B(l)utter
Reversing Flutter Application
by using Dart Runtime

Worawit Wangwarunyoo
Security Researcher, Datafarm
Agenda

- What is Flutter Application?
- Reverse Engineering Challenges
- Building Dart (AOT) Runtime for Reversing
- Getting Information from Dart Snapshot
- Intro to Dart Internal (ARM64)
- Dumping Objects at Runtime with Frida
whoami

- Worawit Wangwarunyoo (@sleepya_)
- Security Researcher
- Working for Datafarm Co., Ltd.
- Public exploits and tools on
  - https://github.com/worawit
What is Flutter Application?
What is Flutter?

- Flutter is an open source framework by Google
  - For building beautiful, natively compiled, multi-platform applications from a single codebase
- Flutter code is powered by Dart platform
- Flutter app developers write code in Dart Language

```dart
void main() {
  if (Random().nextInt(1) == 1) print(devConfig.url);
  final config = kReleaseMode ? prodConfig : devConfig;
  print(URLController._endpoint);
  print(URLController.login);
  print(config.url);
}
```
## Flutter Architectural Overview

<table>
<thead>
<tr>
<th>Framework</th>
<th>Material</th>
<th>Cupertino</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dart</td>
<td>Widgets</td>
<td>Rendering</td>
</tr>
<tr>
<td></td>
<td>Animation</td>
<td>Painting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gestures</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Foundation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Engine</th>
<th>Service Protocol</th>
<th>Composition</th>
<th>Platform Channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>C/C++</td>
<td>Dart Isolate Setup</td>
<td>Rendering</td>
<td>System Events</td>
</tr>
<tr>
<td></td>
<td>Dart VM Management</td>
<td>Frame Scheduling</td>
<td>Asset Resolution</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Frame Pipelining</td>
<td>Text Layout</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Embedder</th>
<th>Render Surface Setup</th>
<th>Native Plugins</th>
<th>Packaging</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform Specific</td>
<td>Thread Setup</td>
<td>Event Loop Interop</td>
<td></td>
</tr>
</tbody>
</table>

Ref: https://raw.githubusercontent.com/flutter/engine/main/docs/flutter_overview.svg
Scope of This Talk

- Target only Mobile Application (Android, iOS)
- ARM64 architecture only
- Release build only
  - Symbols are stripped
  - Full optimization
Flutter Mobile App in Installer Package

- **Dart**
  - **App**
  - **Framework**
  - **Engine**
  - **Embedder**

  - Android: libapp.so
  - IOS: App.Framework

- **C/C++**
  - Android: libflutter.so
  - IOS: Flutter.Framework

![Code Signature](image)
Flutter Mobile Application

- Use Dart Ahead-Of-Time (AOT) compiler
  - for producing machine code (in release build)
- The AOT-compiled code still runs inside a Dart VM
  - Precompiled runtime (a stripped version of Dart VM)
- The code and data are serialized into a binary snapshot
Dart Snapshot

- Serialized state of the Dart VM at a specific point in its execution
- A Dart snapshot is taken just before calling main

```sh
echo $ objdump -T libapp.so
libapp.so: file format elf64-little

DYNAMIC SYMBOL TABLE:

0000000000000080 g DO .text 000000000000005ce6 _kDartVmSnapshotInstructions
0000000000000080 d6 g DO .text 00000000002e6506 _kDartIsolateSnapshotInstructions
0000000000000020 g DO .rodata 000000000000003b39 _kDartVmSnapshotData
00000000000000d4 g DO .rodata 000000000000017b720 _kDartIsolateSnapshotData
00000000000000c8 g DO .note.gnu.build-id 0000000000000020 _kDartSnapshotBuildId
```

```sh
echo $ llvm-objdump -15 --macho -t App
App:

SYMBOL TABLE:

00000000000000de740 g 0 __TEXT,__const__kDartIsolateSnapshotData
00000000000000d5e0 g F __TEXT,__text__kDartIsolateSnapshotInstructions
00000000000000d620 g 0 __TEXT,__const__kDartVmSnapshotData
000000000000007a80 g F __TEXT,__text__kDartVmSnapshotInstructions
```
Problems
Problem 1: Parsing Dart Snapshots

- Universal parser for Dart snapshots cannot be written
  - Dart is constantly evolving
  - The format of snapshots keeps changing

- The snapshot deserialization code is in Dart VM
  - Always included in an installer package
  - Check if the deserializing snapshot is a same version (from hash)

- Known public Dart snapshot parsers
  - Doldrums - https://github.com/rscloura/Doldrums
  - darter - https://github.com/mildsunrise/darter

- Both tools do not work with new Dart versions
  - Updating parser take times
Problem 2: Analyzing Dart (ABI) Code

- Use general purpose registers as special purpose
  - ARM64 R15 -> Dart VM stack pointer
  - ARM64 R27 -> Object pool pointer
  - ...
- Use pool pointer to access object pool
  - No direct references to static data
- Custom calling convention
- Utilize Dart VM functions by calling Dart Stubs
  - Dart stubs are entry points for entering Dart VM from compiled code
- Inline Dart Stubs
// Register aliases.
const Register TMP = R16; // Used as scratch register by assembler.
const Register TMP2 = R17;
const Register PP = R27; // Caches object pool pointer in generated code.
const Register DISPATCH_TABLE_REG = R21; // Dispatch table register.
const Register CODE_REG = R24;
// Set when calling Dart functions in JIT mode, used by LazyCompileStub.
const Register FUNCTION_REG = R0;
const Register FPREG = FP; // Frame pointer register.
const Register SPREG = R15; // Stack pointer register.
const Register IC_DATA_REG = R5; // ICData/MegamorphicCache register.
const Register ARGS_DESC_REG = R4; // Arguments descriptor register.
const Register THR = R26; // Caches current thread in generated code.
const Register CALLEE_SAVED_TEMP = R19;
const Register CALLEE_SAVED_TEMP2 = R20;
const Register HEAP_BITS = R28; // write_barrier_mask << 32 | heap_base >> 32
const Register NULL_REG = R22; // Caches NullObject() value.
#define DART_ASSEMBLER_HAS_NULL_REG 1

// ABI for catch-clause entry point.
const Register kExceptionObjectReg = R0;
const Register kStackTraceObjectReg = R1;
Previous Public Works

- **reFlutter**
  - [https://github.com/Impact-I/reFlutter](https://github.com/Impact-I/reFlutter)
  - [https://swarm.ptsecurity.com/fork-bomb-for-flutter/](https://swarm.ptsecurity.com/fork-bomb-for-flutter/)

- **flutter-re-demo**
  - [https://github.com/Guardsquare/flutter-re-demo](https://github.com/Guardsquare/flutter-re-demo)

- **Andre Lipke’s Blog**
  - [https://blog.tst.sh/reverse-engineering-flutter-apps-part-1/](https://blog.tst.sh/reverse-engineering-flutter-apps-part-1/)

- **Introduction to Dart VM**
  - [https://mracle.ph/dartvm/](https://mracle.ph/dartvm/)
reFlutter Approach

- Patch the Dart Runtime source code to dump a snapshot information while launching an application
  - To avoid writing snapshots parser
- Recompile the flutter engine

Dart

- App
- Framework
- Engine
- Embedder

C/C++

Android: libapp.so
IOS: App.Framework

Android: libflutter.so
IOS: Flutter.Framework

Patched
reFlutter Limitation

• Very difficult to develop and debug patched code
  ○ Hinder the further code analysis development
• Recompiling consumes a lot of resources
  ○ Disk, CPU
  ○ Time
• An application must be repackaged and executed
The Idea

- We only want snapshot deserialization functions
- The functions are only in Dart Runtime
- Can we build Dart SDK as library?
Building Dart Runtime
First Attempt: Building Dart SDK (Failed)

- Following the steps in the Dart wiki page
- Building Dart SDK requires Google’s depot tools
- The tools will fetch all dependencies
  - fetch dart
- The final source code size is >10GB
  - Not good if we have to build multiple versions of Dart Runtime
- The built command builds too many binaries
  - Take times and disk space
Minimize Build to Dart Runtime Only

- Focus on files only in runtime/vm directory

```
    c:\blutter\dart-sdk\v3.0.3\runtime\vm>rg -v "^#" vm_sources.gni
```

```
4: 7:vm_sources = [
8:    "allocation.cc",
9:    "allocation.h",
10:   "app_snapshot.cc",
11:   "app_snapshot.h",
12:   "base64.cc",
13:   "base64.h",
14:   "base_isolate.h",
15:   "bit_vector.cc",
16:   "bit_vector.h",
17:   "bitfield.h",
18:   "bitmap.cc",
19:   "bitmap.h",
20:   "boolfield.h",
21:   "bootstrap.h",
22:   "bootstrap_natives.cc",
```
Minimize Build to Dart Runtime Only

- Create our own CMakeLists.txt
- The defined macros from
  - Generated build files of previous failed attempt
- The source and header files from
  - Parsing the Google’s build script
  - Listing all source file in a subdirectory
  - Adding the missing source files manually (after compiling errors)
- The 3rd party library from
  - Linking error message (only ICU)
  - Use a precompiled one
The Build Result

- Dart SDK clone directory with git sparse checkout
  - Size <100MB
- Building time
  - Less than 5 minutes on my laptop
- Dart Runtime as static library on Windows
  - Size ~20MB
- The target OS and architecture can be selected from
  - DART_TARGET_OS_ANDROID, DART_TARGET_OS_MACOS_IOS
  - TARGET_ARCH_ARM64, TARGET_ARCH_X64
- No source code patching
Start fetching and compiling Dart Runtime

Finished compiling Dart Runtime

Dart Runtime Static Library
Getting Information from Dart Snapshot
Using Dart Runtime Internal API

- To access all loaded information in detail
  - Then fill the information into machine code
- Read Dart SDK source code
  - To learn how to use Internal API
- Use only public class methods
  - Their interfaces should not be changed in a new Dart version
Loading Dart Snapshot

```c
char* error = NULL;
Dart_InitializeParams init_params = { 0 };
init_params.version = DART_INITIALIZE_PARAMS_CURRENT_VERSION;
init_params.vm_snapshot_data = vm_snapshot_data;
init_params.vm_snapshot_instructions = vm_snapshot_instructions;
init_params.start_kernel_isolate = false;
// other params are no needed if snapshot is not run
error = Dart_Initialize(&init_params);

Dart_IsolateFlags flags;
Dart_IsolateFlagsInitialize(&flags);
flags.is_system_isolate = false;
flags.snapshot_is_dontneed_safe = true;
flags.null_safety = true;
auto isolate = Dart_CreateIsolateGroup(nullptr, nullptr,
   isolate_snapshot_data, isolate_snapshot_instructions,
   &flags, nullptr, nullptr, &error);
```
Getting Classes

```cpp
auto table = dart::Isolate::Current()->group()->class_table();
auto& library = dart::Library::Handle();
auto& cls = dart::Class::Handle();

// load from class table
for (intptr_t i = 0; i < table->NumCids(); i++) {
    auto clsPtr = table->At(i);
    if (clsPtr == nullptr) return;
    cls = clsPtr;
    library = cls.library();
    // ...
}
```
Getting Stubs

- Dart Runtime helper functions
- The symbol names are not in the Dart Snapshot
  - They are in Dart Runtime source code

```cpp
#define OBJECT_STORE_STUB_CODE_LIST(DO) DO(dispatch_table_null_error_stub, DispatchTableNullError) DO(late_initialization_error_stub_with_fpu_regs_stub, LateInitializationErrorSharedWithFPURegs) DO(late_initialization_error_stub_without_fpu_regs_stub, LateInitializationErrorSharedWithoutFPURegs) DO(null_error_stub_with_fpu_regs_stub, NullErrorSharedWithFPURegs)

#define DO(member, name) ptr = store->member(); 
    code = ptr; 
    ep_addr = code.EntryPoint(); 
    stub = new DartStub(ptr, DartStub::name ## Stub, ep_addr, code.Size(), #name); 
    stubs[ep_addr] = stub;

OBJECT_STORE_STUB_CODE_LIST(DO);
```

From `<dart_v3.0.3>/runtime/vm/object_store.h`
static _globalContext(/* No info */) async {
    // ** addr: 0x26eff8, size: 0xf8
    // 0x26eff8: stp x29, x30, [x15, #-0x10]!
    // 0x26effc: mov x29, x15
    // 0x26f000: sub x15, x15, #0x28
    // 0x26f004: stur x22, [x29, #-8]
    // 0x26f008: ld r x16, [x26, #0x38]
    // 0x26f00c: cmp x15, x16
    // 0x26f010: b.ls #0x26f0e8
    // 0x26f014: add x0, x27, #0xe, lsl #12
    // 0x26f018: ldr x0, [x0, #0x530]
    // 0x26f01c: bl #0x19faec ; InitAsyncStub
    // 0x26f020: ldr x0, [x26, #0x68]
    // 0x26f024: ldr x0, [x0, #0x1650]
    // 0x26f028: ldr x16, [x27, #0x28]
    // 0x26f02c: cmp w0, w16
    // 0x26f030: b.ne #0x26f03c
    // 0x26f034: ldr x2, [x27, #0x43e8]
    // 0x26f038: bl #0x44bb44 ; InitLateFinalStaticFieldStub
    // 0x26f03c: stur x0, [x29, #-0x10]
    // 0x26f040: add x16, x27, #0xe, lsl #12
    // 0x26f044: ldr x16, [x16, #0x538]
    // 0x26f048: stp x16, x0, [x15]
    // 0x26f04c: bl #0x26f198 ; [package:flutter/src/services/asset_bundle.dart] PlatformAssetBundle::load
Object Pool (PP)

- Global constant objects
  - Also includes immediates and addresses
- Strings are immutable (constants)

```dart
[pp+0x3a38] Null
[pp+0x3a40] String: " in type cast"
[pp+0x3a48] TypeArguments: <_TraversalSortNode>
[pp+0x3a50] Function: [dart:ui] Image::image (0x40f3b4)
[pp+0x3a58] TypeArguments: <_BoxEdge>
[pp+0x3a60] IMM: double(0.1) from 0x3fb999999999999a
[pp+0x3a68] TypeArguments: <_SemanticsSortGroup>
[pp+0x3a70] AnonymousClosure: static (0x40fdec), in [package:flutter/src/semantics
[pp+0x3a78] Type: _BoxEdge
[pp+0x3a80] Null
[pp+0x3a88] String: " in type cast"
[pp+0x3a90] Obj!TextDirection@405381 : {
    Super!_Enum : {
        off_8: int(0x0),
        off_10: "rtl"
    }
}
```
static _globalContext(/* No info */) async {
  // ** addr: 0x26eff8, size: 0xf8
  // 0x26eff8: stp x29, x30, [x15, #-0x10]!
  // 0x26effc: mov x29, x15
  // 0x26f000: sub x15, x15, #0x28
  // 0x26f004: stur x22, [x29, #-8]
  // 0x26f008: ldr x16, [x26, #0x38]
  // 0x26f00c: cmp x15, x16
  // 0x26f010: b.ls #0x26f0e8
  // 0x26f014: add x0, x27, #0xe, lsl #12 ; [pp+0xe530] TypeArguments: <SecurityContext>
  // 0x26f018: ldr x0, [x0, #0x530]
  // 0x26f01c: bl #0x19faec ; InitAsyncStub
  // 0x26f020: ldr x0, [x26, #0x68]
  // 0x26f024: ldr x0, [x0, #0x1650]
  // 0x26f028: ldr x16, [x27, #0x28] ; [pp+0x28] Sentinel
  // 0x26f02c: cmp w0, w16
  // 0x26f030: b.ne #0x26f03c
  // 0x26f034: ldr x2, [x27, #0x43e8] ; [pp+0x43e8] Field <:::rootBundle>: static late final (offset: 0x
  // 0x26f038: bl #0x44bb44 ; InitLateFinalStaticFieldStub
  // 0x26f03c: stur x0, [x29, #-0x10]
  // 0x26f040: add x16, x27, #0xe, lsl #12 ; [pp+0xe538] "assets/certs/server.pem"
  // 0x26f044: ldr x16, [x16, #0x538]
  // 0x26f048: stp x16, x0, [x15]
  // 0x26f04c: bl #0x26f198 ; [package:flutter/src/services/asset_bundle.dart] PlatformAssetBundle::load
Dart Thread Offsets

- List of VM-global objects/addresses cached in each Dart Thread object
- Many objects are accessed through Dart Thread object
- The names are not in the Dart Snapshot

```c
#define DEFINE_OFFSET_INIT(type_name, member_name, expr, default_init_value) \ threadOffsetNames[dart::Thread::member_name##offset()] = #member_name;
CACHED_CONSTANTS_LIST(DEFINE_OFFSET_INIT);
#undef DEFINE_OFFSET_INIT

#define DEFINE_OFFSET_INIT(name) \ threadOffsetNames[dart::Thread::name##_entry_point_offset()] = #name;
RUNTIME_ENTRY_LIST(DEFINE_OFFSET_INIT);
#undef DEFINE_OFFSET_INIT
```
Before

sub_26EFF8

STP  X29, X30, [X15, #0x10]!
MOV  X29, X15
SUB  X15, X15, #0x28 ; '('
STUR X22, [X29, #-8]
LDR  X16, [X26, #0x38]
CMP  X15, X16
B.LS loc_26F0E8

loc_26F014

ADD  X0, X27, #0xe, LSL#12
LDR  X0, [X0, #0x530]
BL  sub_19FAEC
LDR  X0, [X26, #0x68]
LDR  X0, [X0, #0x1650]
LDR  X16, [X27, #0x28]
CMP  W0, W16
B.NE loc_26F03C
LDR  X2, [X27, #0x43E8]
BL  sub_44BB44

After

static __globalContext(/* No info */) async {
    // ** addr: 0x26eff8, size: 0xf8
    // 0x26eff8: stp x29, x30, [x15, #0x10]!
    // 0x26effc: mov x29, x15
    // 0x26f000: sub x15, x15, #0x28
    // 0x26f004: stur x22, [x29, #-8]
    // 0x26f008: ldr x16, [x26, #0x38] ; THR::stack_limit
    // 0x26f00c: cmp x15, x16
    // 0x26f010: b.l.s #0x26f0e8
    // 0x26f014: add x0, x27, #0xe, lsl #12 ; [pp+0xe530] TypeArguments: <SecurityContext>
    // 0x26f018: ldr x0, [x0, #0x530]
    // 0x26f01c: bl #0x19faec ; InitAsyncStub
    // 0x26f020: ldr x0, [x26, #0x68] ; THR::field_table_values
    // 0x26f024: ldr x0, [x0, #0x1650]
    // 0x26f028: ldr x16, [x27, #0x28] ; [pp+0x28] Sentinel
    // 0x26f02c: cmp w0, w16
    // 0x26f030: b.ne #0x26f03c
    // 0x26f034: ldr x2, [x27, #0x43e8] ; [pp+0x43e8] Field <::rootBundle>: static
    // 0x26f038: bl #0x44bb44 ; InitiateFinalStaticFieldStub
    // 0x26f03c: stur x0, [x29, #0x10]
    // 0x26f040: add x16, x27, #0xe, lsl #12 ; [pp+0xe538] “assets/certs/server.pem”
    // 0x26f044: ldr x16, [x16, #0x538]
    // 0x26f048: stp x16, x0, [x15]
    // 0x26f04c: bl #0x26f198 ; [package:flutter/src/services/asset_bundle.dart] Pla
Intro to Dart Internal (ARM64)
Pointer Compression

- Allocate an aligned 4GB region of address space as heap
  - Only lower 32 bits of object pointer is stored in memory
  - Lower memory usage with smaller pointer
  - Not enabled on iOS because it requires an additional application entitlement

- The decompress pointer instruction always be after the loading object instruction

```
ldur  w1, [x0, #7]
add   x1, x1, x28, lsl #32
```
Dart Object Memory Layout (64 bit)

- Object tags (0x00)
- Compressed Ptr (0x08)
- Native int (0x0c)
- Compressed Ptr (0x14)

Object class: size, class id, hash

Instance variable (object)
Instance variable (int)
Instance variable (object)
Dart Calling Convention

- Use R15 register as Dart VM Stack Pointer
- All call arguments are stored in Stack
- Store arguments in reversed order from a typical one

Example for stack frame of function with 3 arguments
Dart Calling with Named Parameters

- Use R4 register as Arguments Descriptor

```assembly
ldr x4, [x27, #0x13c8] ; [pp+0x13c8]
```

RegExp constructor

```dart
RegExp(
  String source,
  {bool multiLine = false,
   bool caseSensitive = true,
   @Since("2.4") bool unicode = false,
   @Since("2.4") bool dotAll = false}
)
```

<table>
<thead>
<tr>
<th>local var x</th>
<th>saved FP</th>
<th>saved LR</th>
<th>arg 0</th>
<th>arg 1</th>
<th>arg 2</th>
<th>arg 3</th>
<th>arg 4</th>
<th>arg 5</th>
<th>Total arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Number of positional argument

[0, 0x6, 0x6, 0x2, "caseSensitive", 0x3, "dotAll", 0x5, "multiLine", 0x2, "unicode", 0x4, Null]
Calling Dart Stub

- Use selected registers as Stub arguments
- Most of them are defined in `constants_<arch>.h`
- Some of them is fixed in compiler

```c
struct InitStaticFieldABI {
  static const Register kFieldReg = R2;
  static const Register kResultReg = R0;
};

struct AllocateObjectABI {
  static const Register kResultReg = R0;
  static const Register kTypeArgumentsReg = R1;
  static const Register kTagsReg = R2;
};

struct AllocateClosureABI {
  static const Register kResultReg = AllocateObjectABI::kResultReg;
  static const Register kFunctionReg = R1;
  static const Register kContextReg = R2;
  static const Register kScratchReg = R4;
};
```

From <dart_v3.0.3>/runtime/vm/constants_arm64.h
Dump Object with Frida
Frida Hooking

- Auto generating script for accessing Dart objects
  - Target application information such as classes
  - Functions for accessing Dart object in memory
- Current support only dumping an Dart object

```javascript
const ShowNullField = false;
const MaxDepth = 5;

function onLibappLoaded() {
  xxx("remove this line and correct the hook value");
  const fn_addr = 0xdeadbeef;
  Interceptor.attach(libapp.add(fn_addr), {
    onEnter: function () {
      init(this.context);
      let objPtr = getArg(this.context, 0);
      const [tptr, cls, values] = getTaggedObjectValue(objPtr);
      console.log(`\${cls.name}@@\${tptr.toString().slice(2)} =`, JSON.stringify(values, null, 2));
    }
  });
}
```
Disassembled code from libapp.so

Frida script for dumping Client.post1 argument
Static final webKey = Key(Uint8List.fromList([11, 22, 33, 44, 55, 66, 77, 88, 99, 255, 0, 128, 64, 32, 16, 8]));

Spawned 're reflut reflut'. Resuming main thread!
[Android Emulator 5554::re reflut reflut] -> CommentData@72006e3f69 = {
  "off_8!String@72006e3e29": "tnlgr/rdWdgxGiXJA4fGiHku9mRYud1D64y3Fxveh4w",
  "off_c!Key@72006e3f59": {
    "parent=Encrypted": {
      "off_8!Uint8List@72006e3ed9": [
        11,
        22,
        33,
        44,
        55,
        66,
        77,
        88,
        99,
        255,
        0,
        128,
        64,
        32,
        16,
        8
      ]
    }
  }
  "off_10!DateTime@72006e3f89": {
    "off_c": "1692689213403087",
    "off_14!bool@7200080b1": false
  }
}
Conclusion

● Using Dart Runtime is allowed us to get a lot of information from a Flutter application
  ○ All symbol names in a Dart Snapshot
  ○ Names of fixed value/ constants that only used in Runtime code
    ■ Stub names
    ■ Thread offset names
● These information make further analysis easier
  ○ This part requires studying Dart internals
● The Blutter tool will be released at
  ○ https://github.com/worawit/Blutter
DEMO
THANK YOU!