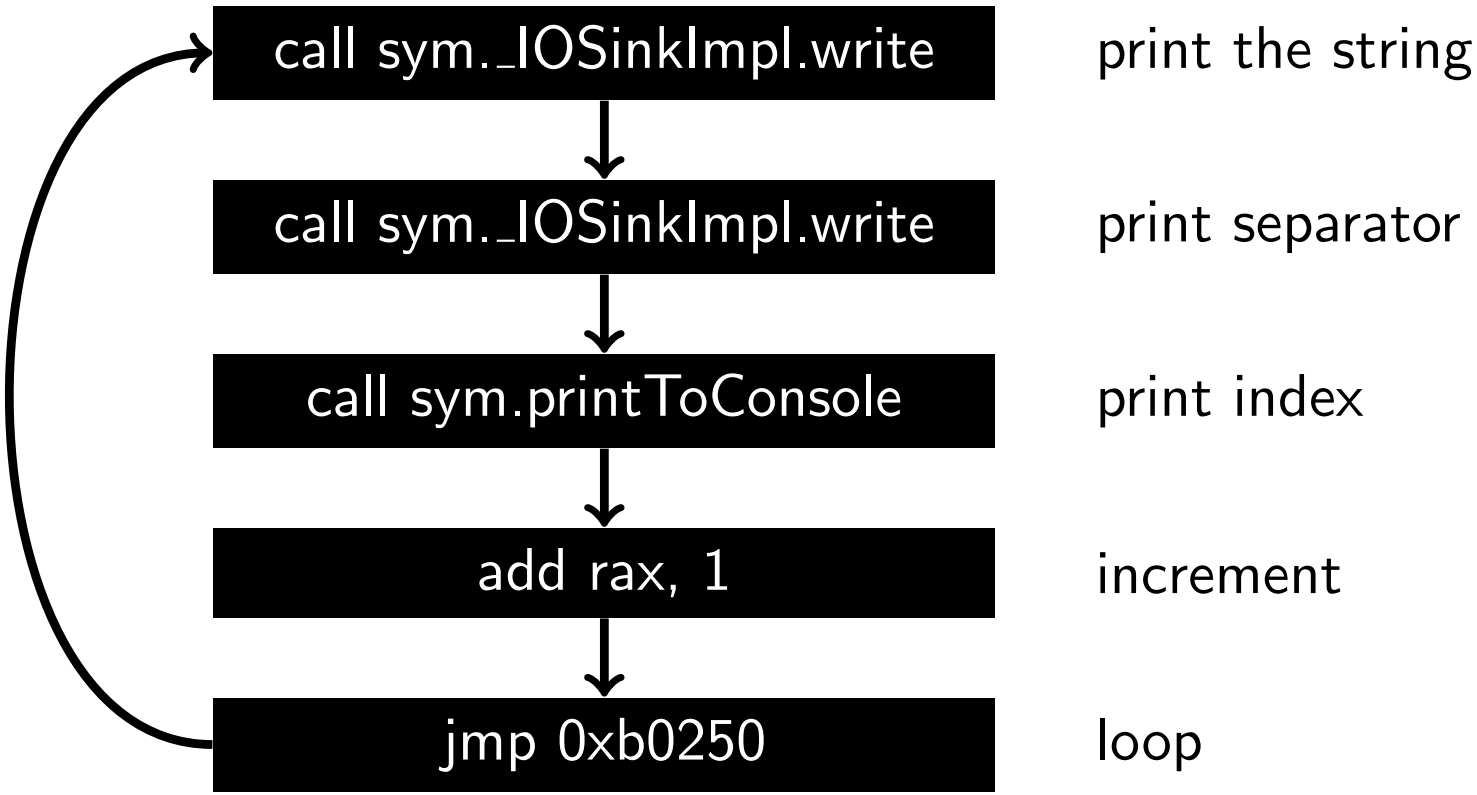
Content	Index
deli	943
ph0wn{	944

Those are the indexes of the supplied Object Pool

```
mov r10d, 0x70
call sym.stub__iso_stub_AllocateArrayStub
mov qword [rbp - 0x18], rax
mov r11d, 0x75e
mov qword [rax + 0x17], r11
mov r11d, 0x760
mov qword [rax + 0x1f], r11
mov r11d, 0x422
```

Index	SMI	Value
$0x1d7f \;//\; 8 = 943_{10}$	$943_{10} << 1 = 0x75e$	deli
$0x1d87 \;//\; 8 = 944_{10}$	$944_{10} << 1 = 0x760$	ph0wn{
$0x108f \;//\; 8 = 529_{10}$	$529_{10} << 1 = 0x422$	{

Loop to print the Object Pool

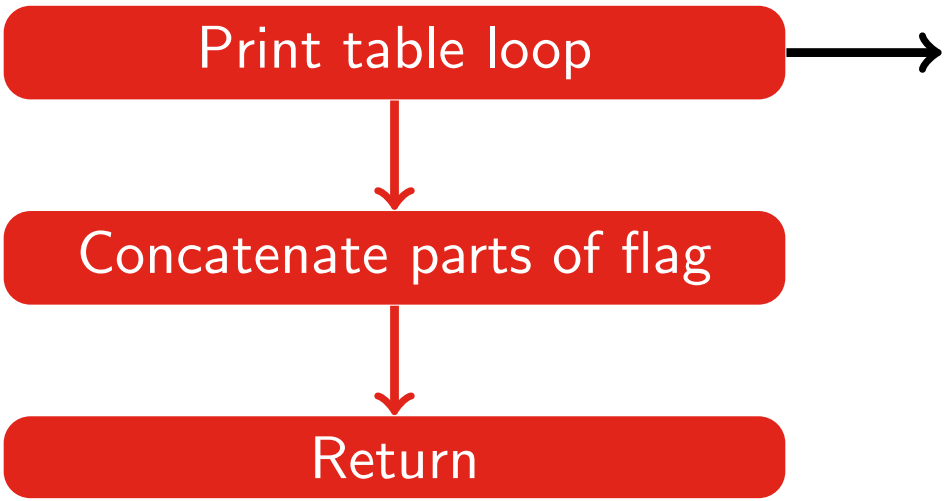


Final part of createFlag

```
mov r11, qword [r15 + 0x115f ]
mov qword [rsp], r11
call fcn.0007880c
mov qword [var_sp_8h], rax
mov r11, qword [r15 + 0x1e5f ]
mov qword [rsp], r11
call fcn.0007880c
mov qword [var_sp_8h], rax
mov r11, qword [r15 + 0x1e77 ]
...
mov rsp, rbp
pop rbp
ret
```

- Access to many objects of the Object Pool
- `fcn.0007880c` is string concatenation
- The result is returned by `createFlag`, so it's the flag!

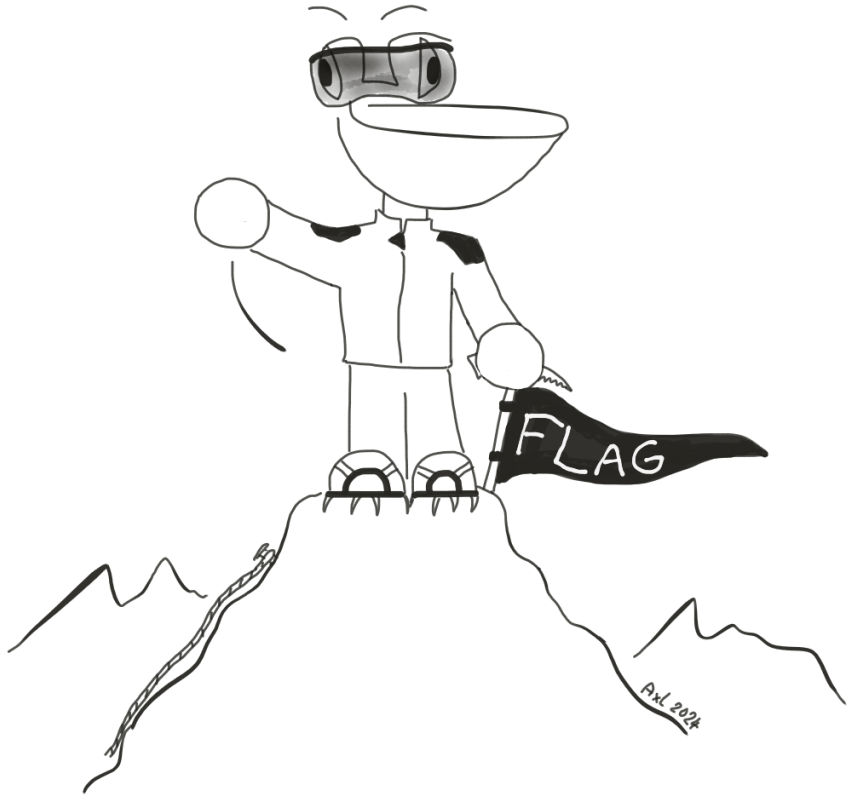
createFlag summary



Content	Index
deli	943
ph0wn{	944
{	529
pico	945
le	946
croco	947
GH23{	948
caviar	949

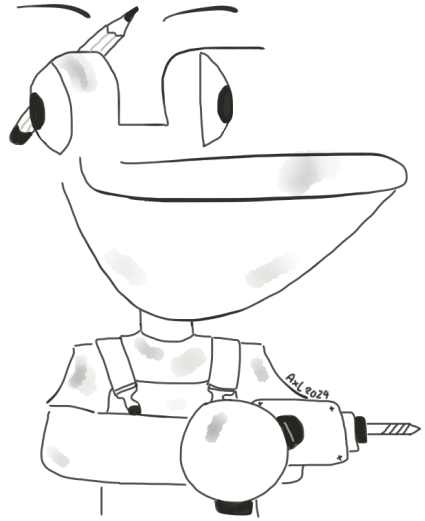
Recover the flag

0x1da7	GH23{
0x115f	_
0x1e5f	s
0x1e77	lurp
0x115f	_
0x1e9f	it
0x115f	_
0x1e5f	s
0x115f	_
0x1d7f	deli
0x1ea7	cious



GH23{_slurp_it_s_delicious_with_some_lobster!}

Next Goal



1. Understand how to reverse **Flutter** applications for Android, especially malware. **Release** applications $\rightarrow$ **Dart AOT snapshot**.
2. ~~Solve **GreHack CTF 2023 Dart challenge** $\rightarrow$ It's a **Dart AOT snapshot**~~. DONE

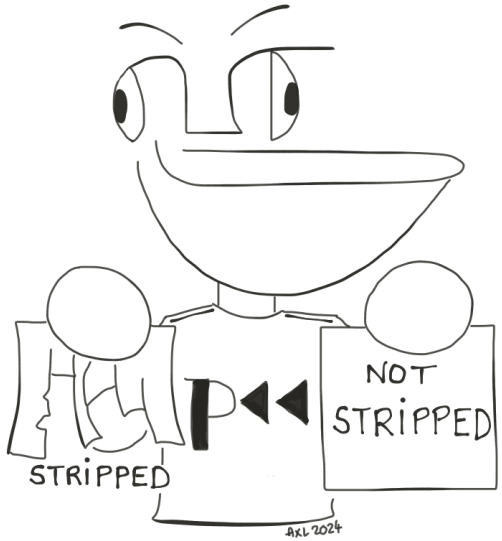
Let's focus on Flutter applications for Android

Flutter application on Android: locating the payload

```
.
├── AndroidManifest.xml
├── assets
│   └── flutter_assets
│       ├── AssetManifest.json
│       ├── assets
│       ├── FontManifest.json
│       ├── fonts
│       ├── NOTICES.Z
│       ├── packages
│       └── shaders
├── classes.dex
├── lib
│   ├── arm64-v8a
│   │   ├── libapp.so
│   │   └── libflutter.so
│   ├── armeabi-v7a
│   │   ├── libapp.so
│   │   └── libflutter.so
│   └── x86_64
│       ├── libapp.so
│       └── libflutter.so
└── resources.arsc
```

- `classes.dex`: contains Java code, and Dalvik to Flutter glue
- `./lib/xxx/libflutter.so`: Flutter implementation
- `./lib/xxx/libapp.so`: payload!

Reversing Flutter applications: what's different?



Releases are stripped

`dart compile aot-snapshot -S ./debuginfo file.dart`

```
[0x00170000]> afl
0x003f7b80    9    212 fcn.003f7b80 <-- no function names
0x003db6c0   28    464 fcn.003db6c0
0x0036f024   14    232 fcn.0036f024
0x00332678   14    232 fcn.00332678
```

Example: Android/SpyLoan (2023)

- Attract victims for easy loans
- Complete a loan application, enter personal information.
- Malware blackmails victims to repay more quickly.

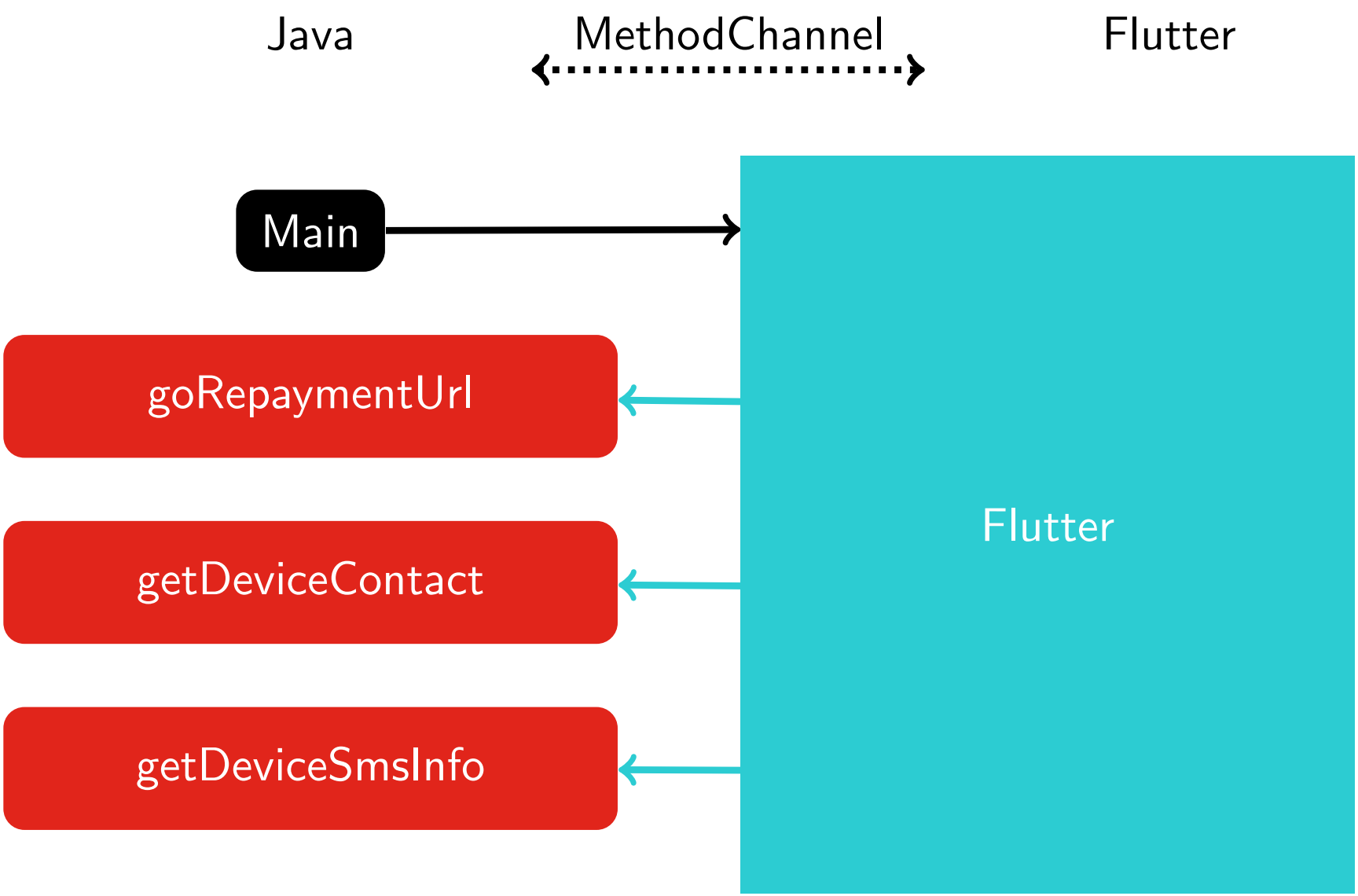
Example: Android/SpyLoan (2023)

- Attract victims for easy loans
- Complete a loan application, enter personal information.
- Malware blackmails victims to repay more quickly.

Source:

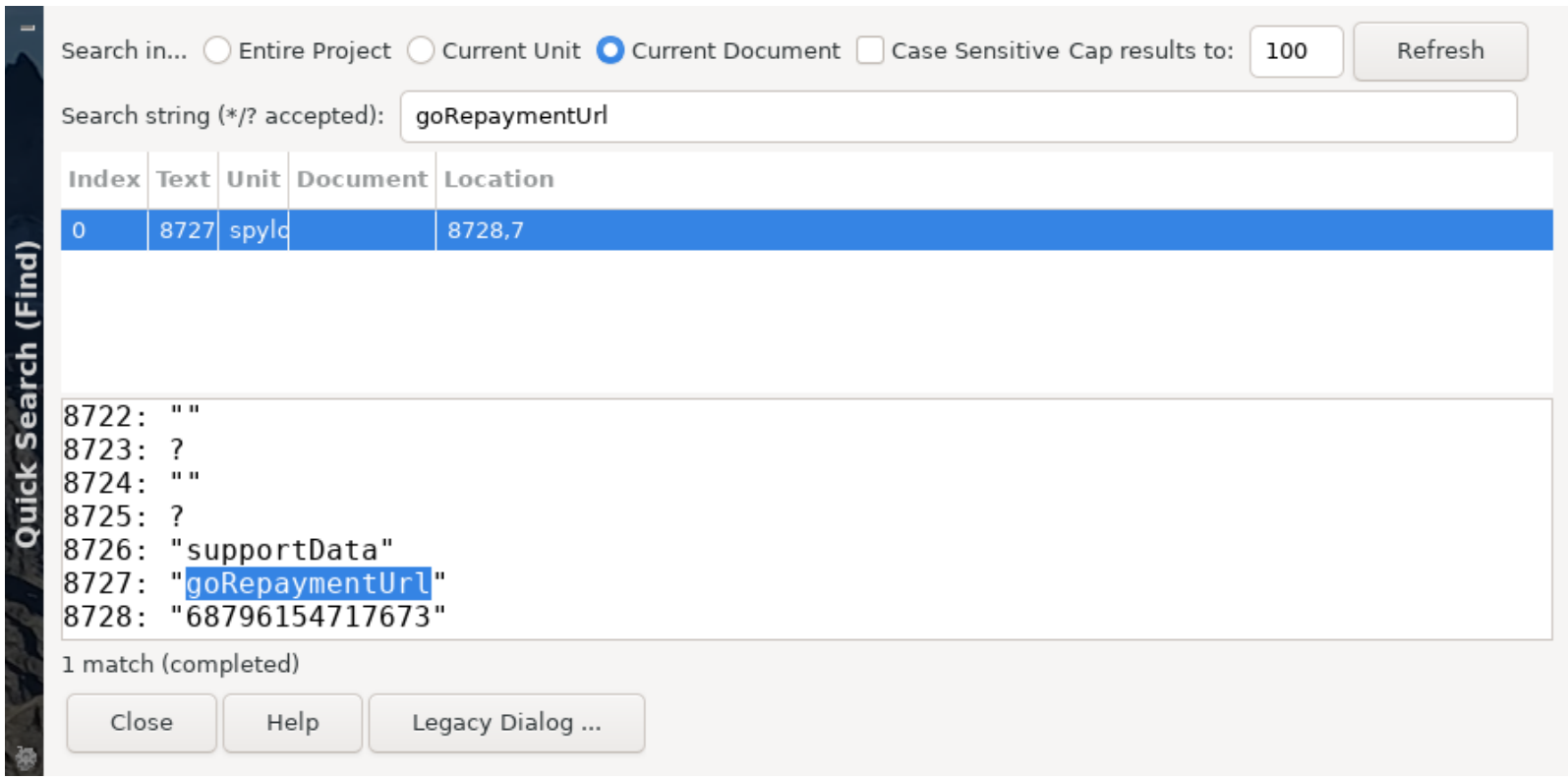
https://www.dailyradar.in/aa-kredit-loan-app-review/

Android/SpyLoan: implementation



Where is goRepaymentUrl called?

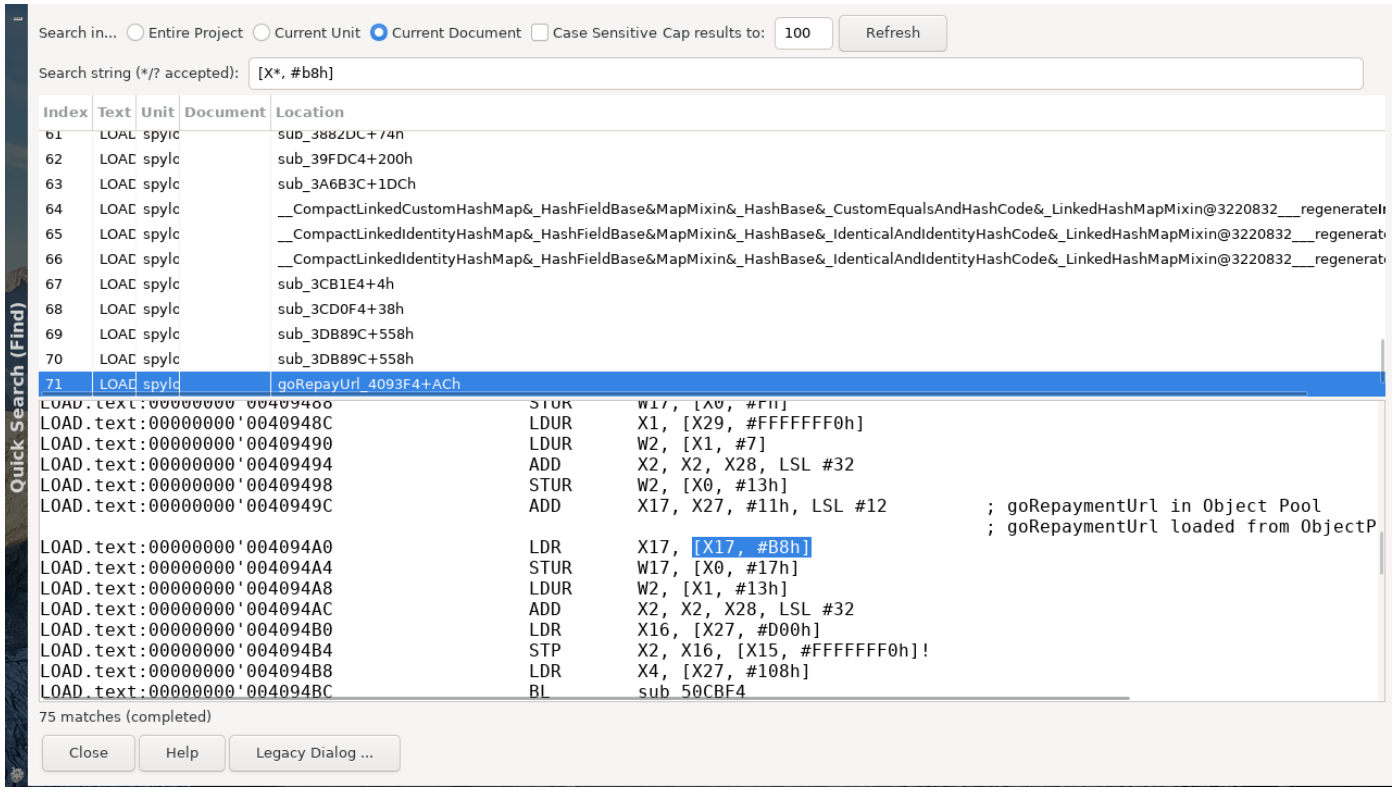
1. ‘‘`goRepaymentUrl`’’ is provided to `MethodChannel`
2. Search ‘‘`goRepaymentUrl`’’ in the Object Pool
3. Index is **8727=0x110b8**



Where is goRepaymentUrl called?

1. ‘‘`goRepaymentUrl`’’ is provided to `MethodChannel`
2. Search ‘‘`goRepaymentUrl`’’ in the Object Pool
3. Index is **8727=0x110b8**
4. Search assembly loading the index:

```
ADD REGISTER, X27, #11h, LSL #12
LDR REGISTER, [REGISTER, #B8h]
```

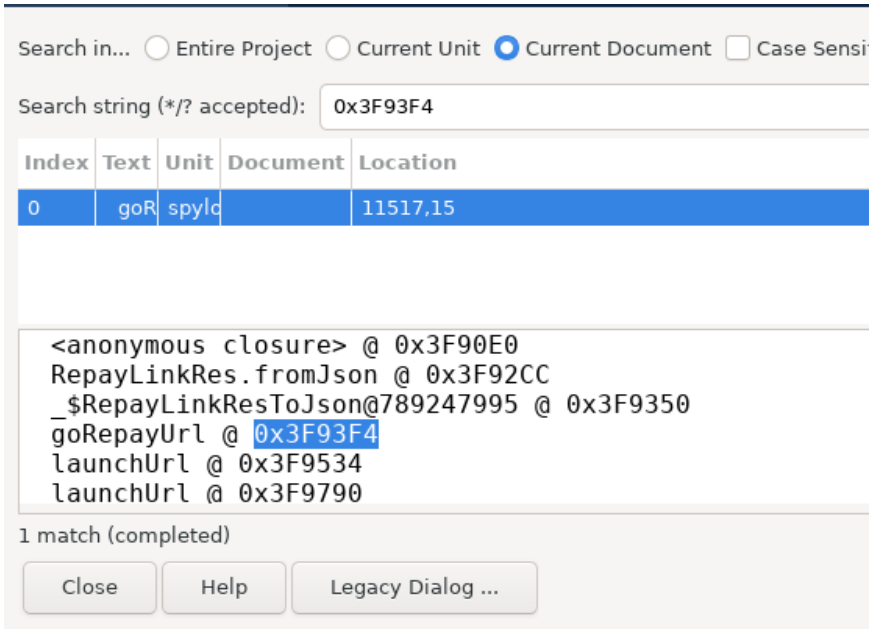


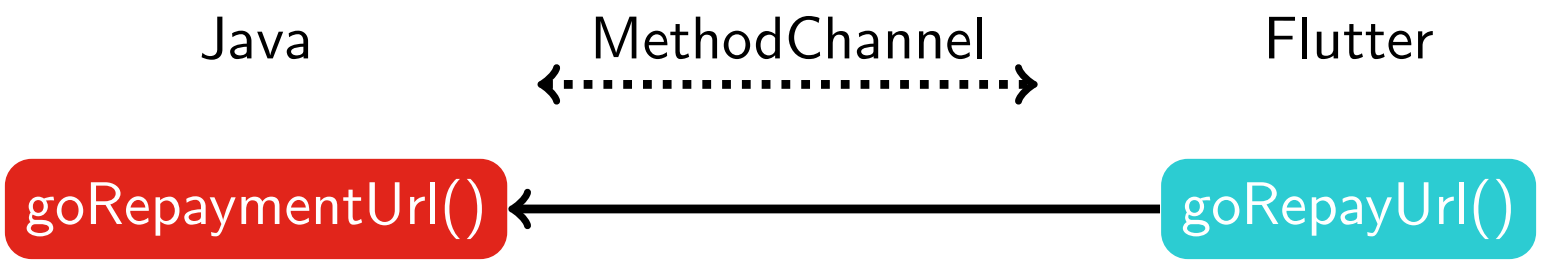
What is the name of this Flutter function?

```
sub_4093F4          proc
...
STUR        X0, [X29, #FFFFFFE8h]
ADD         X17, X27, #Bh, LSL #12
LDR         X17, [X17, #C88h]
STUR        W17, [X0, #Fh]
LDUR        X1, [X29, #FFFFFFF0h]
LDUR        W2, [X1, #7]
ADD         X2, X2, X28, LSL #32
STUR        W2, [X0, #13h]
ADD         X17, X27, #11h, LSL #12 ; goRepaymentUrl loaded from ObjectPool
LDR         X17, [X17, #B8h]
```

- JEB relocation base for zero based relocatable objects: **0x10000** (Options / General / Back-end properties: root, parsers, native, disas)
- The function is at 0x4093F4-0x10000=0x3F93F4
- Search Code Information for **0x3F93F4**

Function name found





Flutter apps for Android AArch64: status

With JEB

- Read the Object Pool: **Yes** (strings only)
- Find function names: **Yes** via Code Information.
- Find string cross references: **Yes** via Search.

Are we lost *without* JEB?

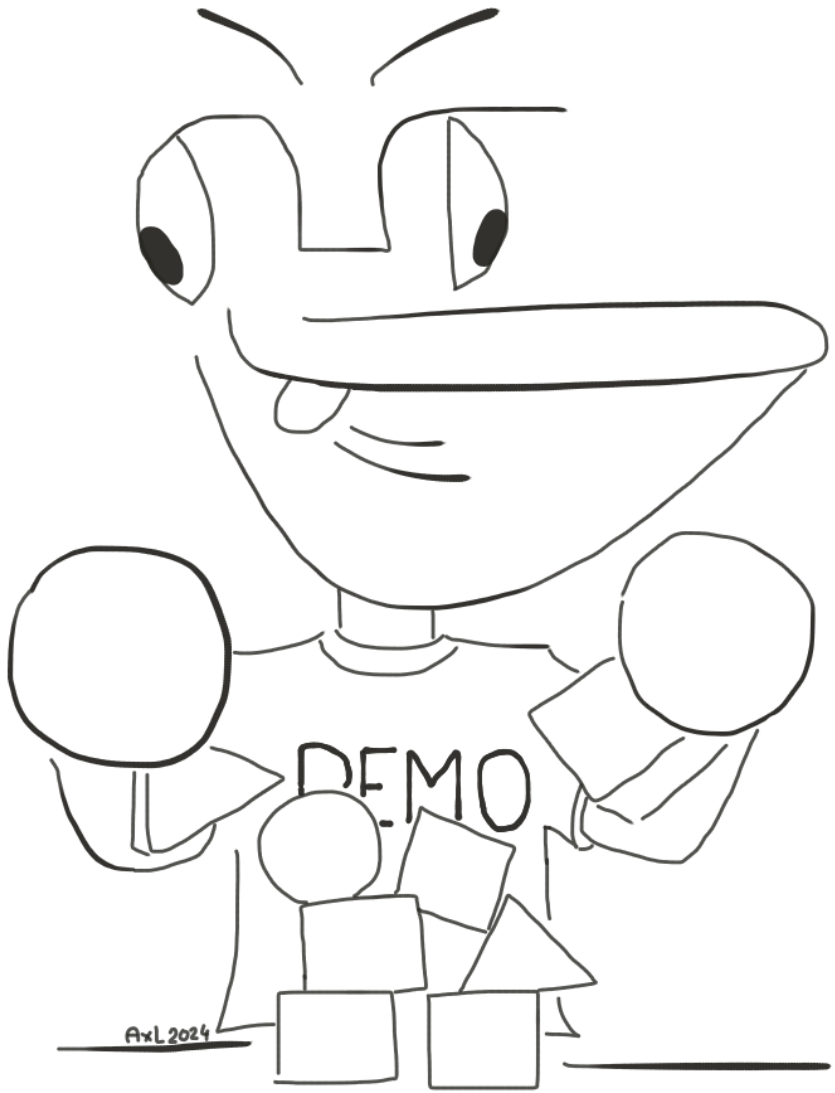
There is still hope

- Blutter: https://github.com/worawit/blutter
- Only works for **recent Android Flutter AAarch64**
- Requires **GCC 13**
- `python3 blutter.py ./malware/spyloan/arm64-v8a outputdir`

Output

- `pp.txt`: all Dart objects in the Object Pool
- `asm/`: assembly code

Demo



Finding goRepaymentUrl with Blutter

grep -C 3 goRepaymentUrl pp.txt

```
[pp+0x110a8] String: ""
[pp+0x110b0] List(7) [0, 0x2, 0x2, 0x1, "mode", 0x1, Null]
[pp+0x110b8] String: "supportData"
[pp+0x110c0] String: "goRepaymentUrl"
[pp+0x110c8] String: "687961547176 73"
[pp+0x110d0] Null
[pp+0x110d8] String: " in type cast"
```

- Blutter finds offset **0x110c0**
- In reality, assembly loads **0x110b8** (0x110c0-8)

Finding function name with Blutter

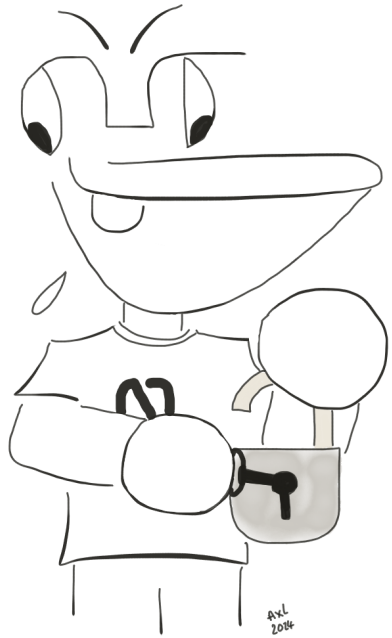
Search for function at **0x3F93F4**

```
$ grep -ri 3F93F4 ./asm/
./asm/flutter_project/plugin/Plugin.dart:    // ** addr: 0x3f93f4
   ...
```

./asm/flutter_project/plugin/Plugin.dart

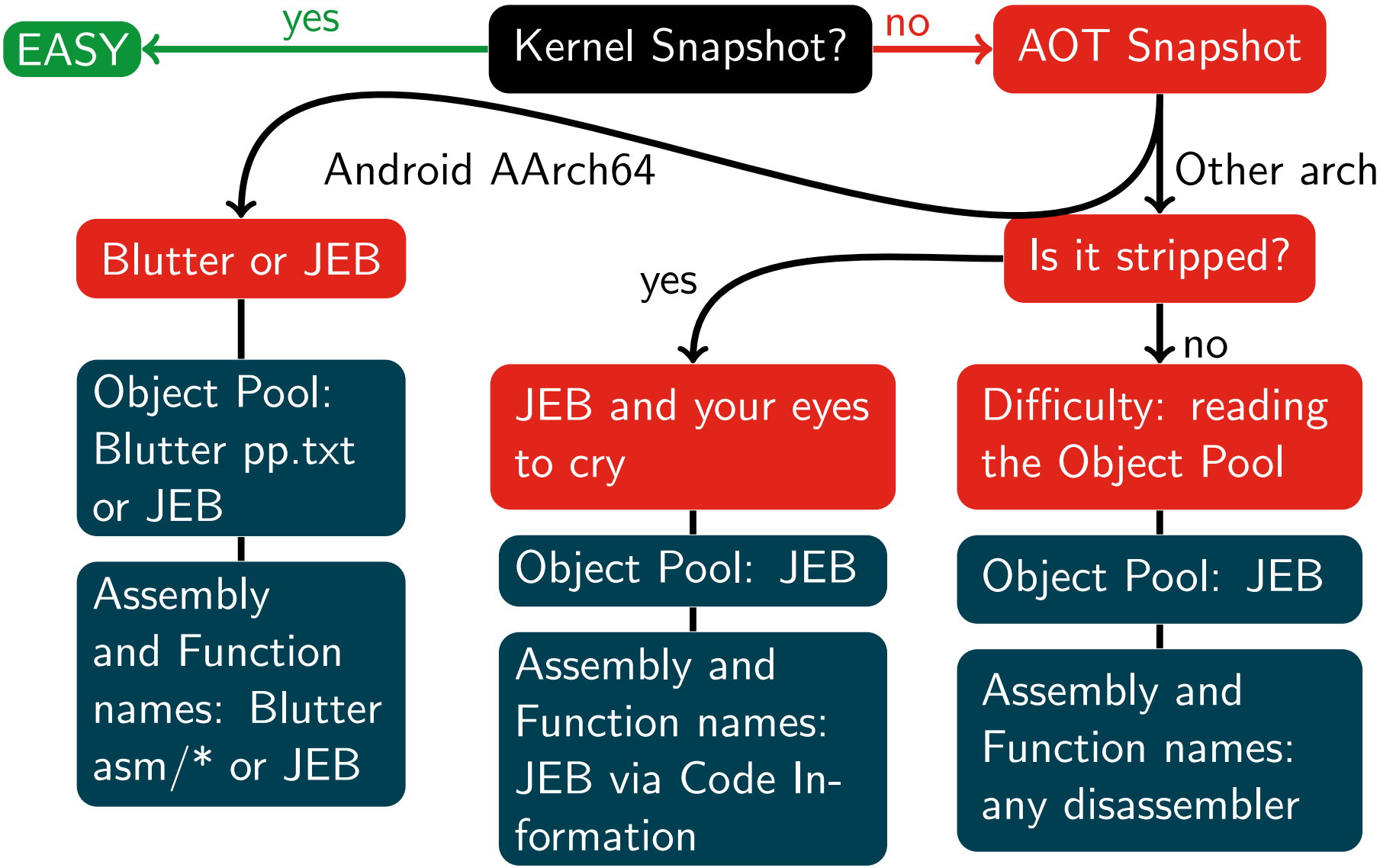
```
static _ goRepayUrl(/* No info */) async {
  // ** addr: 0x3f93f4, size: 0x140
  //  0x3f93f4: EnterFrame
  //      0x3f93f4: stp             fp, lr, [SP, #-0x10]!
  //      0x3f93f8: mov             fp, SP
  //  0x3f93fc: AllocStack(0x18)
  //      0x3f93fc: sub             SP, SP, #0x18
  //  0x3f9400: SetupParameters (dynamic _ /* r1, fp-0x10 */)
```

Goals unlocked

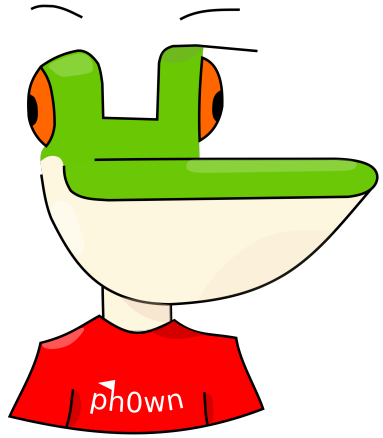


1. ~~Understand how to reverse **Flutter** applications for Android, especially malware. **Release** applications → **Dart AOT snapshot**.~~ DONE
2. ~~Solve **GreHack CTF 2023 Dart challenge** → It's a **Dart AOT snapshot**.~~ DONE

Reversing Dart / Flutter



Thanks for your attention!



- https://github.com/cryptax/talks
- @cryptax (X, Mastodon.social)
- https://ph0wn.org CTF - November 29-30, 2024