tch Calibratisticer (Messi) Lander (Mession Sedenation tch Calibratisticer (Mession) Lander (Mession Sedenation Calibratisticer (Mession) Lander (Mession Sedenation Calibratisticer (Mession) Lander (Mession) Lander (Mession) (Mession)





Practical Guide to Reverse Engineering Flutter Applications

By: Neel Patel (Security Engineer I @SecurityInnovation) Asif Iqbal Gazi (Security Engineer II @SecurityInnovation)

Abstract

Flutter's popularity as a cross-platform framework has made it a go-to choice for building mobile applications with a single codebase, simplifying development across Android and iOS. However, Flutter's architecture and compilation process introduce unique challenges for reverse engineering, especially compared to native Android and iOS apps.

This whitepaper explores these differences and provides a step-by-step guide to reverse engineering Flutter applications—from understanding the app structure and decompiling code to analyzing Dart artifacts. Using an open-source Flutter app called **Minesweeper**, this guide demonstrates tools and techniques for reverse engineering Flutter Application. Whether you are a security researcher or a curious developer, this guide offers practical insights into the process.



Overview of Reverse Engineering Flutter Apps

Disassembling Flutter Binaries with Ghidra

For iOS applications, the core logic is typically found in the binary located in the located in the <app_name>.app directory. However, in Flutter apps the logic resides in <app_name>.app > Frameworks > App.framework > App.



Figure 1: Screenshot of Minesweeper App Extraction/Unzip

- 1. Extract the binary and load it into Ghidra.
- 2. Observe that, the exported symbols in the symbol tree include only:
 - _kDartIsolateSnapshotData
 - _kDartIsolateSnapshotInstructions
 - _kDartVmSnapshotData
 - _kDartVmSnapshotInstructions



Figure 2: Screenshot of Symbol Tree in Ghidra

These symbols relate to the Dart runtime and its snapshot files, which manage compiled code and data.



Decoding Dart Runtime and Snapshots

Understanding the following components is critical for effective analysis:

Dart VM and Snapshots: Flutter applications are powered by the Dart Virtual Machine (VM), which enables the execution of Dart code, managing everything from memory to runtime operations. In production builds, Dart code is compiled Ahead-Of-Time (AOT) into native code, serialized into snapshot files that the Dart VM loads at runtime. Snapshots are serialized representations of compiled Dart code and data, allowing efficient runtime initialization.

- *VM Snapshots:* These contain shared code and data required by all isolates, like core libraries and commonly used functions. VM snapshots load at app startup, providing foundational instructions and data.
- *Isolate Snapshots:* Each isolate, an independent Dart execution context, has its own isolate snapshot with isolate-specific code and data. This allows multiple isolates to run concurrently without shared memory.

Isolates: In Dart and Flutter, concurrency is managed through isolates rather than traditional threads. Each isolate has its own memory space and runs independently, avoiding the risks of shared memory. The Dart VM manages the lifecycle of isolates, including creating, communicating, and terminating them. This architecture enhances stability but adds complexity to reverse engineering, as each isolate operates independently, making shared application state harder to track.

Dart Pool: The Dart Pool (or **Object Pool**) is an internal component of the Dart VM that manages frequently used objects and data structures. Acting as a cache, the Dart Pool reduces redundant memory allocations and optimizes performance. It manages reusable resources like pointers, precompiled instructions, and certain serialized data elements frequently referenced across the application.

The Dart Pool is closely related to Dart VM snapshots and isolates because it holds reusable resources that isolates can access. When a new isolate is created, it doesn't need to load resources from scratch; instead, it can reference objects in the Dart Pool, improving memory use and isolate startup times. This design is especially beneficial in Flutter, where applications often use multiple isolates (e.g., the main UI isolate and background isolates for async tasks).

With the background information provided, the 5 exported symbols have the following significance:

_kDartIsolateSnapshotData represents the initial state of the Dart heap and includes isolate-specific information.

_kDartIsolateSnapshotInstructions contains the Ahead-of-Time (AOT) code that is executed by the Dart isolate.

_kDartVmSnapshotData represents the initial state of the Dart heap shared between isolates. This helps launch Dart isolates faster but doesn't contain any isolate-specific information.

_kDartVmSnapshotInstructions contains AOT instructions for common routines shared between all Dart isolates in the VM. This snapshot is typically extremely small and mostly contains stubs.

_kDartIsolateSnapshotInstructions contains the application code that is executed by Dart Runtime.

We will not delve further into the technical details of the Dart Virtual Machine (DartVM) or Flutter architecture in this guide. However, we have provided references and external resources at the end of the whitepaper for readers interested in exploring these topics in more depth.



Metadata Extraction

Patching and Preparing the Binary

With information obtained from the previous section, if we want to continue our static analysis journey using Ghidra, we need symbolic information like Function Name, and Class Name. To do that, the app needs to be patched manually by modifying the source code of DartVM, compiling it, and replacing the runtime binary in the app. However, this is a tedious process. We can utilize an open-source framework called reFlutter to automate this process.

Using reFlutter framework:

- 1. Patch the app binary following reFlutter's GitHub instructions.
- 2. Extract the patched application.
- 3. Update the info.plist file to support document browsing.
- 4. Compress the Payload folder, rename it to .ipa, and install the app on a device.
- 5. Retrieve the dumped metadata file after running the app.



Figure 3: Screenshot of Patching App Binary using reFlutter



Figure 4: Screenshot of Patched App Binary release.RE.ipa)



•••			Info.plist								
88	< > 🖽 Info.plist										
⊞ (Info.plist > No Selection										
	Кеу		Туре		Value						
~ Int	formation Property List				(33 items)						
	Bundle name	٥			minesweeper						
	DTSDKName	0	String		iphoneos18.2						
	DTXcode	0	String		1620						
>	CFBundleIcons~ipad	0	Dictionary		(1 item)						
	Launch screen interface file base name	0			LaunchScreen						
	DTSDKBuild	0	String		22C146						
	Default localization	0			en						
	Bundle version	0			0.1.0						
	BuildMachineOSBuild	0	String		24C101						
	DTPlatformName	0	String		iphoneos						
	Bundle OS Type code	0			APPL						
	Bundle version string (short)	0			0.1.0						
>	CFBundleSupportedPlatforms	0	Array		(1 item)						
	Application supports indirect input events	0			YES	0					
	Main storyboard file base name	0			Main						
	InfoDictionary version	0			6.0						
	Executable file	0			Runner						
	DTCompiler	0	String	٥	com.apple.compilers.llvm.clang.1_0						
>	Required device capabilities	0			(1 item)						
	MinimumOSVersion	٥	String	٢	12.0						
	Bundle identifier	0			com.example.minesweeper						
>	UIDeviceFamily	0	Array		(2 items)						
	Bundle creator OS Type code	0			????						
	DTPlatformVersion		String		18.2						
	Bundle display name	0			Minesweeper						
>	Icon files (iOS 5)	0			(1 item)						
	DTXcodeBuild	0	String		16C5032a						
	Application requires iPhone environment	0			YES	0					
	Supports Document Browser	000	Boolean		YES	٥					
>	Supported interface orientations	0	Array	0	(3 items)						
	CADisableMinimumFrameDurationOnPhone	0	Boolean		YES	0					
	DTPlatformBuild	0	String		22C146						
>	Supported interface orientations (iPad)	0			(4 items)						

Figure 5: Screenshot of modified info.plist



Figure 6: Screenshot of metadata dump after running the patched application for a while



Resolving Symbols in Ghidra using Dumped Metadata File

We can now resolve the symbols using the dump.dart that we recovered previously.

- 1. Load the dumped metadata (dump.dart) into Ghidra.
- 2. Use the Ghidra script shared below to resolve function names and populate the symbol tree.

Head over to Window > Script Manager and Create a new script based on Java and paste the following code:

// Java Script to resolve symbols in Flutter App using Dart metadata file. // @Neel Patel
import java.io.BufferedReader; import java.io.FileReader; import java.io.IOException; import java.util.HashSet; import java.util.Set;
import ghidra.app.script.GhidraScript;
import ghidra.app.cmd.disassemble.DisassembleCommand;
import ghidra.app.cmd.function.CreateFunctionCmd;
import ghidra.program.model.address.Address;
import ghidra.program.model.symbol.SourceType;
public class ResolveFunctions extends GhidraScript {
/**
* Creates a function at a given address with the specified name.
 * @param offset The offset where the function is located. * @param functionName The name to assign to the function. */
private void createFunction(long offset, String functionName) { // Convert offset to Address object
Address functionAddress = toAddr(offset);
println("Creating function: " + functionName);
// Disassemble the function at the given address
DisassembleCommand disassembleCmd = new
DisassembleCommand(functionAddress, null, true);
If (disassembleCmd.applyTo(currentProgram)) {
} else {
println("Disassembly failed at address: " + functionAddress);
}
<pre>// Check if a function already exists at the address if (getFunctionAt(functionAddress) != null) { println("A function already exists at address " + functionAddress); return:</pre>
}



```
// Create the function at the specified address
    CreateFunctionCmd createFunctionCmd = new CreateFunctionCmd(functionName,
functionAddress, null, SourceType.USER_DEFINED);
    if (createFunctionCmd.applyTo(currentProgram)) {
      println("Function created successfully at address: " + functionAddress);
    } else {
      println("Failed to create function at address: " + functionAddress);
    }
 }
 @Override
 public void run() {
    // Path to the Dart metadata file
    String metadataFilePath = "/Path/to/dump.dart";
    // Hex value to be added to the offset
    String dartSnapshotInstructionHexOffset = "0x0000e900";
    try (BufferedReader reader = new BufferedReader(new FileReader(metadataFilePath)))
// Set to store unique JSON objects
      Set<String> uniqueJsonObjects = new HashSet<>();
      StringBuilder jsonContent = new StringBuilder();
      String line;
// Read all lines from the file and accumulate them into jsonContent
      while ((line = reader.readLine()) != null) {
jsonContent.append(line);
      }
// Split the JSON content into separate JSON objects
      String[] jsonObjects = jsonContent.toString().split("\\}\\{");
// Print the table header for debugging
      printf("%-20s | %-20s | %-20s | %-20s%n", "Method Name", "Offset", "Library URL",
"Class Name");
      printf("------");
// Iterate over each JSON object
      for (String jsonObject : jsonObjects) {
// Clean up JSON object by removing leading '{' and trailing '}'
jsonObject = jsonObject.replaceFirst("^\\{", "").replaceAll("\\}$", "");
// Only process the JSON object if it's unique
if (uniqueJsonObjects.add(jsonObject)) {
// Split the JSON object into key-value pairs
String[] keyValuePairs = jsonObject.split(",");
String methodName = null, offsetHex = null, libraryUrl = null, className = null;
```



```
// Extract values from key-value pairs
for (String pair : keyValuePairs) {
              String[] keyValue = pair.split(":");
              String key = keyValue[0].replaceAll("\"", "").trim();
              String value = keyValue[1].replaceAll("\"", "").trim();
              // Assign the extracted values to respective variables
              switch (key) {
                 case "method_name":
                   methodName = value;
                   break:
                 case "offset":
                   // Convert offset to hexadecimal and add the snapshot instruction offset
                   int originalOffset = Integer.parseInt(value.substring(2), 16);
                   int updatedOffset = originalOffset +
Integer.parseInt(dartSnapshotInstructionHexOffset.substring(2), 16);
                   offsetHex = "0x" + String.format("%08x", updatedOffset);
                   break;
                 case "library_url":
                   libraryUrl = value;
                   break;
                 case "class_name":
                   className = value;
                   break;
              }
}
// Print details for debugging or processing
// Uncomment the line below if you want to print the details to the console
// printf("%-20s | %-20s | %-20s%n", methodName, offsetHex, libraryUrl,
className);
// Create the function using the parsed offset and method name
if (methodName != null && offsetHex != null) {
              createFunction(Long.parseLong(offsetHex.substring(2), 16), methodName);
}
}
       }
    } catch (IOException e) {
       e.printStackTrace();
    }
 }
}
```



In the above script, replace the filePath with the Path of dump.dart file and also change the offset address of _*kDartIsolateSnapshotInstructions* which can be retrieved as shown below:

Program Tree ×	0000e8fb 91 7? 91h 0000e8fc c0 ?? C0h
🗼 Symbol Tree 🗾 🖻 🕅 🗙	0000e8fd 03 ?? 03h 0000e8fe 5f ?? 5Fh _ 0000e8ff d6 ?? D6h
Exports	kDartIsolateSnapshotInstructions
kDartVmSnapshotData kDartVmSnapshotData	0000e901 32 ?? 32h 2 0000e902 1d ?? 1Dh 0000e903 00 ?? 00h
Mathematical Section 2018 S	0000e904 00 ?? 00h 0000e905 00 ?? 00h 0000e905 00 ?? 00h
_kDartIsolateSnapshotData _kDartIsolateSnapshotInstructions	0000e907 00 77 00h 0000e908 40 77 40h @

Figure 7: Screenshot of retrieving offset address in Ghidra using dump.dart

After making required changes and running the script we can see that Symbol Tree is populated with the original function names:

🔜 Symbol Tree 🗾 🖬 🍡	ti ×	
> <u>-</u>		*
> ि= a ∨ ▷= A		undefined Action()
AbstractClassInstantiationErrorc	reate undefined	w0:1 <return></return>
> 🕈 Action		ACCION
> f AnimationController	Function - AbstractClassInstantiationErrorcreate	
> f AnimationController.unbounded	undefined AbstractClassInstantiationErrorcreate (void)	
> f ArgumentError	undefined w0:1 <return></return>	
> f ArgumentError.value		
> f Array		

Figure 8: Screenshot of populated symbol tree in Ghidra using dump.dart and Ghidra Script

Note: The App Binary until this point can be patched. The changes made to the binary after this point will result in an unpatchable binary. Changes made beyond this point are only to make analysis easier.



Resolving Code and Data References

After the previous steps, observe that there are no references between the code and data in the disassembly, as shown in the figure below. (Searching for string "T I M E" from the source code):



Figure 9: Screenshot of the string "T I M E" in the source code



Figure 10: Screenshot of searching the string "T I M E" in Ghidra



Enhancing Dart Code Analysis

The challenge discussed above arise due to:

• Object Pool Indirection - as discussed in section 22.



Figure 11: Screenshot of Dart Serialization and Deserialization

 Non-standard stack pointer usage: Dart VM uses register X15 as a stack pointer instead of the standard SP

The Dart object pool is a big array with pointers to the Dart objects. During runtime, the Dart object pool contains pointers to the deserialized dart objects like string and integers; these objects are accessed using the X27 register. Register X27 points to the Dart Object Pool, and the deserialized Dart objects are accessed through this pointer. As shown in the disassembly below, the object is loaded into the X1 register.

			FUNCTION		*
		undefined HomeP	ageState()		
	undefined	w0:1	<return></return>		
		HomePageState		XREF[2]:	createState:00130a
					0036dT57(*)
	00130a74 td 79 bt as	9 stp	x29,x30,[x15, #-0x10]!		
	00130a78 TO 05 0T 88	a mov	X29,X15		
	001303/C 01 41 00 01	L SUD	X12,X12,#0x10		
	00130000 C3 C2 00 91		x3,x22,#0x30		
	00130404 22 08 00 02		va #8v0		
	00130300 20 01 00 02	. IIIOV	v4 v1		
	00130a00 a1 83 1f ff	stur	v1.[v29. #-0v8]		
	00130a94 50 1f 40 f	ldr	x16.[x26, #0x38]		
	00130a98 ff 01 10 et	, cmp	x15.x16		
	00130a9c a9 07 00 54	b.ls	LAB 00130590		
		LAB_00130aa0		XREF[1]:	00130b94(j)
	00130aa0 80 70 02 f8	3 stur	x0,[x4, #0x27]		
	00130aa4 83 f0 03 f8	3 stur	x3,[x4, #0x3f]		
	00130aa8 82 f0 01 f8	3 stur	x2,[x4, #0x1f]		
	00130aac e1 03 16 aa	s mov	x1,x22		
	00130ab0 02 00 80 d2	2 mov	x2,#0×0		
	00130ab4 6a 85 fb 97	7 bl	_GrowableList		undefined _Grow
	00130ab8 a3 83 5f f8	B ldur	x3,[x29, #-0x8]		
	00130abc 60 f0 02 f8	3 stur	x0,[x3, #0x21]		
	00130ac0 70 10 51 38	8 ldurb	w16,[x3, #-0x1]		
	00130ac4 11 10 51 38	s ldurb	w17,[x0, #-0x1]		
	00130ac8 30 0a 50 88	and and	x16,x17,x16, LSR #0x2		
	00130acc 1T 82 5C 68	a tst	X10, X20, LSK #0X20		
	00130ad4 4c 4d 02 04	+ 0.eq 1 k1	EIN 001c4004		
	00150804 40 40 62 54		101_00104004		differ the ron_c
		LAB_00130ad8		XREF[1]:	00130ad0(j)
0	00130ad8 61 4f 44 fs	€ ldr	x1,[x27, #DAT_00000898]		
	00130adc 02 03 80 d2	2 mov	x2,#0x18		

Figure 12: Screenshot of Non-Standard Stack pointer in use



Also, Dart VM uses X15 as the Stack register instead of the SP register. During the disassembly, decompilers deduce parameters based on the SP register. This is the reason why, in the decompiled code, there are no local variables detected and functions don't have any parameters.

In ARM64, registers X0-X7 are used for passing function parameters. If more than 8 parameters are used, the subsequent values are stored on the stack. However in Dart, function parameters are passed only on the stack. So reverse engineering tools like Ghidra often incorrectly assume the number of function parameters used when performing decompilation.

Steps to address these challenges:

- Add cross-references to deserialized objects.
- Replace X15 with SP for stack pointer recognition.
- Adjust function signatures to properly account for parameters passed on the stack.

Mapping Dart Objects

Establishing cross-references between the Dart code and the deserialized objects in the heap is a challenging task, as the Dart objects are stored in the runtime heap. To retrieve these objects in Ghidra, the following steps must be performed:

- 1. The Dart object pool and other Dart objects must be dumped from the application heap at runtime.
- 2. This memory dump needs to be imported to Ghidra.
- 3. The data types of the Dart object pool and other Dart objects must be defined in Ghidra.
- 4. References to the X27 register must be replaced with the static memory location of the Dart object pool array in the memory dump.

DartVM Registers

Registers used by Dart VM can be found here: <u>https://github.com/dart-lang/sdk/blob/main/runtime/vm/constants_arm64.h</u>



num Penister	• <i>1</i>
RØ = 0,	
R1 = 1,	
R2 = 2,	
R3 = 3,	
R4 = 4,	
R5 = 5,	
R6 = 6,	
R7 = 7,	
R8 = 8,	
R9 = 9,	
R10 = 10.	
R11 = 11.	
R12 = 12	
R12 = 12,	
P14 = 14	
R14 = 14,	// SD in Dart code
R15 = 15,	// TPA aka TND
R10 = 10,	// IPU dKd IPIP
R17 = 17,	// IPI aka IMPZ
R18 = 18,	// reserved on 105, shadow call stack on Fuchsia, TEB on Windows.
R19 = 19,	
R20 = 20,	
R21 = 21,	// DISPATCH_TABLE_REG (AOT only)
R22 = 22,	// NULL_REG
R23 = 23,	
R24 = 24,	// CODE_REG
R25 = 25,	
R26 = 26,	// THR
R27 = 27,	// PP
R28 = 28,	// HEAP_BITS
R29 = 29,	// FP
R30 = 30,	// LR
R31 = 31,	// ZR, CSP
kNumber0fCp	puRegisters = 32,
kNoRegister	· = -1.
kNoRegister	2 = -2.
y	,
// These re	poisters both use the encoding R31, but to avoid mistakes we give
// them dif	ferent values, and then translate before encoding.
CSP = 32	refere valaes, and elen eranstate before encoaring.
70 - 32	
21 - 55,	
// Aliacoc	
TPO - D16	
190 = R10,	
IPI = RI/,	
SP = R15.	

Figure 13: Screenshot of Dart-SDK source code Non-Standard Stack pointer



Dumping Flutter Heap to Retrieve Object Pool

To get deserialized objects which are present in the memory heap, heap memory must be dumped and then it can be imported in Ghidra for further analysis. Frida will be used in this tutorial for the same purpose. First of all, we need to get information about object pool and heap start address which can be retrieved by running the following Frida script:

```
//frida -U -f <package> -l frida.js
function hookFunc() {
var dumpOffset = '0x000f9bdc' // Function to hook
 var argBufferSize = 150
 var baseApp = Module.findBaseAddress('App') // libapp.so (Android) or App (IOS)
 var codeOffset = baseApp.add(dumpOffset)
 console.log(")
 console.log('Wait.....')
 Interceptor.attach(codeOffset, {
    onEnter: function(args) {
      var objectPool = this.context.x27;
      var baseHeap = this.context.x23;
      var bitMask = 0xF00000000;
      var baseHeapMasked = baseHeap.and(0xF0000000);
      let modules = Process.enumerateRanges("r-");
      for (var i = 0; i < modules.length; i++) {
      if((modules[i].base.and(bitMask)).compare(baseHeapMasked) == 0)
console.log(modules[i].base);
      }
      console.log("App Binary Base Address: "+baseApp)
      console.log("Heap Base Address (X23): "+ baseHeap);
      console.log("Object Pool Address (X27): " + objectPool);
      console.log("Number of Objects in the Object Pool:
"+Memory.readPointer(objectPool.add(0x8)));
      console.log("Heap bitmask: "+baseHeap.and(0xF0000000))
    },
    onLeave: function(retval) {
   }
  });
 }
 setTimeout(hookFunc, 1000)
```



The output of the above script is as follows:

0 • •	\%1	frida -U -f com.example.minesweeper -l heapInfo.js
0x1c8188000		
0x1cefe8000		
0x1cf060000		
0x1d5ed8000		
0x1d5f64000		
0x1d8f00000		
0x1d8f3c000		
0x1dfd68000		
0x1dfda0000		
0x1e0000000		
0x1e1980000		
0x1e1988000		
0x1e1994000		
0x1e1998000		
0x1e2000000		
0x1e6bb8000		
0x1e6c24000		
0x1ed8f4000		
0x1ed988000		
0x1f2000000		
0x1f25c0000		
0x1f2624000		
0x1f26a8000		
0x1f664c000		
0x1f8000000		
0x1f8630000		
0x1f8678000		
0x1fa10c000		100 1000
App Binary	Base Address: 0	x106ae4000
Heap Base A	adress (X23): 0	X106/88001
Ubject Pool	Address (X27):	0x108180080
Number of 0	bjects in the O	DJECT POOL: 0X2505
Heap bitmas	K: 0X100000000	

Figure 14: Screenshot of Output from Frida Hook

When the script is executed multiple times, it can be observed that the Heap Base Address and Object Pool Address will change every time. The problem with dumping the entire application memory is that it can be very large and impractical. To address this issue, the script retrieves the heap base address, which in the example case is 0x1067880b1. It should be possible to dump memory addresses that start with 0x1...... which will contain the heap and object pool. It can be accomplished using Bitmask. Using this logic the script below dumps the flutter heap:



```
//frida -U -f <package> -l frida.js
var APP_DATA_DIR = "/var/mobile/Containers/Data/Application/--/Documents/" // make
sure that the directory is writable
function dump_memory(baseHeap, dump_directory){
 let modules = Process.enumerateRanges("r--");
 let i, module;
 let module_file;
  var bitMask = 0xF0000000; //bitMask to retain msb and set remaining to zero
  var baseHeapMasked = baseHeap.and(0xF0000000);
  module_file = new File(dump_directory + "ranges.json", "wb");
  module_file.write(JSON.stringify(modules, null, 2));
  module_file.close();
 for (i = 0; i < modules.length; i++) {
    try {
       module = modules[i];
      if ((modules[i].base.and(bitMask)).compare(baseHeapMasked) == 0) {
console.log(`Dumping memory into ${dump_directory + module.base}`);
module_file = new File(dump_directory + module.base, "wb");
module_file.write(module.base.readByteArray(module.size));
module_file.close();
      }
    } catch(ex) {
      console.log(ex);
      console.log(JSON.stringify(module, null, 2));
    }
 }
}
function hookFunc() {
  var dumpOffset = '0x000f9bdc' //restartGame (Function to hook)
   var baseApp = Module.findBaseAddress('App') // libapp.so (Android) or App (IOS)
   var codeOffset = baseApp.add(dumpOffset)
   console.log(")
   console.log('Wait.....')
   Interceptor.attach(codeOffset, {
     onEnter: function(args) {
var objectPool = this.context.x27;
var baseHeap = this.context.x23;
var bitMask = 0xF0000000;
var baseHeapMasked = baseHeap.and(0xF0000000);
let modules = Process.enumerateRanges("r-");
for (var i = 0; i < modules.length; i++) {
if((modules[i].base.and(bitMask)).compare(baseHeapMasked) == 0)
console.log(modules[i].base);
}
```



dump_memory(baseHeap, APP_DATA_DIR); console.log("App Binary Base Address: "+baseApp) console.log("Heap Base Address (X23): "+ baseHeap); console.log("Object Pool Address (X27): " + objectPool); console.log("Number of Objects in the Object Pool: "+Memory.readPointer(objectPool.add(0x8))); console.log("Heap bitmask: "+baseHeap.and(0xF00000000)) }, onLeave: function(retval) { } } } setTimeout(hookFunc, 1000)

The directory where the Flutter application is installed on the device is usually writable by the app itself. To determine the location of the app installation, the r2Frida tool can be used to print the relevant information, as shown in the figure below.

0 0 0 70381	r2 frida://attach/usb//minesweeper
<pre>Documents/IOS/Au</pre>	rticle via @^ v3.6.0 via ⊡
> r2 frida://att	.ach/usb//minesweeper
INF0: Mounted id) on /r2f at 0x0
Tip: do 'r2;	m -i r2premium; echo "e cfg.fortunes.type = nsfw" >> ~/.radare2rc' for a premium r2 experience
[@:100934000]>	.i
arch	arm
bits	64
os	darwin
pid	9936
uid	501
objc	true
runtime	QJS
swift	false
iava	false
mainLoop	true
pageSize	16384
pointerSize	8
modulename	Runner
modulebase	0x100934000
codeSigningPolic	cy optional
isDebuggerAttack	med false
cwd	/
bundle exename appname appversion	com.example.minesweeper Runner Minesweeper 0.1.0 1081344
minOS	12.0
homedir	12.0
tmpdir	/var/mobile/Containers/Data/Application/A42EB282-BC11-4584-8A45-751C49564BE5
bundledir	/private/var/mobile/Containers/Data/Application/A42EB282-BC11-4584-8A45-751C49564BE5/tmp/
[0x100934000]>	/private/var/containers/Bundle/Application/C7E9DD31-66FF-4A06-865C-8635A2BDFD6E/Runner.app

Figure 15: Screenshot of r2Frida output





Figure 16: Screenshot of Frida dumpMemory.js script

0 🔵 🌒 🔨 🕅	frida - U - f com.example.minesweeper - I dumpMemory.js
Dumping memory into	/var/mobile/Containers/Data/Application/A42E8282-BC11-4584-8A45-751C49564BE5/Documents/0x1c8188000
Dumping memory into	/var/mobile/Containers/Data/Application/A42EB282-BC11-4584-8A45-751C49564BE5/Documents/0x1cefe8000
Dumping memory into	/var/mobile/Containers/Data/Application/A42EB282-BC11-4584-8A45-751C49564BE5/Documents/0x1cf060000
Dumping memory into	/var/mobile/Containers/Data/Application/A42EB282-BC11-4584-8A45-751C495648E5/Documents/0x1d5ed8000
Dumping memory into	/var/mobile/Containers/Data/Application/A42EB282-BC11-4584-8A45-751C49564BE5/Documents/0x1d5f64000
Dumping memory into	/var/mobile/Containers/Data/Application/A42EB282-BC11-4584-8A45-751C49564BE5/Documents/0x1d8f00000
Dumping memory into	/var/mobile/Containers/Data/Application/A42EB282-BC11-4584-8A45-751C49564BE5/Documents/0x1d8f3c000
Dumping memory into	/var/mobile/Containers/Data/Application/A42EB282-BC11-4584-8A45-751C49564BE5/Documents/0x1dfd68000
Sumping memory into	/var/mobile/Containers/Data/Application/A42E8282-8C11-4584-8A45-751C495648E5/Documents/0x1dfda0000
Dumping memory into	/var/mobile/Containers/Data/Application/A42EB282-BC11-4584-8A45-751C495648E5/Documents/0x1e0000000
Dumping memory into	/var/mobile/Containers/Data/Application/A42EB282-BC11-4584-8A45-751C495648E5/Documents/0x1e1980000
Dumping memory into	/var/mobile/Containers/Data/Application/A42EB282-BC11-4584-8A45-751C495648E5/Documents/0x1e1988000
Dumping memory into	/var/mobile/Containers/Data/Application/A42E8282-8C11-4584-8A45-751C495648E5/Documents/0x1e1994000
Dumping memory into	/var/mobile/Containers/Data/Application/A42E8282-8C11-4584-8A45-751C495648E5/Documents/0x1e1998000
Dumping memory into	/var/mobile/Containers/Data/Application/A42EB282-BC11-4584-8A45-751C495648E5/Documents/0x1e2000000
Dumping memory into	/var/mobile/Containers/Data/Application/A42EB282-BC11-4584-8A45-751C495648E5/Documents/0x1e6bb8000
Dumping memory into	/var/mobile/Containers/Data/Application/A42E8282-8C11-4584-8A45-751C495648E5/Documents/0x1e6c24000
Dumping memory into	/var/mobile/Containers/Data/Application/A42E8282-8C11-4584-8A45-751C495648E5/Documents/0x1ed8f4000
Dumping memory into	/var/mobile/Containers/Data/Application/A42E8282-8C11-4584-8A45-751C495648E5/Documents/0x1ed988000
Dumping memory into	/var/mobile/Containers/Data/Application/A42EB282-BC11-4584-8A45-751C495648E5/Documents/0x1f2000000
Dumping memory into	/var/mobile/Containers/Data/Application/A42E8282-8C11-4584-8A45-751C495648E5/Documents/0x1f25c0000
Dumping memory into	/var/mobile/Containers/Data/Application/A42E8282-8C11-4584-8A45-751C495648E5/Documents/0x1f2624000
Dumping memory into	/var/mobile/Containers/Data/Application/A42E8282-8C11-4584-8A45-751C495648E5/Documents/0x1f26a8000
Dumping memory into	/var/mobile/Containers/Data/Application/A42EB282-BC11-4584-8A45-751C495648E5/Documents/0x1f664c000
Dumping memory into	/var/mobile/Containers/Data/Application/A42E8282-8C11-4584-8A45-751C495648E5/Documents/0x1f8000000
Dumping memory into	/var/mobile/Containers/Data/Application/A42E8282-8C11-4584-8A45-751C495648E5/Documents/0x1f8630000
Dumping memory into	/var/mobile/Containers/Data/Application/A42E8282-8C11-4584-8A45-751C495648E5/Documents/0x1f8678000
Dumping memory into	/var/mobile/Containers/Data/Application/A42EB282-BC11-4584-8A45-751C49564BE5/Documents/0x1fa10c000
App Binary Base Add	ress: 0x106058000
Heap Base Address (X23): 0x101f20e00
Object Pool Address	(X27): 0x107780080
Number of Objects i	n the Object Pool: 0x25b5
Heap bitmask: 0x100	000000

Figure 17: Screenshot of dumpMemory.js script in action



As shown in the figure above, all the memory addresses that start with 0x1...... were dumped as Heap Base Address starts with 0x1...... also note that Object Pool Address should be a part of heap. It is important to **note down all the above** addresses as they are required to be used in Ghidra. Copy the memory dump from the device to the PC.

K Back	Minesw	eeper 😒	\bigcirc
Q Search	1		
From 100 M	1B to 1 GB		Show All (9)
0x1a07880 00	0x1acb600 00	0x1c165c0 00	0x1c81880 00
From 10 MB	8 to 100 MB		Show All (19)
0x1a6ed80 00	0x1b3a0c0 00	0x1b659c0 00	0x1b8000 000

Figure 18: Screenshot of all the memory dumps



Importing Memory Dump in Ghidra

After copying the memory dump to the PC, it's time to start importing the memory dumps to Ghidra. Before doing that, the Base Address of the binary needs to be set to match it with the addresses in the dump file. To do that, go to Window > Memory Map



Figure 19: Screenshot of App Binary in Ghidra

Click on Set Image Base on the top right:

	Memory Map [CodeBrowser: minesweeper:/App/App]												aran		
File Edit Help													urun_		
a hrø-															
😃 Memory I	U Memory Map - Image Base: 00000000 + + = 〒 ± + × ★ = ×														
Name	St ⊾	End	Length				Volatile	Artificial	Overlay	Туре		Byte So	Source	Set Iman	a Rasa
TEXT	00000000	00003fff	0x4000							Default	2	AARCH6	AARCH6	IEAI	e base
text	00004000	001e1b7f	0x1ddb80	V		V				Default	V	AARCH6	AARCH6	TEXT	
const	001e1b80	0036488f	0x182d10							Default		AARCH6	AARCH6	_TEXT	
unwin	00364890	00367fff	0x3770	V						Default		AARCH6	AARCH6	TEXT	×10) =
LINKE	0036c000	0037a98f	0xe990							Default		AARCH6	AARCH6	_LINKE	:) = u
LINKE	0037a990	0037bfff	0x1670							Default		uninit[0x	AARCH6	_LINKE	×18)
EXTERNAL	0037c000	0037c007	0x8							Default		uninit[0x8]	Mach-O	NOTE: T	ng */
															1,1);
Filter:														🔁 🗄 ·	7) = 1
															+ (unde
															_i *)(1

Figure 20: Screenshot of App Binary in Ghidra Memory Map Window



Enter the Base Image Address that was retrieved during memory dump output:

Dumping memory into /var/mobile/Containers/Data/Applic 5-7516495648ES/Documents/Bh1F664c080 Dumping memory into /var/mobile/Containers/Data/Applic	otion/A42EB2 otion/A42EB2	82-8C11-458 82-8C11-458	нен <mark>и</mark> ен	S = 🖻 (• C	• • •	4 •	RM A		x C num		6.1.0
5-751C495648ES/Documents/Bklf8800000 Dumping memory into /var/mobile/Containers/Data/Applic		Memory Map (CodeBrowser: minesweepen; App(App)											
5-751C495648E5/Documents/0x1f8630000 Dumping memory into /var/mobile/Containers/Data/Apolic	File Edt Help												
5-751C495648E5/Documents/0x1f8678000	B 6161												
Dumping memory into /vor/mobile/Containers/Data/Applic 5-751C495648E5/Documents/Bx1fa10c000	I Memory											F±�×	🛕 🗏 🗙
op Binary Base Address: 0x106058000	Name	St k	End	Length			Base I	mage Address			. Byte So	Source	Comment
Heap Base Address (X23): 0x101f20e00 Object Pool Address (X27): 0x107f80080	_TEXT	00000000	00003111	0x4000					Defau	e (AARCH6	AARCH6	_TEXT
Number of Objects in the Object Pool: 0x25b5	_text	00024000	001e1b7f	0x1ddb80		0x106	058000	1	Defau	A 6	ARCH6	AARCH6	_TEXT
Heap bitmask: 0x100000000 Process terminated	const	00101080	00364881	8x182d18	N N				Defau	e 6 2 6	AARCHS	AARCHS	TEXT
[iPhone::com.example.minesweeper]->	_UNKE	0035c000	0037a98f	0xc990	ĕ				Defau	e 6	ARCH6	AARCH6	_UNKE
Thesh you for uring faidal	_LINKE	0037a990	00375fff	0x1670					Defau	# [uniniţ0x	AARCH6	_LINKE
mank you for using Pridat	EXTERNAL	0037c000	0037c007	0x8			-		Defau		uninit[0x8]	Mach-O	NOTE: T
Documents/IDS/Article via @^ v3.6.0 via @took 46s													
🍉 S	Filter:												🕘 🗄 •
^t T													

Figure 21: Screenshot of App Binary rebasing in Ghidra Memory Map Window



Copy the memory dumps to a dedicated folder:



Figure 22: Screenshot of Memory dumps in dedicated folder



The following Ghidra script can be used to import memory dumps:

import ghidra.app.script.GhidraScript; import ghidra.program.model.mem.Memory; import ghidra.program.model.mem.MemoryBlock; import ghidra.program.model.address.Address; import ghidra.util.task.TaskMonitor; import java.io.File; import java.io.FileInputStream; import java.io.BufferedInputStream; import java.io.DataInputStream; import java.io.IOException; public class ImportMemoryDump extends GhidraScript { @Override public void run() { try { // Ask the user to select a folder containing the memory dump files File folder = askDirectory("Select folder", "Import Memory Dump Files"); // Early exit if folder is not selected or invalid if (folder == null || !folder.exists() || !folder.isDirectory()) { println("Error: Invalid folder selected. Script aborted."); return; } // List all files in the selected folder File[] dumpFiles = folder.listFiles(); // Check if folder contains memory dump files if (dumpFiles == null || dumpFiles.length == 0) { println("No memory dump files found in the folder."); return; } // Process each memory dump file for (File dumpFile : dumpFiles) { try { importMemoryFromFile(dumpFile); } catch (Exception e) { // Catch exceptions for each individual file and log the error without stopping the script println("Error processing file: " + dumpFile.getName()); println("Exception: " + e.getMessage()); e.printStackTrace(); } } println("All memory dump files processed.");



```
} catch (Exception e) {
      // Catch any unexpected exceptions that occur during the entire script execution
      println("An unexpected error occurred during script execution.");
      println("Exception: " + e.getMessage());
      e.printStackTrace();
    }
 }
 /**
  * Imports a memory dump file into the current program.
  * @param dumpFile The file containing the memory dump.
  */
 private void importMemoryFromFile(File dumpFile) {
    try (FileInputStream fis = new FileInputStream(dumpFile);
BufferedInputStream bis = new BufferedInputStream(fis);
DataInputStream dis = new DataInputStream(bis)) {
      // Get the size of the memory dump file
      long fileSize = dumpFile.length();
      long baseAddress = extractBaseAddressFromFileName(dumpFile.getName());
      // Import the file as a memory block using the InputStream approach
      importMemoryBlockToGhidra(dumpFile.getName(), baseAddress, fis, fileSize);
      println("Successfully imported memory dump file: " + dumpFile.getName());
    } catch (IOException e) {
      // Handle file read exceptions
      println("Error reading memory dump file: " + dumpFile.getName());
      println(e.getMessage());
      e.printStackTrace();
    } catch (NumberFormatException e) {
      // Handle address parsing exceptions
      println("Error parsing address from file name: " + dumpFile.getName());
      println(e.getMessage());
      e.printStackTrace();
    } catch (Exception e) {
      // Catch any other unforeseen exceptions
      println("An unexpected error occurred with file: " + dumpFile.getName());
      println(e.getMessage());
      e.printStackTrace();
    }
 }
```



/**

* Extracts the base address from the file name, assuming the base address is

* encoded as hexadecimal starting from the 3rd character of the file name.

* @param fileName The name of the memory dump file.

* @return The base address as a long.

* @throws NumberFormatException If the address string is not a valid hexadecimal number.

*/

private long extractBaseAddressFromFileName(String fileName) throws NumberFormatException {

// Extract address from file name (assumed to start from the 3rd character)
String addressStr = fileName.substring(2);

return Long.parseLong(addressStr, 16);

}

/**

* Imports the memory dump data into the Ghidra program's memory at the specified address.

*

* @param blockName The name of the memory block (usually the dump file's name).

* @param baseAddress The base address at which to import the memory dump.

* @param inputStream The InputStream containing the memory dump data.

* @param length The length of the memory dump data.

*/

private void importMemoryBlockToGhidra(String blockName, long baseAddress, FileInputStream inputStream, long length) {

try {

Address address = toAddr(baseAddress); Memory memory = currentProgram.getMemory();

// Create a new memory block and initialize it with the InputStream

TaskMonitor monitor = getMonitor(); // You can optionally pass a TaskMonitor to track progress

MemoryBlock block = memory.createInitializedBlock(blockName, address, inputStream, length, monitor, false);

println("Memory block successfully created for file: " + blockName);

} catch (Exception e) {

// Handle errors related to importing memory

println("Error importing memory block: " + blockName);

println(e.getMessage());

e.printStackTrace();



}



After running the above script Memory Map should look like this:

•••				Men	tory	Map	CodeB	rowser: mi	nesweeper:/Ap	p/App]					
File Edit Hel	p														
🗧 🔿 t 🖓															
📕 Memory Map												+	4 ■ Ŧ ±	🔶 X 🏠	= ×
Name	Start ⊾	End	Length				Volatile	Artificial	Overlayed S	Type		Byte Source	Source	Com	ment
0x1006e4000	1006c4000	1006effff	0xc000							Default		init[0xc000]			
0x1006f0000	1006f0000	1006f3fff	8x4888	Ø						Default	Ø	init[0x4000]			
0x1006f4000	1006f4000	1006fbfff	0x8000							Default		init[0x8000]			
0x100700000	100700000	100703fff	8x4888	Ø						Default	Ø	init[0x4000]			
0x100704000	100704000	100713fff	0×10000							Default		init[0x10000]			
0x100714000	100714000	10071bfff	0x8000	2						Default	Ø	init[0x8000]			
0x10071c000	10071c000	100723fff	0x8000							Default		init[0x8000]			
0x100724000	100724000	100743fff	8×28888	Ø						Default	Ø	init[0x20000]			
0x100744000	100744000	100747fff	8×4888							Default		init[0x4000]			
0x100748000	100748000	10074bfff	8x4888	Ø						Default	Ø	init[0x4000]			
0x10074c000	10074c000	10074ffff	8×4888							Default		init[0x4000]			
0x100750000	100750000	100753fff	8×4888	Ø						Default	Ø	init[0x4000]			
0x100754000	100754000	100757fff	8×4888							Default		init[0x4000]			
0x1007ec000	1007ec000	1007effff	8×4888	Ø						Default	Ø	init[0x4000]			
0x100710000	1007f0000	1007f3fff	8×4888							Default		init[0x4000]			
0x100828000	100828000	10082bfff	8x4888	2						Default		init[0x4000]			
0x10082c000	10082c000	10082ffff	8×4888							Default		init[0x4000]			
0x100894000	100894000	10113ffff	0x8ac000	Ø						Default	Ø	init[0x8ac000]			
0x101140000	101140000	1011c7fff	0x88000							Default		init[0x88000]			
0x1011c8000	1011c8000	1011d3fff	0xc000	2						Default	Ø	init[0xc000]			
0x1011d4000	101104000	101217fff	8×44000							Default		init[0x44000]			
0x10121c000	10121c000	10122ffff	0×14000	Ø						Default	Ø	init[0x14000]			
0x101a20000	101a20000	101a33fff	0×14000							Default		init[0x14000]			
0x101be0000	101be0000	101be3fff	8×4888	Ø						Default	Ø	init[0x4000]			
0x101be4000	101be4000	101bf7fff	0×14000							Default		initI0x140000			
0x101bf8000	101bf8000	101bfbfff	8x4888	Ø						Default	Ø	init[0x4000]			
0x101bfc000	101bfc000	101bfffff	8x4888							Default	2	init[0x4000]			
0x101c00000	101c00000	101c13fff	0×14000	Ø						Default	Ø	init[0x14000]			
0x103468000	183468888									Default					
Filter:															2

Figure 23: Screenshot of Resolved Memory Map



Resolving Object Pool Array

After Importing Memory Dump in Ghidra and Jumping onto the Address where object pool array is located:

Stational Station - Station - Station - Station - Comparison - Station -									
	1	S 🗉 🖬	C 🚠	🕒 🗉	•	1 A I			
A45-751C49564BE5/Documents/0x1f8630000								1	
Dumping memory into /var/mobile/Containers/Data/Application/A42EB282-BC11-45B4-8	8					- h (1 2	. •• •	
A45-751C49564BE5/Documents/0x1f8678000	22	2	eeh						
Dumping memory into /var/mobile/Containers/Data/Application/A42EB282-BC11-45B4-8	8 77		eeh						- 18
A45-751C49564BE5/Documents/0x1fa10c000			eeh						
App Binary Base Address: 0x106058000			00h						
Heap Base Address (X23): 0x101f20e00			00h						
Object Pool Address (X27): 3x107780333	27		Øðh						
Number of Objects in the Object Pool: 0x25b5			00h						
Heap bitmask: 0x100000000			00h						
Process terminated			00h						
[iPhone::com.example.minesweeper]->			00h						
	11		eeh						
Thank you for using Frida!	11		eeh						
	11		een						
Documents/IOS/Article via @ v3.6.0 via 🗈 took 46s	11		een .						
			000						
			001						
orts 18778987a 88	33		0011						
10778007€ 00	22		aah						
niene m0 187788858 34	27		34h						
197780081 70	77		7/ah	D					
els 187788882 01	77		01h						

struct DartObjectTag	
{	
charis_canonical_and_gc;	
charsize_tag;	
int16cid;	
};	
The Dart object pool is a Dart object itself thus it starts with a DartObjectTag followed by the number of Dart objects in the pool (at offset 8) and ends with an array of pointers to Dart objects.	
struct DartObjectPool	
{	
DartObjectTagtag;	
int64nb_dart_objects_in_object_pool; DartObject*object_pool_array[]; };	

The addresses stored in the Dart object pool array are odd. Along with addresses, Small Integers (smi) are also stored on the array. To differentiate between Small Integers and Pointers, Small Integers have their LSB set to 0 and Pointers are made odd. Thus odd values on the Dart object pool array are pointers and even values are small integers. The real value of a Pointer is the value of Pointer subtracted by 1 (i.e. if ox1078881 is the value stored on Dart object pool, the real address would be 0x1078880). In case of Small Integers, the value must be right shifted by 1 (i.e. SMI value 50 resolves to integer value 50 > 1 = 25).

	DartObjectPool		Each Pointer Occupies 0x8 bytes
	DartObjectTag	Number of Objects in Object Pool	DartObject *object_pool_array[] (Array of Pointers to Dart Objects)
Located at Offset:	0x0	0x8	0x10



Ghidra needs to be instructed to create the object pool array. The following script can be used to create object pool array (Make sure to replace object pool address in the script) :

```
import ghidra.app.script.GhidraScript;
import ghidra.program.model.address.Address;
import ghidra.program.model.data.PointerDataType;
import ghidra.program.model.data.*;
public class DartObjectPoolPointers extends GhidraScript {
  @Override
 public void run() {
    try {
      // Define the base address of the object pool
      Address objectPoolBaseAddress = toAddr(0x107780080L);
      // Define the start address of the array (object pool base + 0x10)
      Address arrayStartAddress = objectPoolBaseAddress.add(0x10);
      // Define the end address of the array
      Address arrayEndAddress = objectPoolBaseAddress.add(0x8 * 0x25B5);
      // Define the data type for a pointer
      DataType pointerDataType = new PointerDataType();
      int pointerIndex = 0;
      // Traverse the memory addresses in the array
while (arrayStartAddress.compareTo(arrayEndAddress) <= 0) {
pointerIndex++;
try {
Address pointerAddress = toAddr(getLong(arrayStartAddress));
// Check if the pointer address is misaligned (odd)
if ((pointerAddress.getOffset() & 1) != 0) {
             println("Before: "+arrayStartAddress.toString());
              // Clear the existing data at this address
              clearListing(
                arrayStartAddress,
                arrayStartAddress.add(pointerDataType.getLength() - 1)
              );
              // Correct the misalignment
              setLong(arrayStartAddress, pointerAddress.getOffset());
              // Create a new pointer data type at this address
              createData(arrayStartAddress, pointerDataType);
              //println("Corrected pointer: " + pointerAddress);
              println("Pointer index: " + pointerIndex);
```







Figure 24: Screenshot of DartObjectPoolParser Ghidra Script

After running the above script, object pool array gets created and pointers to dart objects should be visible as shown below:



107780090	61 40 05 01	d1 00	addr	DAT_105d14061
107780098	00 00 e1 40 05 01	d1 00	addr	DAT_105d140e1
1077800a0	00 00 61 41 05 01	d1	addr	DAT_105d14161
1077800a8	00 00 e1 41	d1	addr	DAT_105d141e1
1077800b0	05 01 00 00 61 42	00 d1	addr	DAT_105d14261
1077000-0	05 01 00 00	00	e del a	
107780068	61 6C	aı	addr	DA1_105016C61

Figure 25: Screenshot of Dart Object Pool Array in Ghidra

Creating Dart Objects

After creating the pointers in the Dart Object Pool Array, Ghidra doesn't have any information about the structure of Dart Objects as shown below:

		DAT_105d140e1		
105d140e1	28	??	28h	(
105d140e2	01	??	01h	
105d140e3	00	??	00h	
105d140e4	e2	??	E2h	
105d140e5	3f	??	3Fh	?
105d140e6	06	??	06h	
105d140e7	00	??	00h	
105d140e8	d8	??	D8h	
105d140e9	26	??	26h	&
105d140ea	06	??	06h	
105d140eb	06	??	06h	
105d140ec	01	??	01h	
105d140ed	00	??	00h	
105d140ee	00	??	00h	
105d140ef	00	??	00h	
105d140f0	d8	??	D8h	

Figure 26: Screenshot of unresolved Dart Object in Ghidra



Identification of Dart Objects

The structure of Dart Deserialized Object is as follows (as evident from dart-sdk source code):

```
struct DartDeserializedObject
{
    char is_canonical_and_gc;
    char size_tag;
    __int16 cid;
    <intpadding>
    __int64 s_len; //s_len is stored as Small Integer so real length is s_len>>1;
    char s[];
};
```

In order to create the objects, the object pool array has to be traversed and the structure of the objects need to be created manually by writing Ghidra script. We will utilize the ClassId which is 0x5 or 5L (source link)



Figure 27: Screenshot of ClassId from Dart-SDK Source Code



The below script can be used to create Dart String Objects:

import ghidra.app.script.GhidraScript; import ghidra.program.model.address.Address; import ghidra.program.model.data.*; import ghidra.program.model.listing.*; import ghidra.program.model.mem.Memory; import ghidra.util.exception.CancelledException; public class DartObjectPoolParser extends GhidraScript { private static final long OBJECT_POOL_START_OFFSET = 0x10L; // Change this to the number of section as per your dump file private static final long NUM_OBJECTS = 0x25B5L; // Pointer size is 8 for aarch64 private static final int POINTER_SIZE = 8; @Override public void run() { try { // Define object pool boundary long objectPoolStartAddress = 0x107780080L + OBJECT_POOL_START_OFFSET; long objectPoolEndAddress = objectPoolStartAddress + (POINTER_SIZE * NUM_OBJECTS); Memory memory = currentProgram.getMemory(); byte[] pointerBytes = new byte[POINTER_SIZE]; // Iterate through the object pool for (long currentAddress = objectPoolStartAddress; currentAddress < objectPoolEndAddress; currentAddress += POINTER_SIZE) { monitor.checkCanceled(); // Handle script cancellation try { memory.getBytes(toAddr(currentAddress), pointerBytes); long objectAddress = bytesToLong(pointerBytes); if (objectAddress != 0) { processObject(memory, objectAddress, currentAddress); } catch (Exception e) { println("Error at address " + Long.toHexString(currentAddress) + ": " + e.getMessage()); } }



```
println("Dart object pool structures created successfully!");
    } catch (CancelledException e) {
       println("Script canceled by user.");
    } catch (Exception e) {
       println("Error: " + e.getMessage());
    }
 }
 private void processObject(Memory memory, long objectAddress, long pointerAddress)
throws Exception {
    byte[] structBytes = new byte[4];
    memory.getBytes(toAddr(objectAddress), structBytes);
    // Extract CID field
    short cid = (short) ((structBytes[2] & 0xFF) | ((structBytes[3] & 0xFF) << 8));
    // Filter objects based on CID values
    if (cid == 0x5) {
       println("CID: " + Integer.toHexString(cid));
       println("Pointer value at " + Long.toHexString(pointerAddress) + ": " +
Long.toHexString(objectAddress));
       createStringDataStructure(toAddr(objectAddress));
    }
 }
  private void createStringDataStructure(Address structAddress) {
    try {
       // Read string length and content
       long stringLength = getLong(structAddress.add(0x8)) >> 1;
       StringBuilder stringContent = new StringBuilder();
for (int i = 0; i < stringLength; i++) {</pre>
byte b = getByte(structAddress.add(0x10 + i));
stringContent.append((char) (b & 0xFF));
       }
       String stringName = stringContent.toString();
       // Define DartString structure
       Structure dartStringStruct = new StructureDataType(stringName, 0);
       dartStringStruct.add(ByteDataType.dataType, "is_canonical_and_gc", null);
       dartStringStruct.add(ByteDataType.dataType, "size_tag", null);
       dartStringStruct.add(ShortDataType.dataType, "cid", null);
       dartStringStruct.add(DWordDataType.dataType, "padding", null); // Padding for
alignment
       dartStringStruct.add(QWordDataType.dataType, "string_length", null);
       dartStringStruct.add(new ArrayDataType(CharDataType.dataType, (int)
stringLength, 1), "string_content", null);
```



```
DataUtilities.createData(
currentProgram,
structAddress,
dartStringStruct,
dartStringStruct.getLength(),
DataUtilities.ClearDataMode.CLEAR_ALL_UNDEFINED_CONFLICT_DATA
       );
       println("DartString structure created successfully at address " + structAddress);
    } catch (Exception e) {
       println("Failed to create DartString structure: " + e.getMessage());
    }
 }
 private long bytesToLong(byte[] bytes) {
    long value = 0;
    for (int i = bytes.length - 1; i \ge 0; i-) {
       value = (value << 8) | (bytes[i] & 0xFF);
    }
    // Adjust for object pool pointer odd values
    if (value > 0 && (value % 2L == 1L)) {
return value - 1;
    }
return 0L;
 }
}
```

After running the above script, it can be seen that Dart String Objects have been successfully created:

-									
1	106368540 32	e2	05			Ima	age::h		
	00	8e	3b						
	1b	38	1a						
	106368540	32					db	32h	is_canonical
	106368541	e2					db	E2h	size_tag
	106368542	05	00				short	5h	cid
	106368544	8e	3b	1b	38		ddw	381B3B8Eh	padding
	106368548	1a	00	00	00	00	dq	1Ah	string_length
		00	00	00					
	106368550	49	6d	61	67	65	char[13]	"Image::height"	string_content
		3a	3a	68	65	69			

Figure 28: Screenshot of Dart String Object



Before moving forward, one more change needs to be applied to the Dart Object Pool array as the pointers are odd in order to differentiate between Integer and the pointers in the array. In order for Ghidra to set the correct reference to string, the pointers should be pointing to start of the string. In order to resolve that, one should subtract all the pointers in the Dart Object Pool using the below script:

```
import ghidra.app.script.GhidraScript;
import ghidra.program.model.address.Address;
import ghidra.program.model.data.PointerDataType;
import ghidra.program.model.data.*;
public class DartObjectPoolPointers extends GhidraScript {
  @Override
  public void run() {
    try {
      // Define the base address of the object pool
      Address objectPoolBaseAddress = toAddr(0x107780080L);
      // Define the start address of the array (object pool base + 0x10)
      Address arrayStartAddress = objectPoolBaseAddress.add(0x10);
      // Define the end address of the array
      Address arrayEndAddress = objectPoolBaseAddress.add(0x8 * 0x25B5);
      // Define the data type for a pointer
      DataType pointerDataType = new PointerDataType();
      int pointerIndex = 0;
      // Traverse the memory addresses in the array
while (arrayStartAddress.compareTo(arrayEndAddress) <= 0) {
pointerIndex++;
try {
Address pointerAddress = toAddr(getLong(arrayStartAddress));
// Check if the pointer address is misaligned (odd)
if ((pointerAddress.getOffset() & 1) != 0) {
              //println("Misaligned address found at: " + arrayStartAddress);
             pointerAddress = pointerAddress.subtract(1);
             println("Before: "+arrayStartAddress.toString());
              // Clear the existing data at this address
              clearListing(
                arrayStartAddress,
                arrayStartAddress.add(pointerDataType.getLength() - 1)
              );
              // Correct the misalignment
              setLong(arrayStartAddress, pointerAddress.getOffset());
```





It can be seen that Object Pool Array pointers now point to the correct address:

	00	00				
107780108	00	63	88	addr	DAT_107886300	34h
	07	01	00			
	00	00				
107780110	60	Ød	34	addr	key_106340d60	
	06	01	00			
	00	00				
107780118	80	80	d0	addr	DAT_105d08080	72h
	05	01	00			
	00	00	~~	- 44-		
107780120	CØ QC	C5	23	addr	value_10623c5c0	
	00	01	00			
107700120	10	00	73	adda	DAT 100720010	
10//80128	10	00 01	73	addr	DA1_106738010	
	00	01	00			
107780130	40	69	88	addr	DAT 107886940	34h
107700150	07	01	88	0001	081_10/000340	5-111
	00	00				
107780138	10	bØ	23	addr	DAT 10623b010	32h
	06	01	00			
	00	00				
107780140	cØ	95	30	addr	dart:ui_1063095c0	
	06	01	00			
	00	00				
107780148	40	85	36	addr	Image::height_106368540	
	06	01	00			
	00	00				

Figure 29: Screenshot of Dart Pool Array with Correct Address



Adding References Between Code and Data

To add references between code and data, the register X27, which points to the Object Pool Array, needs to be provided a default value. Then, Ghidra will be able to add references between the code and data automatically.

In the figure shown below, it is still not known to Ghidra what object is being loaded, as the value of X27 is unavailable to Ghidra.

Listing: App			h 🖒	ъ 🖷 И	🕆 💵 - ×
	106151d70 45 f0 42 f8	ldur	x5.[x2. #0x2f]		
	106151d74 a0 f0 40 f8	ldur	x0.[x5, #0x1]		10
	106151d78 01 fc 41 93	asr	x1,x0,#0x1		
	106151d7c e0 03 01 aa	mov	x0,x1		
	106151d80 e1 03 04 aa	mov	x1,x4		
	106151d84 3f 00 00 eb	cmp	x1,x0		
	106151d88 42 05 00 54	b.cs	LAB_106151e30		
	106151d8c a0 70 41 f8	ldur	x0,[x5, #0x17]		
	106151d90 10 0c 04 8b	add	x16,x0,x4, LSL #0x3		
	106151d94 01 72 41 f8	ldur	x1,[x16, #0x17]		
	106151d98 ff 05 00 a9	stp	xzr,x1,[x15]		
	106151d9c 04 00 80 d2	mov	x4,#0x0		
	106151da0 e0 05 40 f9	ldr	x0,[x15, #0x8]		
4	106151da4 70 27 40 91	add	x16,x27,#0k9, LSL #12]	
	106151da8 10 c2 35 91	add	x16,x16,#0xd70		
	106151dac 05 7a 40 a9	ldp	x5,x30, <u>[x16]=>DAT_106061d70</u>		= F6h
	106151db0 c0 03 3f d6	blr	x30		
	106151db4 81 07 80 d2	mov	x1,#0x3c		
	106151058 60 00 00 36	tbz	w0,#0x0,LAB_106151dc4		
	106151dbc 01 10 51 18	ldur	x1,[x0, #-0x1]		
	106151dc0 21 /c 4c d3	UDTX	x1,x1,#0xc,#0x14		
		LAB 106151dc4		XREF[1]:	106151db
	106151dc4 ff 01 00 a9	stp	xzr.x0.[x15]		
	106151dc8 e0 03 01 aa	mov	x0.x1		
	106151dcc fe 03 00 aa	mov	x30.x0		
	106151dd0 be 7a 7e f8	ldr	x30,[x21, x30, LSL #0x3]		
	106151dd4 c0 03 3f d6	blr	x30		
	106151dd8 20 01 20 36	tbz	w0,#0×4,LAB_106151dfc		
	106151ddc a2 83 5e f8	ldur	x2,[x29, #-0x18]		
	106151de0 61 27 40 91	add	x1,x27,#0x9, LSL #12		
	106151de4 21 c0 46 f9	ldr	x1,[x1, #0xd80]=>DAT_106061d80		= EEh
	106151de8 70 2b 03 94	bl	FUN_10621cba8		
	106151dec a1 83 5f f8	ldur	x1,[x29, #-0x8]		
	106151df0 e2 03 00 aa	mov	x2,x0		
	106151df4 ad f3 fe 97	bl	setState		
+	106151df8 08 00 00 14	b	LAB_106151e18		

Figure 30: Screenshot of X27 register mystery

The value of the X27 register, which was retrieved during the memory dump, can be set.

As the TEXT/text section contains the code, the value of X27 can be replaced for the addresses that contain the TEXT/text section. Head over to the Memory Map to find the addresses. As shown in the figure below, the start address would be 0x106058000 and end address 0x106239b7f.



				Mer	nor;	y Ma	p [CodeB	rowser: m	inesweeper:/Ap	p/App]					l i
File Edit Hel															
a.															
U Memory Map												÷	+ ■ 〒 ± 4)×	≡ >
Name 🛡	Start	End	Length	R			Volatile	Artificial	Overlayed S	Type		Byte Source	Source	Comm	ent
EXTERNAL	1063d4000	1063d4007	0x8							Default		uninit[0x8]	Mach-O Loa	NOTE: Th	is
unwind info	1063bc890	1063bffff	8x3778	1	-		_			Default	1	AARCH64-64	AARCH64-64	TEXT	
text	10605c000	106239b7f	0x1ddb80	1						Default	1	AARCH64-64	AARCH64-64	TEXT	
TEXT	106058000	10605bfff	8x4888	Ø		1				Default	Ø	AARCH64-64	AARCH64-64	TEXT	
_UNKEDIT	1063d2990	1063d3fff	0×1670							Default		uninit[0x1670]	AARCH64-64	UNKED	лт
_UNKEDIT	1063c4000	1063d298f	0xe990							Default	1	AARCH64-64	AARCH64-64	UNKED	π
const	106239588	1063bc88f	0x182d10							Default	2	AARCH64-64	AARCH64-64	TEXT	
0x1fa10c000	1fa10c000	200173111	0x6e68000	Ø						Default	Ø	init[0x6e68000]			
0x1f8678000	1f8678000	1fa10bfff	0x1a94000							Default		init[0x1a94000]			
0x1f8630000	1f8630000	1f8677fff	0x48000	Ø						Default	1	init[0x48000]			
0x1f8000000	118000000	1f862ffff	0x630000							Default		init[0x630000]			
0x1f664c000	1f664c000	1f7ffffff	8x19b4888	Ø						Default	Ø	init[0x19b4000]			
0x1f26a8000	1f26a8000	1f464bfff	0x1fa4000							Default		init[0x1fa4000]			
0x1f2624000	1f2624000	1f26a7fff	0x84000	Ø						Default	Ø	init[0x84000]			
0x1f25c0000	1f25c0000	1f2623fff	0x64000							Default		init[0x64000]			
0x1f2000000	1f2000000	1f25bffff	0x5c0000	Ø						Default	2	init[0x5c0000]			
0x1ed988000	1ed988000	1f1ffffff	0x4678000							Default		init[0x4678000]			
0x1ed8f4000	1ed8f4000	led987fff	0x94000							Default		init[0x94000]			
0x1e6c24000	1e6c24000	1ed8f3fff	0x6cd0000							Default		init[0x6cd0000]			
0x1e6bb8000	1e6bb8000	1e6c23fff	0x6c000	Ø						Default	Ø	init[0x6c000]			
0x1e2000000	1e2000000	1e6bb7fff	0x4bb8000							Default		init[0x4bb8000]			
0x1e1998000	1e1998000	lelffffff	0x668000							Default		init[0x668000]			
0x1e1994000	1e1994000		0×4000							Default		init[0x4000]			
0x1e1988000	1e1988000	le1993fff	0xc000							Default		init[0xc000]			
0x1e1980000	1e1980000	1e1987fff	0668×0							Default		init[0x8000]			
0x1e0000000	1e0000000	le197ffff	0×1980000							Default		init[0x1980000]			
0x1dfda0000	1dfda0000		0x260000							Default		init[0x260000]			
0x1dfd68000	1dfd68000	1dfd9ffff	0x38000							Default		init[0x38000]			
0x1d8t3c000	1d8f3c888		8x6e2c888							Default		init/0x6e2c0001			
Filter:														-	

Figure 31: Screenshot of entire memory map

Run the script below:

```
import java.math.BigInteger;
import ghidra.app.script.GhidraScript;
import ghidra.program.model.address.Address;
import ghidra.program.model.lang.Register;
import ghidra.program.model.lang.RegisterValue;
import ghidra.program.model.listing.ProgramContext;
public class SetRegisterX27 extendsGhidraScript {
 @Override
 public void run() {
    try {
      // Define the register to modify
      Register x27Register= currentProgram.getRegister("x27");
      if (x27Register== null) {
         println("Register 'x27' not found in the current program.");
         return;
      }
      // Define the value to set for the register
      BigInteger registerValue= new BigInteger("107780080", 16); // Object Pool Address
      RegisterValue x27Value= new RegisterValue(x27Register, registerValue);
```



	// Define the address range to apply the value
	Address startAddress= toAddr(0x106058000L); // Replace with your start address
	Address endAddress= toAddr(0x106239B7FL); // Replace with your end address
	// Set the register value within the specified range
	ProgramContext programContext= currentProgram.getProgramContext();
	programContext.setRegisterValue(startAddress, endAddress, x27Value);
	println("Successfully set the register 'x27' value in the specified range.");
} c	catch (Exception e) {
	println("Error: " + e.getMessage());
}	
}	
, ,	
J	

Go to register manager as shown below:

		C	odeBrowser	: minesweeper:/App/App				
Select	Tools	Window Help		Čen				
DUL	F V	🗸 Bookmarks	жв	Cy 🚠 🜔 🗉 🔶 🔳 🔯 🛔				
🖪 Listin	a: App	Bundle Manager						
1		Bytes: App		(5, [x2, #0x2f]				
		Checksum Generator		(0,[x5, #0xf]				
		Comments		<1,x0,#0x1				
		💻 Console		<1,x4				
		🛅 Data Type Manager		<1,x0				
1		Data Type Preview		.AB_106151630 (0.[x5. #0x17]				
		Cf Decompile: revealBoxNumbers	ЖE	<16,x0,x4, LSL #0x3				
		I Defined Data		<1,[x16, #0x17] <zr.x1.[x15]< td=""></zr.x1.[x15]<>				
		Defined Strings		<4,#0×0				
æ		Disassembled View		<pre>k0,[x15, #0x8] <16,x27 #0x9, ISI #12</pre>				
		Equates Table		<16,x16,#0xd70				
		External Programs		<5,x30, <u>[x16]=>DAT_1066</u>				
		Function Call Graph		(30				
		S Function Call Trees		<1,#0x3c				
		🚓 Function Graph		<pre>x0,#0x0,LAB_1001510C4 x1,[x0, #-0x1]</pre>				
		Function Tags		<1,x1,#0xc,#0x14				
	i i	f Functions						
		ᇢ Jython		(zr,x0,[x15]				
		📧 Listing: App		(0,×1 (30,×0				
		🖽 Memory Map		<30,[x21, x30, LSL #0x3]				
		Program Trees		(30 				
		🔶 Register Manager		<2, [x29, #-0x18]				
		Relocation Table		<1,x27,#0x9, LSL #12				
		Script Manager		UN_10621cba8				
		🔥 String Search		<1,[x29, #-0x8]				
		Symbol References		setState				
l ÷	THE L	Symbol Table	ЖТ	.AB_106151e18				
		Symbol Tree						
DartObj	ole - Scr ectPool	String Search [CodeBrowser: mi	nesweep	واللافيدين ومعالكها				

Figure 31b: Screenshot of Register Manager



Make sure that the Start, and the Value should be set to the address of the Object Pool:

	Register Manager [Co	odeBro	wser: minesweeper:/App/App]	
File Edit Help				
🗄 🗠 t 🖉 t				
Register Manager				≥ × ≡
> 🔶 x21 (64)	Start Address		End Address	Value
> 🔶 x22 (64)	106058000		106239b7f	0×107780080
> 🔷 x23 (64)				
> 👽 x24 (64)				
> \Rightarrow x26 (64)				
√ 🔶 x27 (64)				
🔶 w27 (32)				
> 🔶 x28 (64)				
> 🔷 x29 (64)				
> 🔷 x3 (64)				
> • x4 (64)				
> 🔶 x5 (64)				
> 🔶 x6 (64)				
> 🔶 x7 (64)				
> 🔶 x8 (64)				
> 🔷 x9 (64)				
Filter:				

Figure 32: Screenshot of Register Manager Start and End Address

File	Edit	Analysis	Graph	Naviga	tion	Se	earc	h	Sele	ect	То	ols	w	ind	ow	Help			
	← •	Auto A	Analyze 'A	\pp'	А	8	Ι	D	U	L	F	v	В	•	~	ía	5	• (P I -
Prog	ram Tr	Analyz	ze All Ope	en			×		🗉 Lis	sting	у: А	рр							
~ 🔽	🌱 Ap	One S	hot				I					Т			1061	51d70	45	fØ	42
	90) 1	E Analvz	ze Stack												1061	51d74	a0	fØ	40
>															1061	51d78	01	fc	41
>	`	LINKEDIT													1061	51d7c	e0	03	01
	10	0x1998b800	00												1061	51d80	e1	03	04
	1	0x1b659c00	10												1061	51d84	3f	00	00
		0x1200-000	,0 10												1061	51d88	42	05	00
	100 A	0x17090000													1061	51d8c	a0	70	41
	1	0x10368800	0												1061	51d90	10	0c	04
	10	0x10074400	0												1061	51d94	01	72	41
	10	0x105d8000	00												1061	51d98	ff	05	00
	10	0x1006f400	0												1061	51d9c	04	00	80

Now Auto Analysis needs to be done:

Figure 33: Screenshot of Auto Analysis



C 106360e20 32 00 f1 106360e20 106360e20 106360e22 106360e22 106360e28 106360e28 106360e20 106360e20 106360e20	E2 05 T I M E 306336 1d 37 1 8 db	32h 5h 1EF1371Dh Eh 1 "T I M E" 	<pre>XREF(3): is_canonical size_tag cid padding string_length string_content</pre>	build:10(build:10(84 (Vari 85 *(lon 86 *(und 87 *(und 88 lVari 90 *(lon 90 *(lon 91 *(lon 92 lVari 93 *(lon 94 *(T I 95 lVari 95 iVari 96 *(lon 97 *(und 98 uVar2 99 *(und 90 iVar1
🗧 🧅 🌒 Reference	es to T I M E_106360e20	- 3 locations [CodeBrowser:	minesweeper:/App/A	.pp]	
Edit Help					
References to T I M E_106360e20 - 3				N the S =	🛯 🖬 🔺 🗶
Location	Label	Code Unit		Context	
106150568	۱	ldr x0=>T I M E_106360e2	0,[PARAM		
106150570	2	stur x0=>T I M E_106360e	20, DATA		
107789308 PTR_	TIME_107789308 a	addr T I M E_106360e20	DATA		
Filter:					🛃 🛱 •

It can be observed that the references between code and data have been resolved:

Figure 33b: Screenshot of Auto Analysis Result

106150528 0: 82 00 91 add x0,x22,#0x20 76 *(thorg + 106150528 0: 82 00 00 f8 stur x1,(x29, #=0x20) 77 *(undefi 106150534 0: 07 00 00 f8 stur x8,x27,#0x9, LSL #12 79 *(undefi 106150534 0: 07 00 3 f8 stur x8,x27,#0x9, LSL #12 99 *(undefi 106150536 20 70 03 f8 stur x8,x27,#0x9, LSL #12 99 *(undefi 106150536 20 70 03 f8 stur x8,x27,#0x9,LSL #12 99 *(undefi 106150540 10 09 44 bl FUL_106150bd8 99 *(undefi 106150554 0: 03 00 aa mov x1,x0 106150556 20 f0 00 f8 stur x8,x1, \$x0,11 106150556 20 f0 00 f8 stur 106150556 20 f0 00 f8 stur x8,x27,#0x9, LSL #12 99 *(long * 106150556 20 f0 00 f8 stur x8,x27,#0x9, LSL #12 99 *(long * 106150556 20 f0 00 f8 stur x8,x27,#0x9, LSL #12 99 *(long * 106150556 20 f0 00 f8 stur x8,x27,#0x9, LSL #12 99 *(long * 1061505566 00 f0 f6 stur x8,x27,#0x9, LS
10615052c al 03 1e f8 stur x1, [x29, #-0x20] 77 *(undefi 106150532 a0 70 00 f6 stur x0, [x1, #0x7] 78 uVar2 = 106150534 60 27 40 91 add x0, x27, #0x9, LSL #12 79 *(undefi 106150536 00 18 41 f9 Udr x0=x0AT_10670d9c0, [x1, #0x37] = 106 106150540 a6 01 00 94 bl FUN_106150bd8 undefi 106150544 e1 03 00 aa mov x1, x0 = 106 106150540 a6 01 00 94 bl FUN_106150bd8 undefi 106150540 a6 01 00 94 bl FUN_106150bd8 undefi 106150540 a6 01 00 94 bl x0, [x27, #0x6a70]=>DAT_106065ea70 = 006 106150542 a1 83 1c f8 stur x1, [x29, #-0x38] = 106 106150554 a0 03 5e f8 Idur x0, [x27, #0x9, LSL #12 = 006 106150555 20 f0 00 f8 stur x3, x0 = 106 94 106150556 a0 03 30 be f8 Idur x0, [x27, #0x9, LSL #12 = 006 = 106 106150556 a0 33 31 e f8 stur x3, (x29, #-0x20) = 106 94 *(Long * 1061505570 6e f0 00 f8 stur </td
106150530 20 70 00 f8 stur x0, [x1, #0x7] 106150534 60 27 40 91 add x0, x27, #0x9, LSL #12 106150538 00 18 41 f9 ldr x0=>DAT_10670d9c0, [x0, #0x230]=>DAT_106061230 = 001 106150536 20 70 03 f8 stur x0=>DAT_10670d9c0, [x1, #0x37] = 106 106150536 20 70 03 f8 stur x0=>DAT_10670d9c0, [x1, #0x37] = 001 106150540 a6 01 00 94 bl FUN_106150bd8 under 83 106150544 e1 03 00 aa mov x1, x0 = 106 85 *(long * 106150544 e1 03 00 aa mov x1, x0 = 106 85 *(long * 106150544 e1 03 00 aa mov x1, x0 = 106 86 *(long * 106150544 e1 03 00 f8 stur x1, [x29, #-0x38] = 106 86 *(long * 106150554 a0 03 5e f8 ldur x0, [x27, #0x9, LSL #12 = 001 86 *(lond f1 106150556 20 f0 00 1f8 stur x0, [x27, #0x9, LSL #12 = 106 97 *(lond f1 106150556 20 f0 00 1f8 stur x0, x27, #0x9, LSL #12 = 106 97 *(lond f1 106150556 60 f0 01
106150534 60 27 40 91 add x0,x27,#0x9, LSL #12 79 *(undefi 106150538 00 18 41 f9 Ldr x0=>DAT_10670d9c0, [x1, #0x230]=>DAT_106061230 = 09h 80 uVar2 = 10615053c 20 70 03 f8 stur x0=>DAT_10670d9c0, [x1, #0x37] = 106 81 *(undefi 10615053c 20 70 03 f8 stur x0=>LT_10670d9c0, [x1, #0x37] = 106 81 *(undefi 106150544 e1 03 00 aa mov x1, x0 = 106 83 Gestured 106150546 a1 83 1c f8 stur x1, [x29, #=0x38] = 106 86 *(long * 106150554 a0 03 5c f8 Ldur x0, [x27, #0x6a70]=>DAT_10605ea70 = 09h 86 *(long * 106150554 a0 03 5c f8 Ldur x0, [x29, #=0x20] = 106 88 !(undefi 106150556 c3 01 00 o9 4 b1 FUN_106150bd8 undefi 90 *(long * 106150556 c3 03 00 aa mov x3, x0 = 106 90h *(long * 106150556 c3 03 01 e16 stur x3, [x29, #=0x20] = 106 90h *(long * 106150556 c3 03 01 e16 stur x3, [x29, #=0x20] = 106<
106150538 00 18 41 f9 ldr x0=>DAT_10670d9c0, [x0, #0x230]=>DAT_106061230 = 00h = 80 uVar2 = 10615053c 20 70 03 f8 stur x0=>DAT_10670d9c0, [x1, #0x37] = 106 = 81 *(undef1 10615053c 20 70 03 f8 stur x0=>DAT_10670d9c0, [x1, #0x37] = = 83 #(undef1 106150544 e1 03 00 aa mov x1, x0 = = = 85 *(long # 106150544 e1 03 00 aa mov x1, x0 = = = 85 *(long # 106150544 e1 03 00 aa mov x1, x0 = = = 86 *(long # 106150544 e1 03 00 aa mov x1, x0 = = = 86 *(long # 106150546 e0 38 00 aa mov x1, [x29, #=0x38] = = = = = *(long # 106150554 e0 69 356 f8 lod 178 stur x0, [x27, #0x38] = = = = *(long # 1061505552 e0 f0 00 f8 stur x0, [x27, #0x9, LSL #12 = = = = = = <t< td=""></t<>
106150553c 20 70 03 f8 stur x0=>DAT_10670d9c0, [x1, #0x37] = 106 81 *(undefindefindefindefindefindefindefindefi
10615053c 20 70 03 f8 stur x0=>DAT_10670d9c0, [x1, #0x37] 82 #extraou 106150544 a6 01 00 94 bl FUN_106150bd8 undef 83 Gestured 106150544 a6 03 00 aa mov x1, x0 = 001 = 001 = 001 = 001 106150546 a1 83 1c f8 stur x1, [x29, #=0x38] = 001 = 001 = 85 *(long # 106150554 a0 03 5e f8 ldur x0, [x29, #=0x38] = 001 = 85 *(long # 106150554 a0 03 5e f8 ldur x0, [x1, #0x11] = 001 = 89 *(long # 106150556 20 f0 00 f8 stur x0, [x29, #=0x20] = 014 = 014 = 014 106150556 20 f0 00 f8 stur x0, [x1, #0x11] = 001 = 014 = 014 106150556 20 f0 00 f8 stur x0, [x1, #0x12] = 014 = 014 = 014 106150556 20 f0 00 f8 stur x0, [x1, #0x12] = 014 = 014 = 014 106150566 c0 30 30 aa mov x1, x0 = 016 = 014 = 014 = 014 106150566 c0 30 31 e f8 stur x3, [x29, #=0x20] = 016 </td
106158540 a6 01 00 94 bl FUN_106150bd8 undef 83 GestureD 106158544 e1 03 00 aa mov x1,x0 = 001 84 Uvar1 = 106158548 60 3b 75 f9 ldr x0,[x27, #0x6a70]=>DAT_10605ea70 = 001 86 *(long * 106158546 a1 83 1c f8 stur x1, [x29, #-0x38] = 106 86 *(long * 106158554 a0 03 5e f8 ldur x0,[x29, #-0x20] = 89 *(long * 106158555 20 f0 01 f8 stur x0,[x27,#0x6170] = 99 *(long * 106158555 20 f0 01 f8 stur x0,[x29, #-0x20] = 99 *(long * 106158556 20 f0 00 94 bl FUN_10615808 undef 91 *(long * 106158556 20 60 00 94 bl FUN_10615808 undef 91 *(long * 106158556 20 90 aa mov x3,x0 = 93 *(long * 106158556 20 90 444 19 ldr x0=-7 IM E 106360e20, [x0, #0x28]=>DAT_106061 901 *(long * 106158578 60 f0 00 f8 stur x3, [x29, #-0x20] = 96 *(long * 97 *(undefi
106158544 e1 03 00 aa mov x1,x0 a 001 a 84 Uvr1 = 106150548 60 3b 75 f9 ldr x0,[x27, #0x6a70]=>DAT_106005ea70 = 001 a 84 Uvr1 = 106150546 a1 83 1c f8 stur x1,[x29, #-0x38] = 106 87 *(long # 106150554 a0 83 5e f8 ldur x0,[x27, #0x6a70]=>DAT_106005ea70 = 001 88 *(long # 106150554 a0 83 5e f8 ldur x0,[x2, #-0x20] = 87 *(undefi 106150555 10 10 83 5e f8 ldur x0,[x1, #0x11] = 90 *(0 **)(106150556 c1 01 00 94 bl FUN_106150bd8 = 91 *(undefi 106150566 c3 03 00 aa mov x0,x27,#0x9, LSL #12 = 93 *(long # 106150566 c3 03 03 1e f8 stur x3,[x29, #-0x20] = = 93 *(long # 106150576 6e f0 00 f8 stur x3,[x29, #-0x20] = = 96 *(long # 106150576 6e f0 00 f8 stur x3,[x29, #-0x20] = = 96 *(undefi 106150576 6e f0 00 80 d2 mov x1,x22
106150548 60 3b 75 f9 ldr x0, [x27, #0x6a70]=>DAT_10605ea70 = 09h = 00h = 09h = 00h # (long # * (long # * (long # * (long # * (lond #) * (l
= 106 106150554 ca 1 83 1 c f8 stur x1, [x29, #-0x38] 106150556 20 f0 00 f8 stur x8==0_10623f600, [x1, #0xf] 106150554 a0 83 5e f8 ldur x0, [x29, #-0x20] 106150556 20 f0 01 f8 stur x0, [x1, #0x1f] 106150556 20 f0 01 f8 stur x0, [x1, #0x1f] 106150556 20 f0 00 f8 stur x3, x0 106150556 20 f0 00 f8 stur x3, x0 106150556 20 f0 00 f8 stur x3, x0 106150556 a0 44 41 f9 ldr x0==7 I H E 106360e20, [x0, #0x288]=>DAT 106061 = 001 106150576 a1 00 94 stur x3, [x29, #-0x20] 106150576 a1 00 86 stur x3, [x29, #-0x20] 106150576 a1 00 86 stur x3, [x29, #-0x20] 106150576 a1 00 86 stur x1, x22 106150576 a1 00 80 d2 mov x1, x22 106150578 a2 00 80 d2 mov x1, x24 106150578 a2 00 80 d2 mov x1, x24 106150578 a2 00 88 d2 mov x2, #0x4 8 var2 = 99 *(undef1 106150578 a2 00 88 d2 mov x2, #0x4 106150578 a2 00 88 d2 mov x2, #0x4 107 mov x2, #0x4 108 mov x2,
10615054 a0 83 1c f8 stur x1, [x29, #=0x38] 106150552 02 f0 00 f8 stur x8->[10623f600, [x1, #0xf] 106150554 a0 03 5c f8 ldur x0, [x29, #=0x20] 1061505554 a0 03 5c f8 ldur x0, [x29, #=0x20] 106150555 40 00 1 f8 stur x0, [x1, #0x1f] 106150556 e3 00 0a an mov x3, x0 106150556 e3 03 0a aa mov x3, x27, #0x9, LSL #12 106150556 e3 03 1e f8 stur x3, [x29, #=0x20] 106150576 60 f0 00 f8 stur x3, [x29, #=0x20] 106150578 82 00 80 d2 mov x2, #0x4 90 *(undefi 91 *(undefi 92 *(undefi 93 *(undefi 94 *(undefi 95
106150550 20 f0 00 f8 stur x0=0=10623f060, [x1, #0x1] 88 88 [Var1 = 106150554 a0 03 5c f8 1dur x0, [x2, # 0x20] 90 *(0 **)(106150558 20 f0 01 f8 stur x0, [x1, #0x1] 90 *(0 **)(106150556 20 f0 00 94 b1 FUN_106150b08 91 *(undef1 106150556 c3 03 00 aa mov x3, x0 92 1Var1 = 106150556 c3 03 00 aa mov x3, x27, #0x9, LSL #12 93 *(undef1 106150556 a0 04 44 11 f9 ldr x0=>T H E 106360e20, [x0, #0x288]=>DAT_106061 90 106150556 a3 03 1e f8 stur x3, [x29, #=0x20] 106150578 60 f0 00 f8 stur x3, [x29, #=0x20] 1061505778 60 f0 00 f8 stur x3, [x29, #=0x20] 96 *(undef1 106150578 61 00 80 d2 mov x1, x22 #0x4 97 *(undef1 06150578 62 00 80 d2 mov x1, x22 #0x4 99 *(undef1 061 References to T I M E_106360e20 - 3 locations [CodeBrowser: minesweeper:/App/App] 99 *(undef1 99 *(undef1 Help 10 10
100150534 30 03 5e 18 (du'r) x0, [x29, #=0x20] 89 89 *(long * 100150558 20 f0 01 f8 stur x0, [x1, #0x17] 90 *(0 **)(100150556 20 f0 00 94 bl FLN_106158bd8 91 *(undefi 100150556 02 03 00 aa mov x3, x0 92 1Var1 = 100150556 02 03 00 aa mov x3, x0 93 *(long * 100150556 02 03 00 aa mov x3, x0 93 *(long * 100150556 02 04 44 11 f9 ldr x0=>T M E 100360e20, [x0, #0x288]=>DAT_106061 = 000 93 *(long * 1001508566 03 03 1e f8 stur x3, [x29, #=0x20] 96 *(long * 1001508576 00 f0 00 f8 stur x1, x22 96 *(long * 1001508578 82 00 80 d2 mov x2, #0x4 99 *(undefi 80< O
100130530 20 10 00 94 bl FW_10(1, #0x11) 90 #10 #10 100150550 91 00 94 bl FW_106150508 undef 91 *(undefi 100150550 92 01 00 94 bl FW_106150508 undef 92 lVar1 = 100150550 92 01 00 94 add x0,x27,#0x9, LSL #12 93 *(long * 100150556 93 03 00 aa mov x3,x0 FW = 106366020, [x0, #0x288]=>DAT_106061 93 *(long * 100150556 a3 03 1e f8 stur x3, [x29, #-0x20] stur1 s106150576 60 f0 00 f8 stur x3, [x29, #-0x20] 100150576 60 f0 00 f8 stur x1, x22 96 *(long * 97 *(undefi 96 v/ar2 97 *(undefi 100150576 82 00 80 d2 mov x1, x22 99 *(undefi 99 *(undefi 00150578 82 00 80 d2 mov x1, x22 99 *(undefi 99 *(undefi 00150578 82 00 80 d2 mov x1, x22 99 *(undefi 99 *(undefi 00150578 82 00 80 d2 mov x2, #8x4 99 *(undefi 99 *(undefi 00 References to T I M E_1063600e20 - 3 locations [CodeBrowser: minesweeper:/App/App] 90 * (undefi 90 * (undefi
10013033C 91 01 00 94 01 rtw_100130030 01001 91 101001 1001305560 02 03 00 aa mov x3,x0 2 10411 93 10411 100150560 02 04 091 add x0,x27,#0x9, LSL #12 93 *(1ong * 100150566 a3 03 1e f8 stur x3=7 I M E 100360020, [x0, #0x288]=>0AT_106061 99 94 #(T I M 100150576 60 f0 00 f8 stur x3, [x29, #-0x20] 96 *(1ong * 1001505778 60 f0 00 f8 stur x2, #0x4 97 *(undefi 00150578 82 00 80 d2 mov x2, #0x4 99 *(undefi S ○ Ø References to T I M E_106360020 - 3 locations [CodeBrowser: minesweeper:/App/App] 99 *(undefi
106159566 65 63 08 aa mov x3, x27, #0x9, LSL #12 92 f(long + 106150566 60 27 40 91 add 106150566 60 27 40 91 add x0, x27, #0x9, LSL #12 94 #(T I M 106150566 a0 44 41 f9 ldr x0=> 106150660620, [x0, #0x288]=>DAT_106061 = 091 94 106150576 60 f0 00 f8 stur x3, [x29, #-0x20] = 106 96 *(long + 106150577 60 f0 00 f8 stur x0=>
100130304 00 27 40 91 add x0,x27,m0x3, C2C #12 s1 s3 s1(Unity #0x3), C2C #12 106150568 00 44 41 f9 ldr x0=>^T I M E_106360e20, [x0, #0x288]=>DAT_106061 e01 s1 106150576 c3 03 1e f8 stur x3, [x29, #-0x20] s106 s106 s106 106150576 60 f0 00 f8 stur x8=>^T I M E_106360e20, [x3, #0x1] s106 s106 s106 106150578 82 00 80 d2 mov x1, x22 s104 s10 s10 s10 8 O References to T I M E_106360e20 - 3 locations [CodeBrowser: minesweeper:/App/App] s1 s1 s1 Edit Help s1 s2 s1 s2 s1
106150500 00 44 41 15 Cui X0-1 10 Cui X0-1 10 Cui Y0-1 10 Cui
106150556c a3 03 1e f8 stur x3, [x29, #-0x20] 96 *(long + 106150570 60 f0 00 f8 stur x0-> T N E 106360e20, [x3, #0x1] 97 *(undef1 106150574 e1 03 16 aa mov x1, x22 98 uVar2 = 99 *(undef1 98< ○ ②
106158570 60 f0 00 f8 stur x0=>FI H E 106366e20, [x3, #0xf] 97 *(undefi 106150574 e1 03 16 aa 106150574 e1 03 16 aa mov x1, x22 98 uVar2 = 106150578 82 00 80 d2 mov x2, #8x4 99 *(undefi S ● Ø References to T I M E_106360e20 - 3 locations [CodeBrowser: minesweeper:/App/App] Edit Help
106150574 e1 03 16 aa mov x1,x22 98 98 y2r2 = 106150578 82 00 80 d2 mov x2,#0x4 99 *(undefine
106150578 82 00 80 d2 mov x2,#0x4 99 *(undefinition of the second o
References to T I M E_106360e20 - 3 locations [CodeBrowser: minesweeper:/App/App] Edit Help
Edit Help
Edit Help
References to T I M E_106360e20 - 3 locations
Location ⊾ Label Code Unit Context
106150568 [ldr x0=>T I M E_106360c20,[PARAM
106150570 stur x0=>T I M E_106360e20 DATA
107789308 PTR TIME 107789308 addr T I M E 106360e20 DATA
Filter:

Figure 34: Screenshot of Auto Analysis Result Cont.



Handling Dart VM's Custom Stack

Since Dart VM uses X15 as the stack pointer, tools like Ghidra fail to identify local variables and parameters. To resolve this:

- 1. Replace X15 with SP across relevant code ranges.
- 2. Set X27 (object pool register) to a known value in Ghidra's memory map.
- 3. Perform auto-analysis to establish cross-references between code and data.

The script below replaces X15 register with SP register for a given range of addresses:

```
import ghidra.app.script.GhidraScript;
import ghidra.program.model.address.Address;
import ghidra.program.model.address.AddressSet;
import ghidra.program.model.address.AddressSetView;
import ghidra.program.model.listing.Instruction;
import ghidra.program.model.listing.InstructionIterator;
import ghidra.program.model.listing.Listing;
import ghidra.program.model.lang.Register;
import ghidra.app.plugin.assembler.Assembler;
import ghidra.app.plugin.assembler.Assemblers;
public class ReplaceX15WithSP extends GhidraScript {
 @Override
 public void run() {
    // Retrieve the SP register
    Register spRegister = currentProgram.getProgramContext().getRegister("sp");
    // Retrieve the x15 register
    Register x15Register = currentProgram.getProgramContext().getRegister("x15");
    // Define the address range
    AddressSet addressRange = new AddressSet(
      toAddr("0x106058000L"),
      toAddr("0x106239B7FL")
    );
    // Replace occurrences of x15 with SP in the defined address range
    replaceRegisterInRange(addressRange, spRegister, x15Register);
 }
 private void replaceRegisterInRange(AddressSetView addressRange, Register
spRegister, Register x15Register) {
    Listing listing = currentProgram.getListing();
    Assembler assembler = Assemblers.getAssembler(currentProgram);
    InstructionIterator instructions = listing.getInstructions(addressRange, true);
```



```
// Iterate through instructions in the address range
while (instructions.hasNext() && !monitor.isCancelled()) {
       Instruction instruction = instructions.next();
       // Check each operand of the instruction for the x15 register
for (int operandIndex = 0; operandIndex < instruction.getNumOperands(); operandIndex++)
{
Object[] operands = instruction.getOpObjects(operandIndex);
// Replace x15 with SP if found
for (Object operand : operands) {
if (operand instanceof Register && operand.equals(x15Register)) {
              String patchedInstruction = instruction.toString()
                 .replace(x15Register.getName(), spRegister.getName());
              println("Patching: " + patchedInstruction);
              // Get the instruction's address
              Address instructionAddress = instruction.getAddress();
              try {
                 // Use the assembler to write the patched instruction to memory
                 assembler.assemble(instructionAddress, patchedInstruction);
              } catch (Exception e) {
                 println("Failed to patch instruction at address: " + instructionAddress);
                 e.printStackTrace();
              }
}
}
       }
    }
 }
}
```



Before running the script:

		FUNCTION						
	undefined GridV	iew.builder()						
	assume x27 =	0×107780080						
undefined	w0:1	<return></return>						
	GridView.builde	r	XREF [8]:					
1061508c4 fd 79 bf ag) stp	x29,x30,[x15, #-0x10]!						
1061508c8 fd 03 0f aa	a mov	x29,x15						
1061508cc ef 81 00 d1	l sub	x15,x15,#0x20						
1061508d0 e0 03 02 aa	a mov	x0,x2						
1061508d4 e4 03 01 aa	a mov	x4,x1						
1061508d8 a1 83 1f f8	3 stur	x1,[x29, #-0x8]						
1061508dc e1 03 06 aa	a mov	x1,x6						
1061508e0 a3 03 1f f8	3 stur	x3,[x29, #-0x10]						
1061508e4 a5 83 1e f8	3 stur	x5,[x29, #-0x18]						
1061508e8 a6 03 1e f8	3 stur	x6,[x29, #-0x20]						
1061508ec 80 70 09 f8	3 stur	x0,[x4, #0x97]						
1061508f0 90 f0 5f 38	3 ldurb	w16,[x4, #-0x1]						
1061508f4 11 f0 5f 38	3 ldurb	w17,[x0, #-0×1]						
1061508f8 30 0a 50 8a	and and	x16,x17,x16, LSR #0x2						
1061508fc 1f 82 5c ea	a tst	x16,x28, LSR #0x20						
106150900 40 00 00 54	b.eq	LAB_106150908						
106150904 c8 2d 03 94	↓ bl	FUN_10621c024						
			in an is a					
	LAB_106150908		XREF[1]:					
106150908 35 00 00 94	bl bl	FUN_1061509dc						
10615090c a1 03 5f f8	3 ldur	x1,[x29, #-0x10]						
106150910 01 70 00 f8	stur	x1,[x0, #0x7]						
106150914 a1 83 5e f8	3 ldur	x1,[x29, #-0x18]						

Figure 35: Screenshot of Disassembly Window before Stack handling script execution



After running the script:

		\$	
		FUNCTION	
	undefined GridV	iew.builder()	
	assume x27 =	0×107780080	
undefined	w0:1	<return></return>	
	GridView.builde	r	XREF[8]:
1061508c4 fd 7b bf a	ə stp	x29,x30,[sp, #-0x10]!	
1061508c8 fd 03 00 91	L mov	x29,sp	
1061508cc ff 83 00 d1	L sub	sp,sp,#0x20	
1061508d0 e0 03 02 aa	a mov	x0,x2	
1061508d4 e4 03 01 aa	a mov	x4,x1	
1061508d8 a1 83 1f f8	3 stur	x1,[x29, #-0x8]	
1061508dc e1 03 06 aa	a mov	x1,x6	
1061508e0 a3 03 1f f8	3 stur	x3,[x29, #-0x10]	
1061508e4 a5 83 1e f8	3 stur	x5,[x29, #-0x18]	
1061508e8 a6 03 1e f8	3 stur	x6,[x29, #-0x20]	
1061508ec 80 70 09 f8	3 stur	x0,[x4, #0x97]	
1061508f0 90 f0 5f 38	3 ldurb	w16,[x4, #-0x1]	
1061508f4 11 f0 5f 38	3 ldurb	w17,[x0, #-0×1]	
1061508f8 30 0a 50 8a	a and	x16,x17,x16, LSR #0x2	
1061508fc 1f 82 5c ea	a tst	x16,x28, LSR #0x20	
106150900 40 00 00 54	b.eq	LAB_106150908	
106150904 c8 2d 03 94	\$bl	FUN_10621c024	
	LAR 106150000		VDEE[1].
106150008 35 00 00 0	LAD_100120900	EIN 1061500dc	XKEF[1];
106150906 35 00 00 94	+ UL 2 Idur	v1 [v20 #_0v10]	
		1 [0 + 0, 10]	
106150910 01 70 00 10	3 Idur	v1 [v29 #_0v18]	

Figure 36: Screenshot of Disassembly Window After Stack handling script execution

After patching the stack pointer, the Dart stack is considered to be the regular stack by Ghidra. Next, let Ghidra know that the parameters it is looking for are in the stack, not in the registers that it would usually use.

Correcting Function Parameter Interpretation

Unlike ARM64's standard use of X0-X7 for parameters, Dart passes all parameters on the stack. Adjust function signatures in Ghidra manually to reflect this.

- 1. Navigate to the function signature editor.
- 2. Define parameters based on stack values.

This can be accomplished by using Edit Function Signature as shown below:



			- N - N	PH + I-			startGame 😚 🖧 R			÷ • ×
	******			***			and the data second			21
					115	old restarts	ane under unede conor		Tinese	paran_2)
							Edit Function Signature			
	underaned rest	derticker()			1.1	Long LVa	Rename Function			
	restartGame				6	undefiner				
				restartGame:000fb1	- 7	long in_:	Commit Params/Return			
				0036d8ec(+)		long una:	Commit Local Names			
beergodc fd 79 bf at	stp stp	x29,x30,[x15, #-0x10]!			1.2	Long una				
00019bc0 fd 03 0f a		x29, x15			10	undefine	Secondary Highlight			
00019be4 ef 21 00 d	L sub	x15,x15,#0×8			- H	underanei				
00019be8 a1 83 1f fi	stur	x1,[x29, #-0x8]			12	al and at the	Copy		1.000	
000195ec 50 11 40 1	ldr	x16,[x26, #2x38]			14	alundafi			a.	
00019010 TT 01 10 cl	cmp	x15,x10			15	+fundef1:	Commenta		· 2:	
66613014 29 62 66 5	0.13	LAB_00019C38			16	if (inux)	Compare Function(s)		x26	
	LAB_00019518		XREF[1]:		17					
00019518 21 00 80 d	2 mov	x1,#0×1			18		Find			
00019b1c fc 2a 03 9		FUN_001c47ec			19	tVar1 =	References			
000f9c00 el 03 00 a		x1,x0				uVar2				
00019c04 a0 83 51 11	ldur	x0,[x29, #-0x8]				-tunders	Properties		1 6 6	
00019c08 20 70 01 1	stur	x0,[x1, #0x17]			22	setState()	Included Loads all	n v15		al disc?)
00019C0C 62 03 01 a	nov	32,31			24	returns				
00019210 01 27 40 9	800	X1,X27,WERY, USL W12			251					
abador10 of 75 83 9	l uar	EIN ABIGADAR			26					
anaforic al Bi Sf fi	Idur	v1. [v29. #-0v1]								
88819c28 e2 83 88 a	mov.	x2.x8								
00019c24 21 14 fe 9	bl.	setState								
000f9c28 e0 03 16 a	a mov	x0,x22								
000f9c2c ef 03 1d a		x15,x29								
00019c30 fd 79 c1 al	l ldp	x29,x38,[x15], #8x18								
00019c34 c0 03 51 d										
	148.00010-18			000455-44143						
00019c38 41 21 03 0	L bl	FUN 001c5974		undefined stat d						
11111111111 41 A1 43 9										

Figure 37: Screenshot of edit function signature in Ghidra

•••		Edit Functi	on at 000f9bdc		00300000147				
void restartGame (undefined8 param_1, undefined8 param_2)									
Function Name: Calling Convention	restartGame		Fi I	unction Attri Varargs No Return	butes: In Line Use Custom Sto	orage			
	Datatype		Nama		Storage				
void	Datatype		Name	<void></void>	Otorage	E			
1 undefine	d8	param_1		x0:8		×			
2 undefine	d8	param_2		x1:8					
Call Fixup:									
-NONE-	~								
Commit all return	/parameter details								
		ОК	Cancel						

Figure 37b: Screenshot of edit function signature in Ghidra cont.

Conclusion

In this guide, we improved Ghidra's ability to decompile Flutter applications by:

- Establishing cross-references between Dart code and deserialized objects.
- Correcting the stack pointer usage for proper decompilation.
- Adjusting function signatures to align with Dart's parameter-passing model.

This methodology provides a strong foundation for reverse engineering Flutter apps on iOS, enabling deeper insights into their architecture and functionality.



About Security Innovation/ Bureau Veritas

Security Innovation is a leader in software security, providing comprehensive assessment solutions to secure software from design to deployment, across all environments, including web, cloud, IoT, and mobile. Leveraging decades of expertise and as part of Bureau Veritas, a global leader in Testing, Inspection, and Certification, we seamlessly integrate world-class security into development processes, safeguarding the way companies build and deliver products.

Security Innovation is a Bureau Veritas company. Bureau Veritas (BV) is a publicly listed company specialized in testing, inspection and certification. BV was founded in 1828, has over 80,000 employees and is active in 140 countries.

References

- Flutter and Dart Source Code/Documentation
- reFlutter Framework
- <u>Guardsquare Blog</u>
- Guardsquare Blog 2
- <u>Minesweeper Source YT</u>
- Dart Blog Numeric Computation



Interested?

Contact us today to start raising your cyber resilience.

Sisales@securityinnovation.com
+1 877 839 7598

securityinnovation.com