

iOS Reversing on a Budget: Frida, Ghidra, and a Jailbroken Device

Melvin Lee
Nicholas Lim

Self Introduction

Meet The Speakers



We are from PastelOps, an information security consultancy dedicated to helping organizations identify, manage, and defend against digital threats.



Melvin Lee (Main Speaker)

Senior Security Consultant

With over nine years of experience in offensive security, Melvin specializes in penetration testing and has a strong focus on mobile application security.



Nicholas Lim

Principle Security Consultant

With over a decade in offensive security, Nicholas has tested everything from legacy systems to modern SaaS, uncovering vulnerabilities that help organizations stay ahead.

Agenda



Things To Prepare

This section outlines the essential requirements for starting iOS application development.



Basics of iOS

This section introduces the fundamentals of iOS programming languages and compiled binaries.



Jailbreaking

This section covers the current state of jailbreaking and useful tweaks for iOS reverse engineering.



Using Ghidra

This section explores Ghidra's interface and the basics of decompiling and interpreting iOS applications.



Reversing Strategy

This section shares effective strategies we have used for reverse engineering iOS applications.

Things To
Prepare

Devices

<\$160

Device Specification

Model	iPad 6 th Gen (2018)
Latest Firmware	17.7.10 (Signed)
Jailbreak Method	palera1n*
Receiving New iOS?	No



Software

\$0

Software and Tools

Ghidra	Free
--------	------

Frida (Version 16.5.2)	Free
------------------------	------

iGameGod (App)	Free
----------------	------

Filza (App)	Free
-------------	------

Swift (Linux, macOS, Windows)	Free
-------------------------------	------

<https://github.com/NationalSecurityAgency/ghidra>

<https://github.com/frida>

<https://igamegod.app/repo/>

<https://www.tigisoftware.com/>

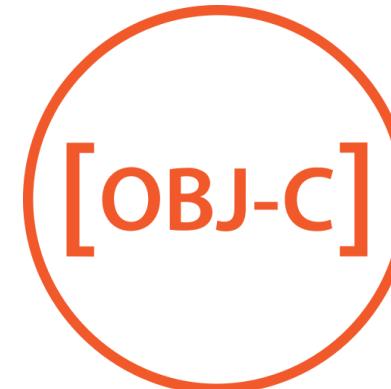
<https://www.swift.org/install/>



Basics of
ios

Programming Language

The primary programming languages used for developing iOS applications are Objective-C and Swift.

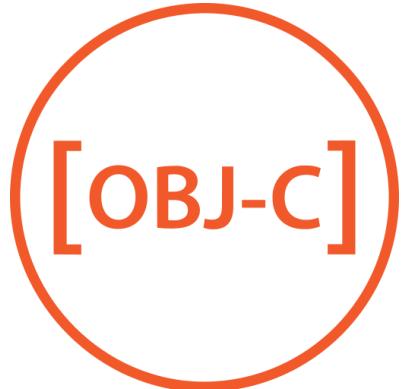


Objective-C



Swift

Objective-C

 [OBJ-C]

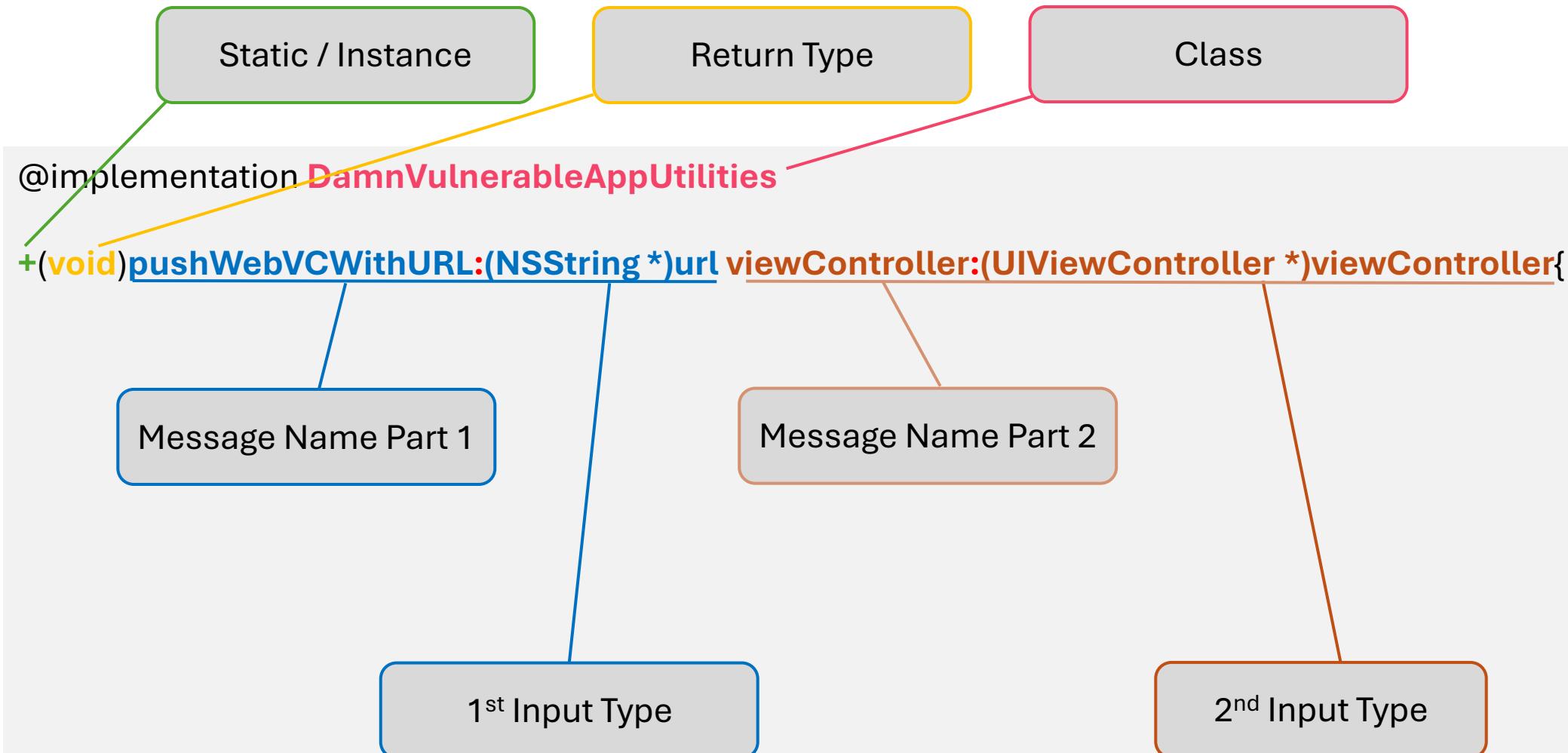
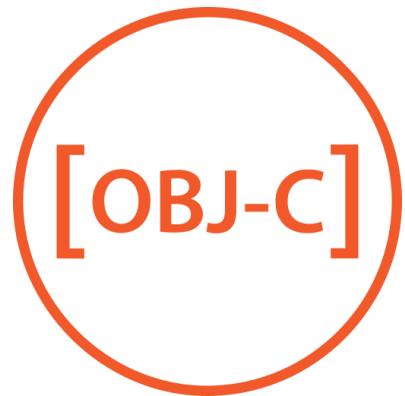
```
@implementation DamnVulnerableAppUtilities

+(void) pushWebVCWithURL:(NSString *)url viewController:(UIViewController *)viewController{
    UIStoryboard *mainStoryBoard = [UIStoryboard storyboardWithName:@"Main"
                                                               bundle:[NSBundle mainBundle]];

    GenericWebViewVC *vc = [mainStoryBoard
                           instantiateViewControllerWithIdentifier:@"GenericWebViewVC"];
    vc.url = url;
    vc.navigationItem.title = viewController.navigationItem.title;
    [viewController.navigationController pushViewController:vc animated:YES];

}
```

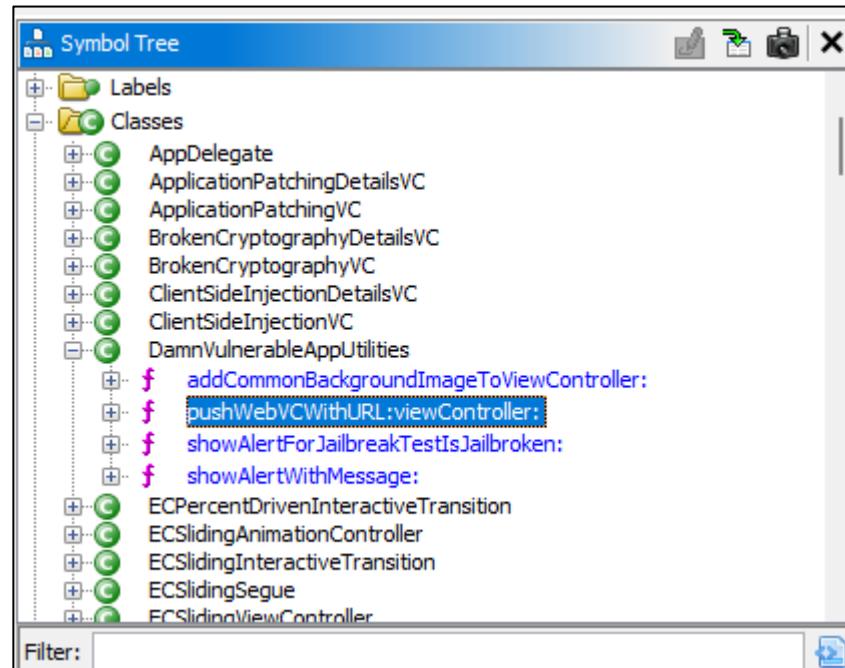
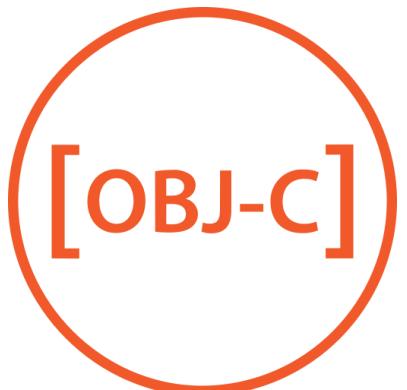
Objective-C



Objective-C

+[DamnVulnerableAppUtilities pushWebVCWithURL:viewController:]

(Ghidra)



(Frida-Trace)

```
c:\tmp> frida-trace -U -f com.highaltitudehacks.dvia -m "*[DamnVulnerableAppUtilities pushWebVC*]"
Waiting for USB device to appear...
Instrumenting...
+[DamnVulnerableAppUtilities pushWebVCWithURL:viewController:]: Auto-generated handler at "c:\tmp\newController_.js"
Started tracing 1 function. Web UI available at http://localhost:23880/
```

Swift



```
class DVIAUtilities {

    class func loadWebView(withURL url: String, viewController: UIViewController) {
        if UIApplication.shared.canOpenURL(URL(string: url)!) {
            guard let url = URL(string: url) else { return }
            if #available(iOS 10.0, *) {
                UIApplication.shared.open(url, options: [:], completionHandler: nil)
            } else {
                // Fallback on earlier versions
                UIApplication.shared.openURL(url)
            }
        }
    }
}
```

Swift



```
class DVIAUtilities {  
    class func loadWebView(withURL url: String, viewController: UIViewController) {
```

Class

Optional Alias

Method Name

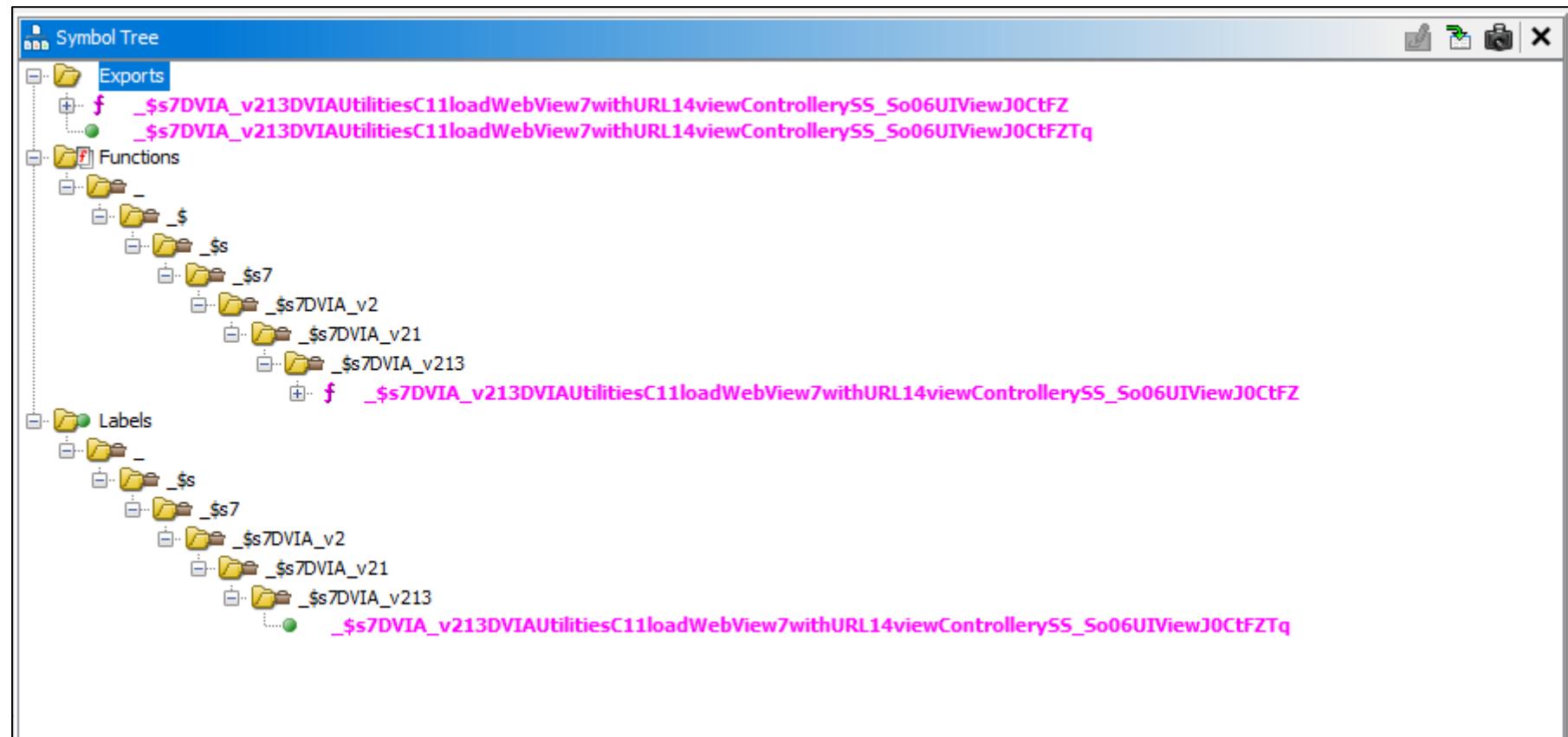
1st Parameter

2nd Parameter

Swift



(Ghidra)



Swift



`_\$s7DVIA_v213DVIAUtilitiesC11loadWebView7withURL14viewControllerySS_So06UIViewJ0CtFZ`

Prefix: `_\$s`

- `_` – Part of the standard symbol format
- `\$s` – Swift Symbols (`$S` for old mangling style pre-swift 4)

Module Name: `7DVIA_v2`

Type: `13DVIAUtilitiesC`

- `C` – Class Type (V for struct, O for enum, P for protocol)

Function Name: `11loadWebView`

1st Parameter: `7withURL`

2nd Parameter: `14viewController`

`y` Separator: Used by Swift to separate parameter labels from their type list.

Parameter Types: `SS_So06UIViewJ0C`

- `SS` – Swift String
- `_` – Separator for parameters
- `So06UIViewJ0C` – Imported symbol from Objective-C and with some name substitution for shorting the name reference

`t` – Tuple end marker for parameters

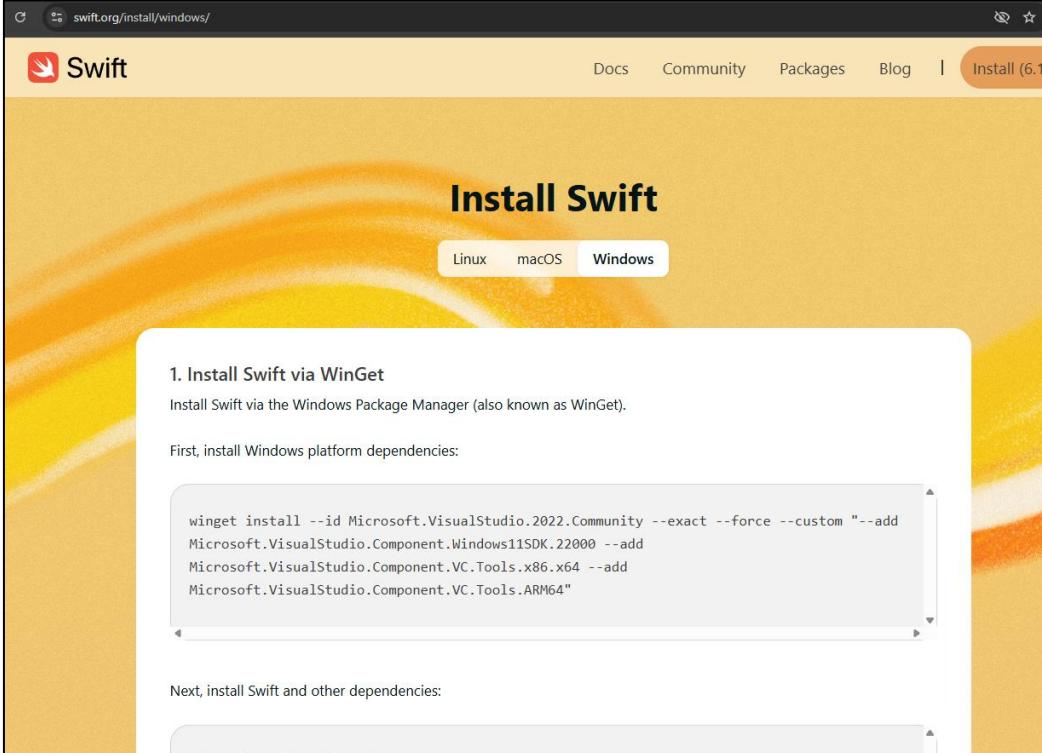
`F` – Function signature

`Z` – Represents static/class method (FZ means function & static)

Swift



<https://www.swift.org/install/windows/>



The screenshot shows the "Install Swift" section of the Swift website for Windows. It features a large orange and yellow background image. At the top, there are tabs for "Linux", "macOS", and "Windows". Below the tabs, a heading says "1. Install Swift via WinGet". It provides instructions to install Swift via the Windows Package Manager (WinGet) and lists platform dependencies. A command-line snippet for WinGet is shown:

```
winget install --id Microsoft.VisualStudio.2022.Community --exact --force --custom "--add Microsoft.VisualStudio.Component.Windows11SDK.22000 --add Microsoft.VisualStudio.Component.VC.Tools.x86.x64 --add Microsoft.VisualStudio.Component.VC.Tools.ARM64"
```

Below this, it says "Next, install Swift and other dependencies:" followed by another command-line snippet.

Paste in

```
c:\>swift demangle -compact  
_$_s7DVIA_v213DVIAUtilitiesC11loadWebView7withURL14viewControllerySS_So06UIViewJ0CtFZ  
static DVIA_v2.DVIAUtilities.loadWebView(withURL: Swift.String, viewController: __C.UIViewController) -> ()
```

Mach-O Binaries

TEXT Segment

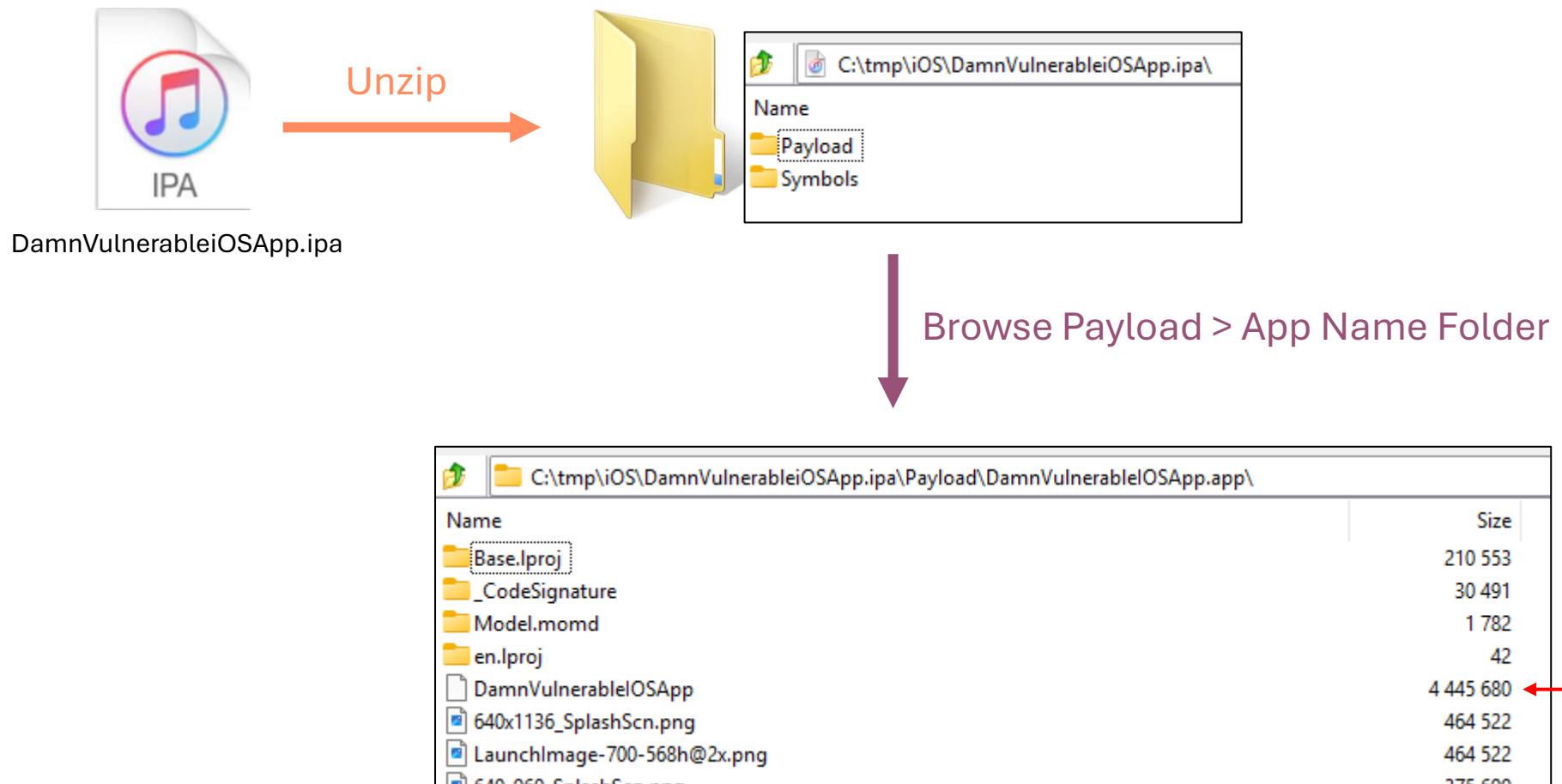
Section	Description
<code>_text</code>	The compiled machine code for the executable
<code>_const</code>	The general constant data for the executable
<code>_cstring</code>	Literal string constants (quoted strings in source code)
<code>_picsymbol_stub</code>	Position-independent code stub routes used by the dynamic linker (dyld)

DATA Segment

Section	Description
<code>_data</code>	Initialized global variables (for example <code>int a = 1;</code> or <code>static int a = 1;</code>).
<code>_const</code>	Constant data needing relocation (for example, <code>char * const p = "foo";</code>).
<code>_bss</code>	Uninitialized static variables (for example, <code>static int a;</code>).
<code>_common</code>	Uninitialized external globals (for example, <code>int a;</code> outside function blocks).
<code>_dyld</code>	A placeholder section, used by the dynamic linker.
<code>_la_symbol_ptr</code>	"Lazy" symbol pointers. Symbol pointers for each undefined function called by the executable.
<code>_nl_symbol_ptr</code>	"Non lazy" symbol pointers. Symbol pointers for each undefined data symbol referenced by the executable.

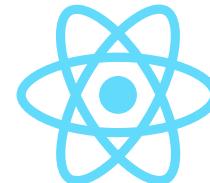
iOS applications are typically compiled into the Mach-O file format, an important detail to understand when performing reverse engineering.

Mach-O Binaries





Basics of iOS



React Native



Kotlin
Multiplatform



Cross Platform Frameworks

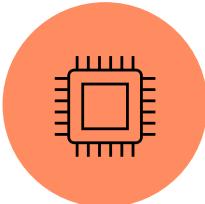
This is not covered in this talk, as these frameworks typically run within their own virtual machine or engine.

Obfuscators



Jailbreaking

State of Things

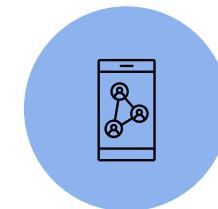


As of today, jailbreaking has become increasingly restricted, with the notable exception of the renowned checkm8 hardware exploit discovered in 2019.

Devices with chips up to A11 remain jailbreakable.



Updated Inventory Database
<https://appledb.dev>



Current Jailbreakable iOS versions (Recent)
iPad (7th generation) (2019) iPadOS 18.3.2 (Not Signed) (palera1n)
iPad (6th generation) (2018) iPadOS 17.7.10 (**Signed**) (palera1n)
iPad Pro 10.5" / 12.9" (2nd gen 2017) iPadOS 17.7.10 (**Signed**) (palera1n)



Future iOS Device Support 
Upcoming iOS version 26 (Only available for A12 chips and above)

Useful Apps & Tools



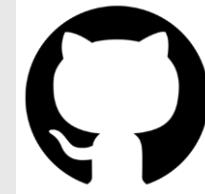
Filza

Useful for File Transfer



iGameGod

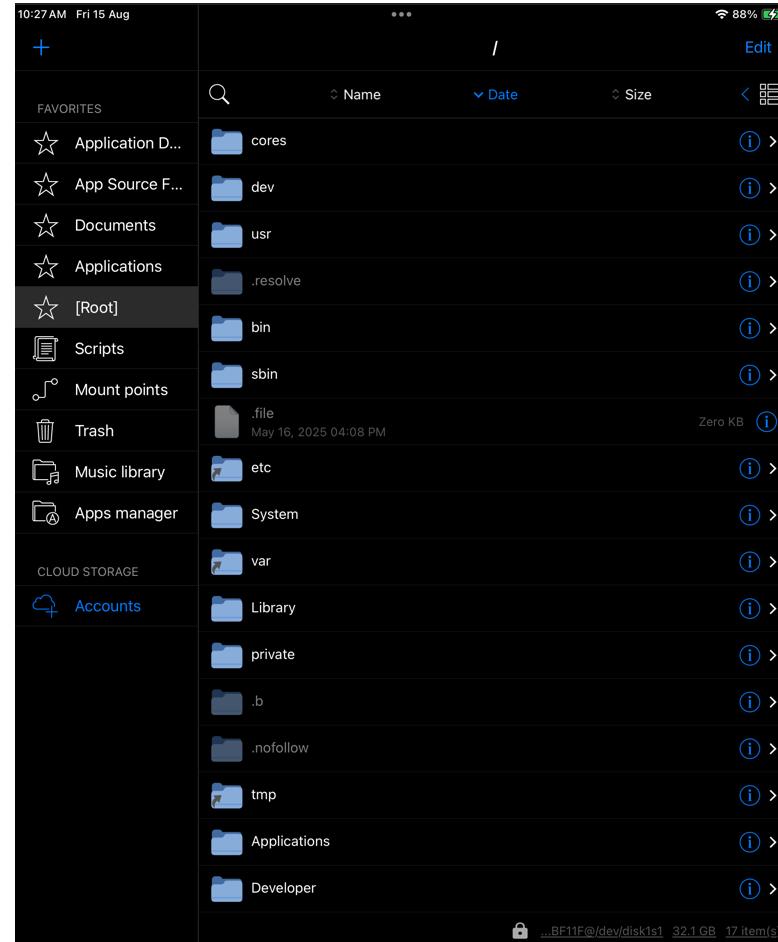
Useful for decrypting and extracting IPA files



frida-ios-dump

In case iGameGod fails to decrypt IPA files

Useful Apps & Tools



Not secure 192.168.1.129:11111

/ Root < Back Current folder: /

Open in Filza Install Upload New folder

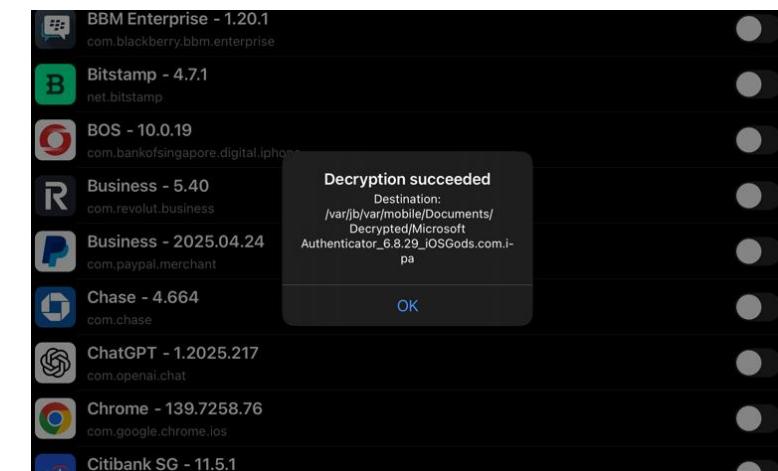
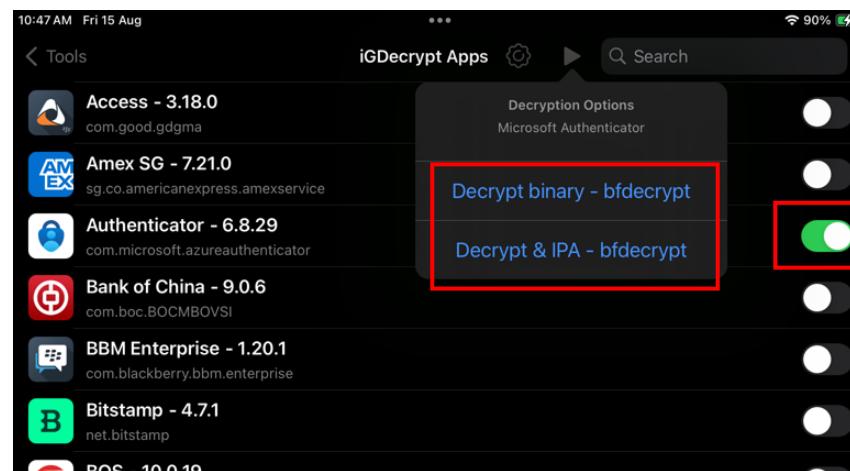
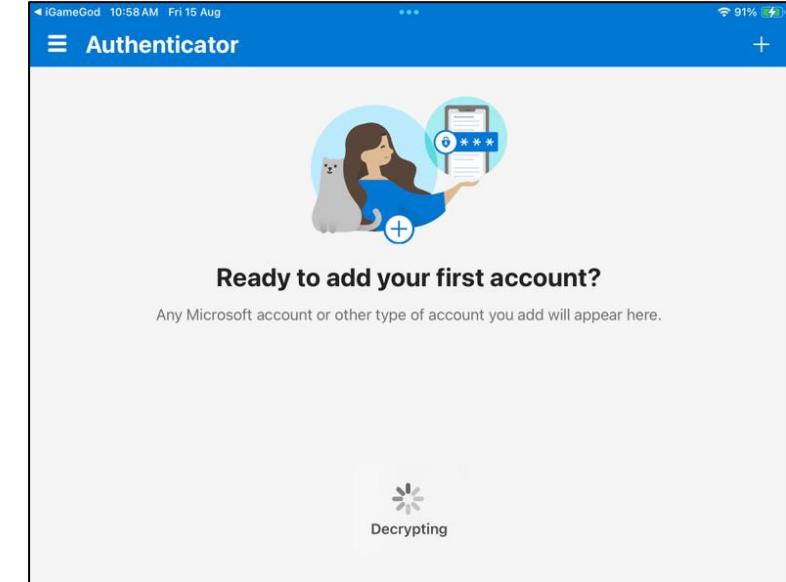
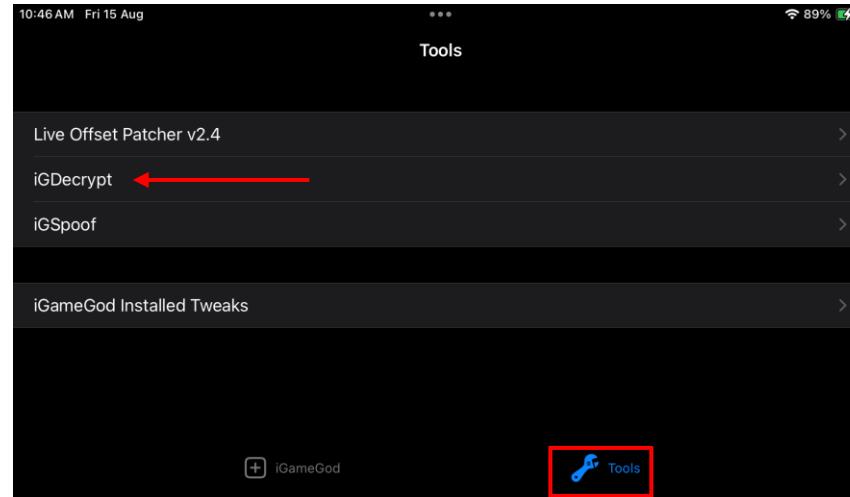
Name	Permissions	Size	Modified
.ba	drwx-----	64 bytes	Jul 16, 2024 10:56 AM
.mb	-----	0 bytes	Jul 16, 2024 10:56 AM
Applications	drwxrwxr-x	3.56kb	Jul 16, 2024 10:56 AM
bin	drwxr-xr-x	128 bytes	Jul 16, 2024 10:56 AM
cores	drwxrwxr-t	64 bytes	Jul 16, 2024 10:56 AM
dev	dr-xr-xr-x	1.37kb	Jul 31, 2025 10:55 PM
Developer	drwxrwxr-x	64 bytes	Jul 16, 2024 10:56 AM
etc	drwxr-xr-x	11 bytes	Jul 16, 2024 10:56 AM
Library	drwxr-xr-x	672 bytes	Jul 16, 2024 10:56 AM
private	drwxr-xr-x	224 bytes	Jul 16, 2024 10:56 AM
sbin	drwxr-xr-x	576 bytes	Jul 16, 2024 10:56 AM
System	drwxr-xr-x	160 bytes	Jul 16, 2024 10:56 AM
tmp	drwxr-xr-x	15 bytes	Jul 16, 2024 10:56 AM
usr	drwxr-xr-x	256 bytes	Jul 16, 2024 10:56 AM
var	drwxr-xr-x	11 bytes	Jul 16, 2024 10:56 AM

Convenient Functions

Useful Apps & Tools



iGameGod



Useful Apps & Tools



frida-ios-dump

```
(kali㉿kali)-[~/Desktop/frida-ios-dump]
└─$ ./dump.py -H 192.168.1.129 -p 22 -u mobile -K ~/.ssh/id_ed25519 sg.ndi.sp
Start the target app sg.ndi.sp
Dumping SingPass Mobile to /tmp
[frida-ios-dump]: Load SPMNetworking.framework success.
[frida-ios-dump]: Load SPMBioWrapper.framework success.
[frida-ios-dump]: Load GoogleAppMeasurementIdentitySupport.framework success.
[frida-ios-dump]: Load iProov.framework success.
[frida-ios-dump]: Load NuDetectSDK.framework success.
[frida-ios-dump]: Load SPMTheme.framework success.
[frida-ios-dump]: Load Lottie.framework success.
[frida-ios-dump]: Load GoogleAppMeasurement.framework success.
[frida-ios-dump]: Load SPMSecurity.framework success.
[frida-ios-dump]: Load RecaptchaEnterpriseSDK.framework success.
[frida-ios-dump]: Load SPMLogger.framework success.
[frida-ios-dump]: Load FirebaseAnalytics.framework success.
start dump /private/var/containers/Bundle/Application/DCF2AFE1-F3E3-43E7-AD38-83496C853688/SingPass.app/SingPass
SingPass.fid: 100%|██████████| 96.2M/96.2M [01:11<00:00,
start dump /private/var/containers/Bundle/Application/DCF2AFE1-F3E3-43E7-AD38-83496C853688/SingPass.app/Frameworks/SPMB
framework/SPMBioWrapper
SPMBioWrapper.fid: 100%|██████████| 308k/308k [00:00<00:00
start dump /private/var/containers/Bundle/Application/DCF2AFE1-F3E3-43E7-AD38-83496C853688/SingPass.app/Frameworks/SPMN
framework/SPMNetworking
SPMNetworking.fid: 100%|██████████| 223k/223k [00:00<00:00
start dump /private/var/containers/Bundle/Application/DCF2AFE1-F3E3-43E7-AD38-83496C853688/SingPass.app/Frameworks/iPro
pripriseSDK.framework/RecaptchaEnterpriseSDK
RecaptchaEnterpriseSDK.fid: 100%|██████████| 34.8k/34.8k [00:00<00:00, 107kB
start dump /private/var/containers/Bundle/Application/DCF2AFE1-F3E3-43E7-AD38-83496C853688/SingPass.app/Frameworks/FirebaseAn
alytics.framework/FirebaseAnalytics
FirebaseAnalytics.fid: 100%|██████████| 34.8k/34.8k [00:00<00:00, 142kB
SPMInfoBannerView.nib: 117MB [01:38, 1.25MB/s]
0.00B [00:00, ?B/s]
Generating "SingPass Mobile.ipa"
```

Useful Apps & Tools



frida-ios-dump

Note: frida-ios-dump is incompatible with iGameGod:
<https://github.com/frida/frida/issues/2588>

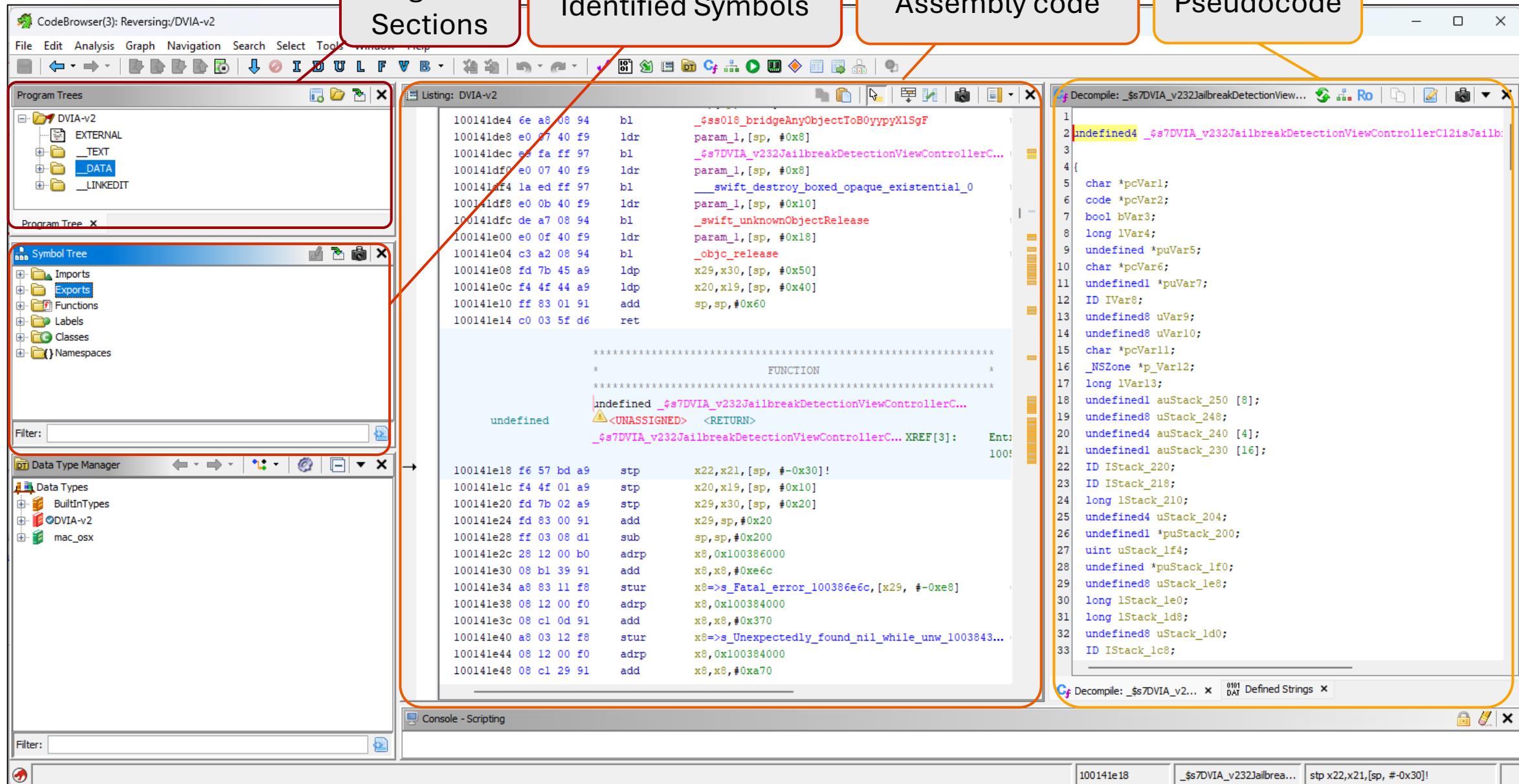
```
ics.framework/FirebaseAnalytics
FirebaseAnalytics.fid: 100%|██████████| 34.8k/34.8k [00:00<00:00, 113kB/s]
SPMInfoBannerView.nib: 117MB [02:28, 823kB/s]
0.00B [00:00, ?B/s]
Generating "SingPass Mobile.ipa"
[Errno 2] No such file or directory: '/tmp/Payload/SingPass.app/SharedFrameworks/libloader.dylib'
```

Need to modify dump.js to skip the following file:
/SharedFrameworks/libloader.dylib

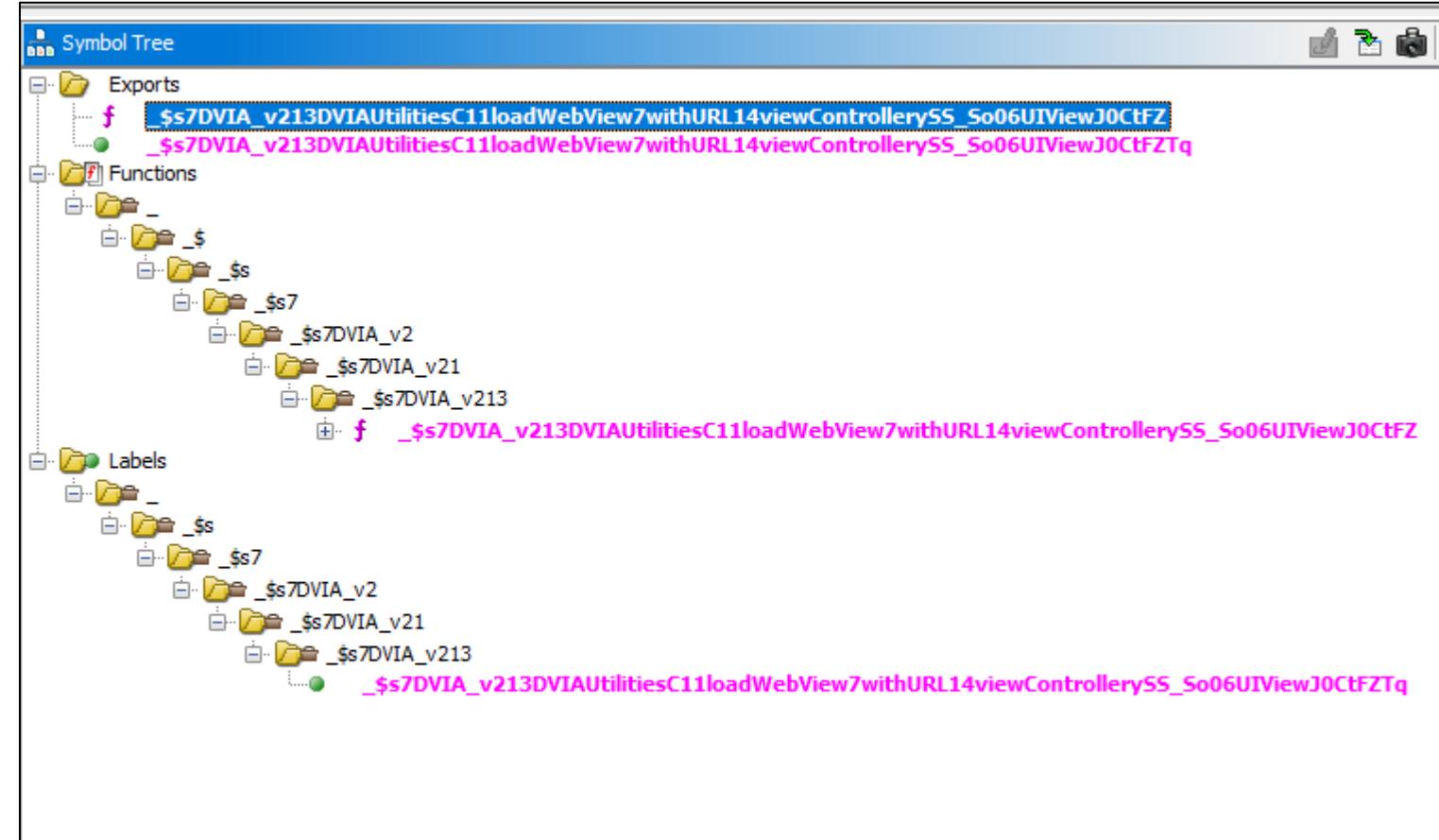
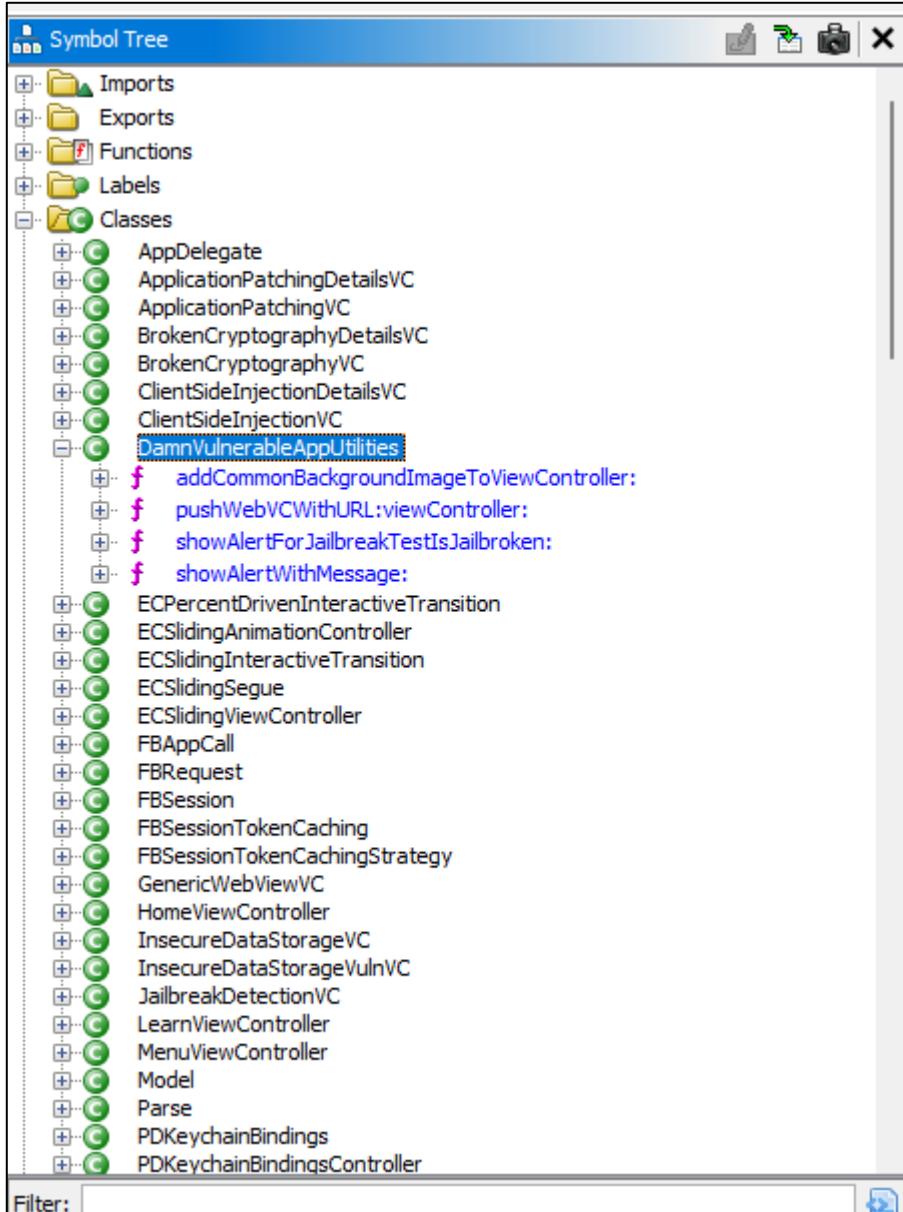
```
function handleMessage(message) {
    modules = getAllAppModules();
    var app_path = ObjC.classes.NSBundle mainBundle().bundlePath();
    loadAllDynamicLibrary(app_path);
    // start dump
    modules = getAllAppModules();
    for (var i = 0; i < modules.length; i++) {
        if (modules[i].path.endsWith("/SharedFrameworks/libloader.dylib")) {
            continue;
        }
        console.log("start dump " + modules[i].path);
        var result = dumpModule(modules[i].path);
        send({ dump: result, path: modules[i].path });
    }
    send({app: app_path.toString()});
    send({done: "ok"});
    recv(handleMessage);
}
```

Using
Chidra

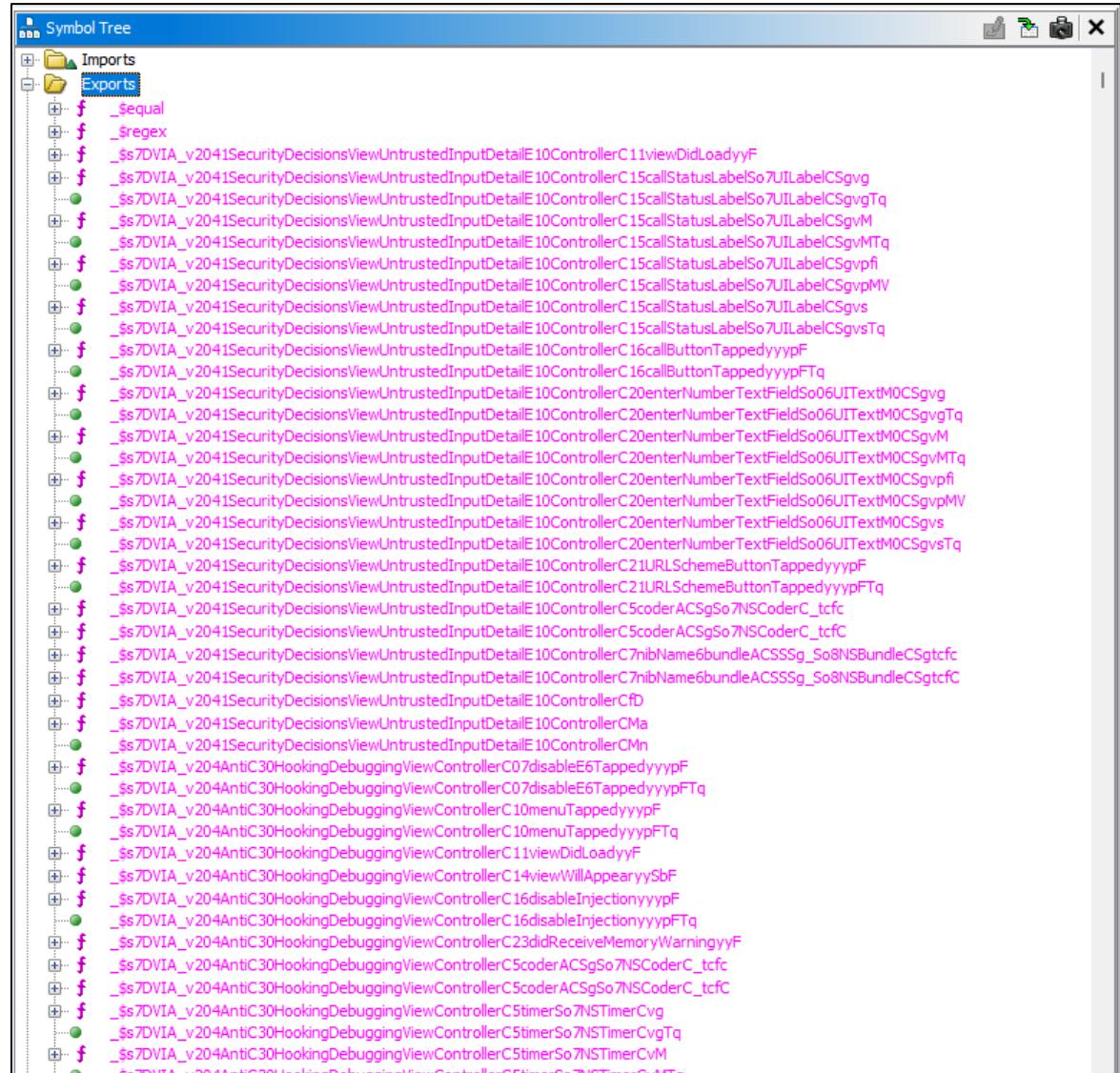
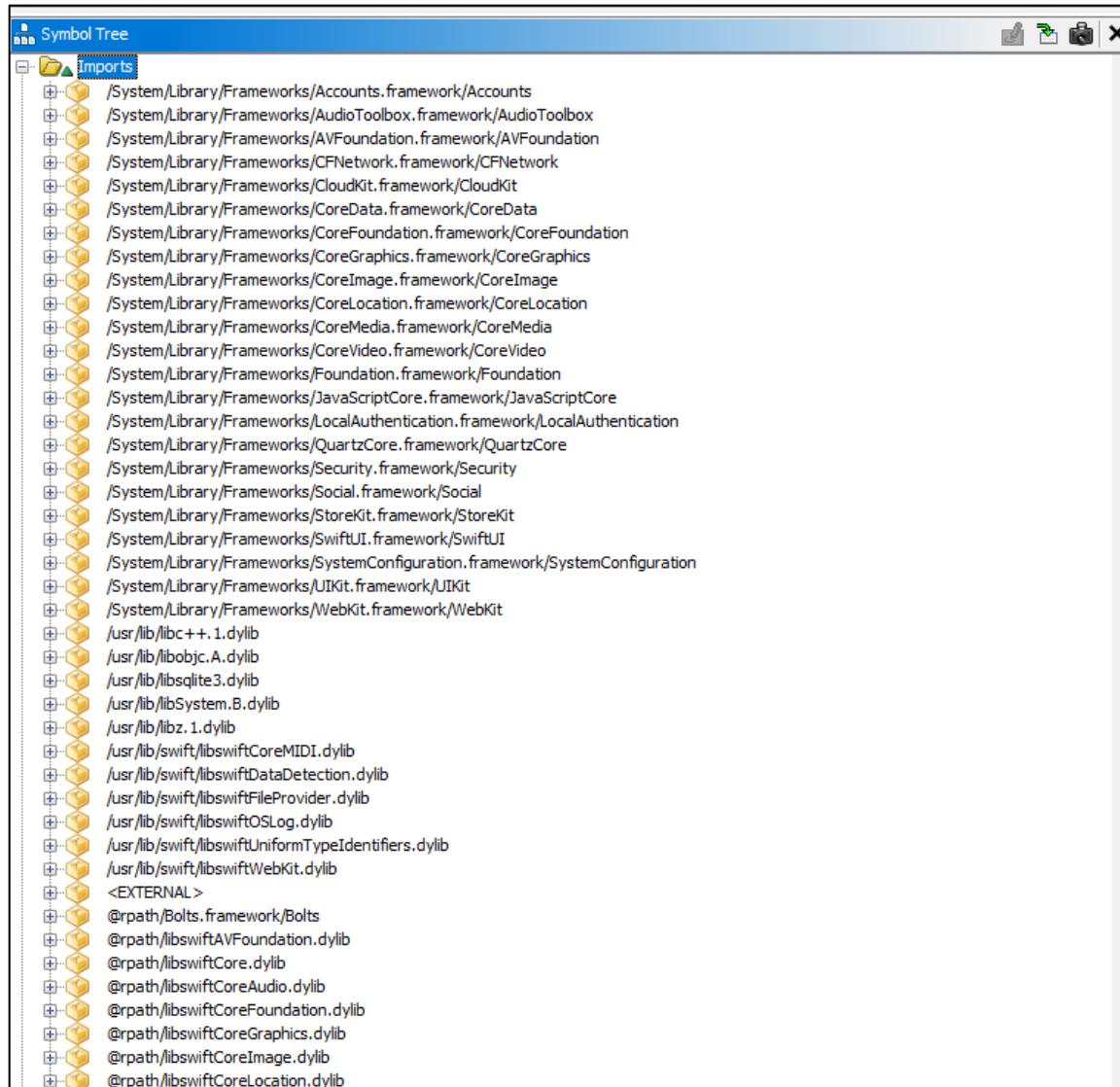
Navigating UI



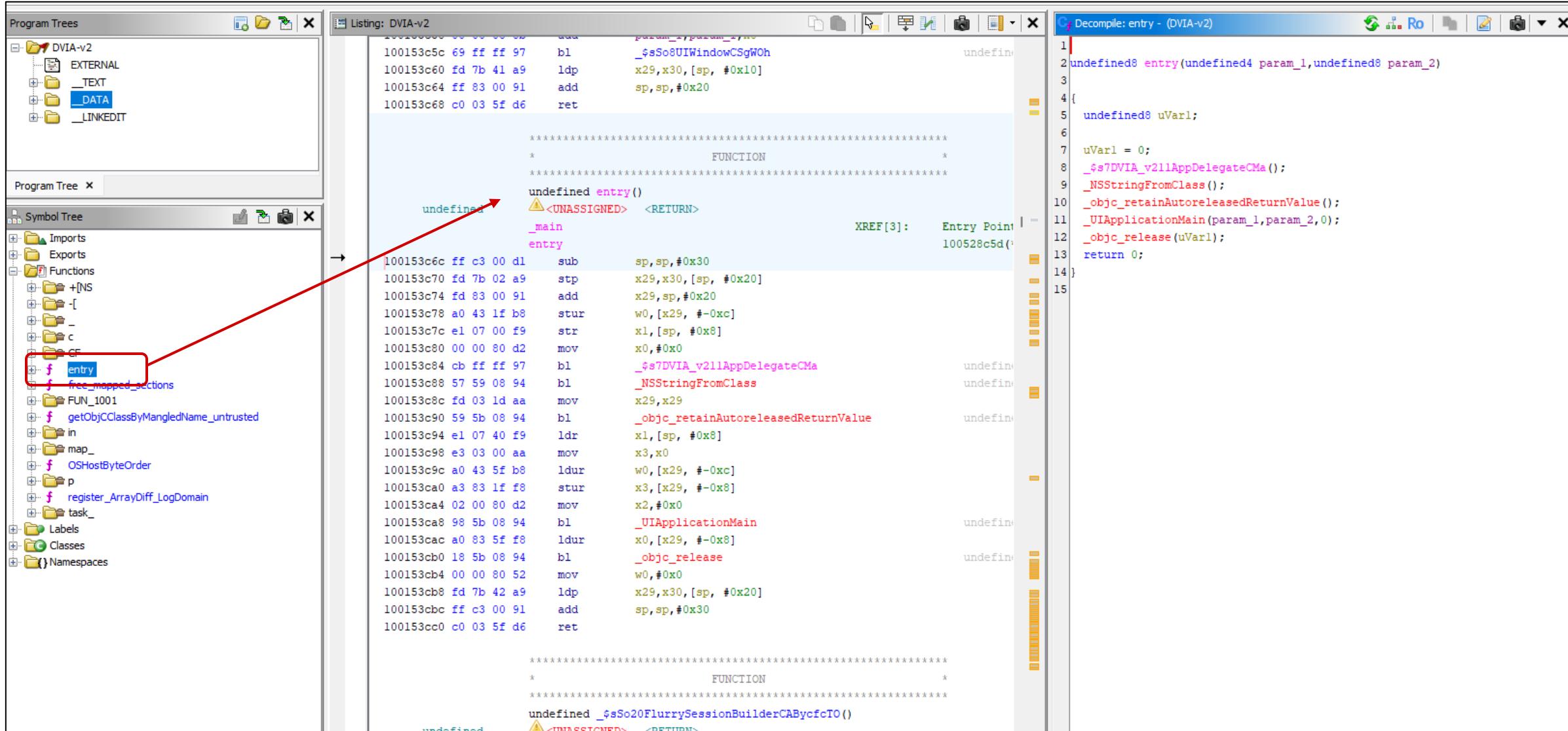
Decompiled Method and Function Names



Imports & Exports



Entry Function



The screenshot shows the Ghidra interface with three main panes:

- Program Trees** pane on the left, showing the file structure of DVIA-v2 with sections like EXTERNAL, _TEXT, DATA, and _LINKEDIT.
- Listing** pane in the center, displaying assembly code for the entry function. A red arrow points from the highlighted 'entry' symbol in the Symbol Tree to the entry point in the Listing pane.
- Decompile** pane on the right, showing the C-like pseudocode for the entry function.

Symbol Tree pane details:

- Shows various symbols including imports, exports, and functions.
- A symbol named **entry** is highlighted with a red box and has a red arrow pointing to it from the Listing pane.

Decompile pane (entry - DVIA-v2):

```
1 undefined8 entry(undefined4 param_1,undefined8 param_2)
2 {
3     undefined8 uVar1;
4
5     uVar1 = 0;
6     _$s7DVIA_v211AppDelegateCMa();
7     _NSStringFromClass();
8     _objc_retainAutoreleasedReturnValue();
9     _UIApplicationMain(param_1,param_2,0);
10    _objc_release(uVar1);
11    return 0;
12 }
13
14 }
```

Generating Pseudo Codes

The screenshot displays the Ghidra interface with three main panes:

- Program Trees** (Left): Shows the project structure with various files like Dvia-v2, Dvia-v3, and Dvia-v4. A red arrow points from the 'Symbol Tree' section of this pane towards the 'Decompile' pane.
- Listing** (Middle): Displays assembly code for the current function. The assembly code is color-coded with blue for labels, green for comments, and purple for instructions. A red arrow points from the 'Symbol Tree' section of the 'Program Trees' pane towards this listing.
- Decompile** (Right): Shows the generated pseudo code for the function. The pseudo code is written in C-like syntax, using underscores for variable names and symbols. It includes imports for Foundation and Swift libraries.

Symbol Tree (Program Trees)

```

Dvia-v2
  - EXTERNAL
  - TEXT
  - DATA
  - LINKEDIT

  - Functions
    - $s7Dvia_v213DviaUtilitiesC9showAlert28forJailbreakTestIsJailbroken
    - $s7Dvia_v213DviaUtilitiesC9showAlert28forJailbreakTestIsJailbroken
    - $s7Dvia_v232JailbreakDetectionViewControllerC12isJailbrokenSbyF
    - $s7Dvia_v232JailbreakDetectionViewControllerC12isJailbrokenSbyFTq
    - $s7Dvia_v213DviaUtilitiesC9showAlert28forJa
    - $s7Dvia_v21
    - $s7Dvia_v213
    - $s7Dvia_v23
    - $s7Dvia_v232
    - $s7Dvia_v232JailbreakDetectionViewControllerC1
  - Labels
  - Classes
    - F
      - Flurry
        - FlurryU
          - FlurryUtil
            - f_deviceIsJailbroken
    - J
      - JailbreakDetection
        - f_isJailbroken
  - Filter: isJailbroken
  - Data Type Manager

```

Listing: Dvia-v2

```

100141e14 c0 03 5f d6 ret
100141e18 1f 00 01 91 add
100141e1c 00 00 00 00 add
100141e20 fd 7b 02 a9 stp x22,x21,[sp, #-0x30]!
100141e24 fd 83 00 91 add x29,sp,#0x20
100141e28 ff 03 08 d1 sub sp,sp,#0x200
100141e2c 28 12 00 b0 adrp x8,0x1000386000
100141e30 08 b1 39 91 add x8,x8,#0xe6c
100141e34 a8 83 11 f8 stur x8=>s_Fatal_error_100386e6c,[x29, #-0xe8]
100141e38 08 12 00 f0 adrp x8,0x1000384000
100141e3c 08 c1 0d 91 add x8,x8,#0x370
100141e40 a8 03 12 f8 stur x8=>s_Unexpectedly_found_nil_while_unwinding
100141e44 08 12 00 f0 adrp x8,0x1000384000
100141e48 08 c1 29 91 add x8,x8,#0xa70
100141e4c a8 83 12 f8 stur x8=>s_Dvia_v2/JailbreakDetectionViewControllerC12isJailbrokenSbyF
100141e50 88 16 00 f0 adrp x8,0x100414000
100141e54 08 9d 42 f9 ldr x8,[x8, #0x538]=>__stack_chk_guard
100141e58 08 01 40 f9 ldr x8,[x8]=>__stack_chk_guard
100141e5c a8 83 1d f8 stur x8,[x29, #-0x20]
100141e60 bf 83 1b f8 stur x8,[x29, #-0x48]
100141e64 bf 83 1a f8 stur x8,[x29, #-0x58]
100141e68 bf 03 1b f8 stur x8,[x29, #-0x50]
100141e6c 00 00 80 d2 mov x0,#0x0
100141e70 a0 03 13 f8 stur x0,[x29, #-0xd0]
100141e74 bf 03 18 f8 stur x8,[x29, #-0x80]
100141e78 60 1d 00 90 adrp x0,0x1004ed000
100141e7c 00 00 23 91 add x0=>s10Foundation3URLVSGMD,x0,#0x8c0
100141e80 7b eb ff 97 bl __swift_instantiateConcreteTypeFromMangledName
100141e84 08 80 5f f8 ldur x8,[x0, #-0x10]
100141e88 08 21 40 f9 ldr x8,[x8, #0x40]
100141e8c 08 fd 40 d3 lsr x8,x8,#0x0
100141e90 08 3d 00 92 add x8,x8,#0xf
100141e94 09 ed 7c 92 and x9,x8,#-0x10
100141e98 e8 03 00 91 mov x8,sp
100141ea0 00 01 09 eb subs x0,x8,x9
100141ea4 0f 00 00 91 mov x0,[x29, #-0xc0]
100141ea8 00 00 80 d2 mov x0,#0x0
100141eb0 82 a1 08 94 bl __$s10FoundationE8EncodingVMA
100141eb4 08 03 14 f8 stur x0,[x29, #-0xc0]
100141eb8 08 80 5f f8 ldur x8,[x0, #-0x8]
100141ebc a8 83 14 f8 stur x8,[x29, #-0xb8]
100141ebc 08 21 40 f9 ldr x8,[x8, #0x40]
100141ec0 0f fd 40 d3 lsr x8,x8,#0x0
100141ecd 08 3d 00 91 add x8,x8,#0xf
100141ec8 09 ed 7c 92 and x9,x8,#-0x10
100141ec0 e8 03 00 91 mov x8,sp
100141ed0 00 01 09 eb subs x0,x8,x9
100141ed4 a0 03 15 f8 stur x0,[x29, #-0xb0]
100141ed8 1f 00 00 91 mov sp,x0

```

Decompile: __\$s7Dvia_v232JailbreakDetectionViewControllerC12isJailbrokenSbyF

```

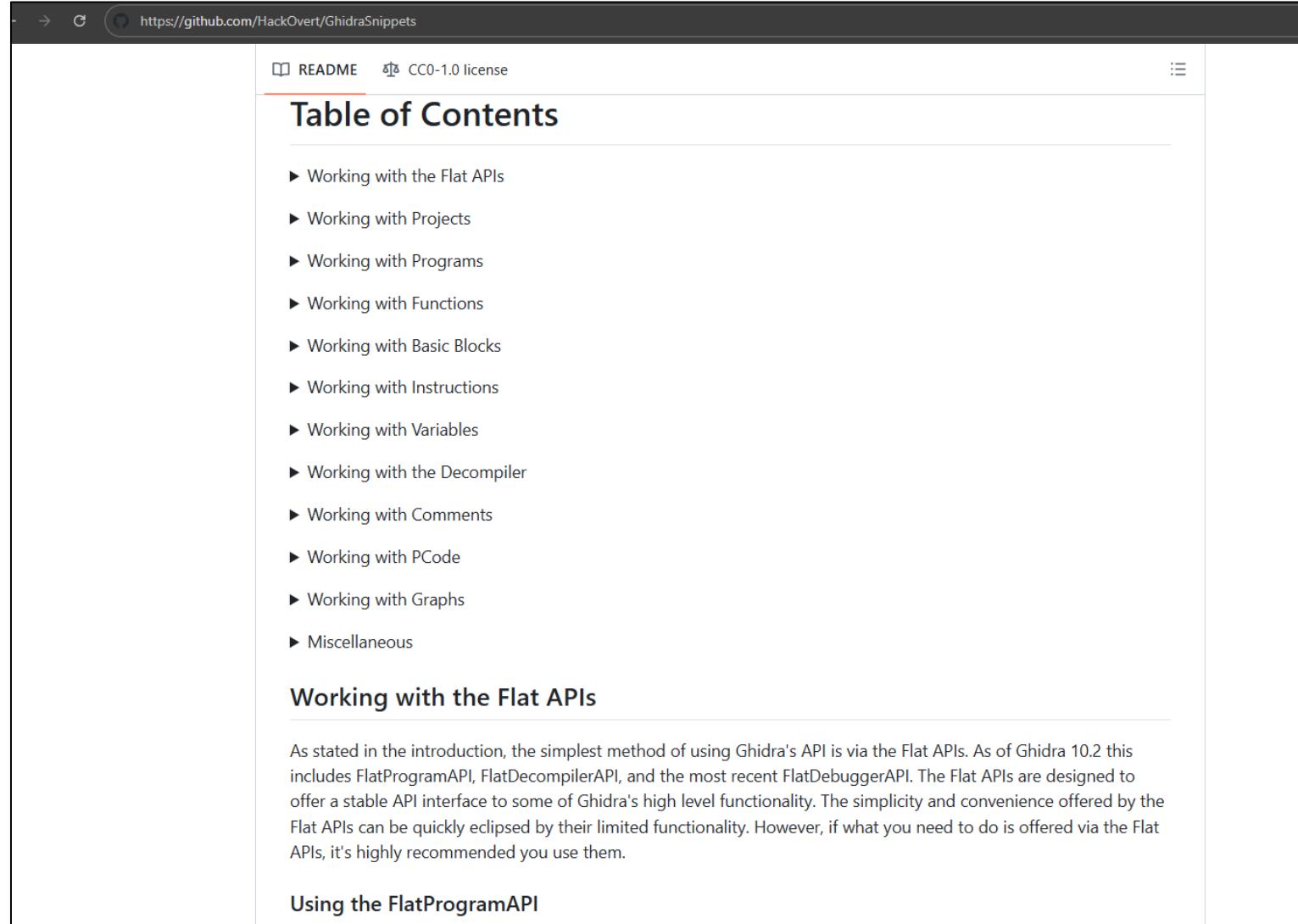
82 long lStack_40;
83 long lStack_38;

84 pcStack_f8 = "Fatal error";
85 pcStack_f0 = "Unexpectedly found nil while unwrapping an Optional value";
86 pcStack_e8 = "Dvia_v2/JailbreakDetectionViewController.swift";
87 lStack_38 = *(long *)PTR__stack_chk_guard_100414538;
88 pcStack_68 = (char *)0x0;
89 uStack_60 = 0;
90 lStack_e0 = 0;
91 lStack_90 = 0;
92 puVar5 = s10Foundation3URLVSGMD;
93 __swift_instantiateConcreteTypeFromMangledName();
94 lVar13 = *(long *)*(long *)(puVar5 + -8) + 0x40;
95 lVar4 = 0;
96 puStack_d8 = auStack_230 + -(lVar13 + 0xFU & 0xfffffffffffff0);
97 __$s10FoundationE8EncodingVMA();
98 lStack_c8 = *(long *)lVar4 + -8;
99 lVar13 = (long)(auStack_230 + -(lVar13 + 0xFU & 0xfffffffffffff0)) - (* (long *) (lStack_c8 + 0x40) + 0xFU & 0xfffffffffffff0));
100 puVar5 =_OBJC_CLASS_$_NSFileManager;
101 lStack_d0 = lVar4;
102 lStack_c0 = lVar13;
103 __swift_getInitializedObjCClass();
104 _objc_msgSend();
105 _objc_retainAutoreleasedReturnValue();
106 pcVar11 = "/Applications/Cydia.app";
107 uVar9 = 0x17;
108 puStack_a8 = puVar5;
109 __$s21_builtinStringLiteral17utf8CodeUnitCount7isASCIISSBp_BwBil_tcfC
110 ("Applications/Cydia.app",0x17,1);
111 uStack_b8 = uVar9;
112 __$s10FoundationE19_bridgeToObjectiveCSo8NSStringCyF();
113 pcStack_b0 = pcVar11;
114 __swift_bridgeObjectRelease(uStack_b8);
115 puVar5 = puStack_a8;
116 _objc_msgSend(puStack_a8,"fileExistsAtPath:",pcStack_b0);
117 uStack_9c = (uint)puVar5;
118 _objc_release(pcStack_b0);
119 _objc_release(puStack_a8);
120 if ((uStack_9c & 1) == 0) {
121 puVar5 = _OBJC_CLASS_$_NSFileManager;
122 __swift_getInitializedObjCClass();
123 _objc_msgSend();
124 _objc_retainAutoreleasedReturnValue();
125 pcVar11 = "/Library/MobileSubstrate/MobileSubstrate.dylib";
126 uVar9 = 0x2e;
127 puStack_108 = puVar5;
128 __$s21_builtinStringLiteral17utf8CodeUnitCount7isASCIISSBp_BwBil_tcfC
129 ("Library/MobileSubstrate/MobileSubstrate.dylib",0x2e,1);
130 uStack_118 = uVar9;
131 __$s10FoundationE19_bridgeToObjectiveCSo8NSStringCyF();
132 pcStack_110 = pcVar11;
133 __swift_bridgeObjectRelease(uStack_118);
134 puVar5 = puStack_108;
135 _objc_msgSend(puStack_108,"fileExistsAtPath:",pcStack_110);
136 uStack_100 = (uint)puVar5;
137 ...

```

Writing Ghidra Scripts

<https://github.com/HackOvert/GhidraSnippets>



The screenshot shows a GitHub repository page for 'GhidraSnippets'. At the top, there are links for 'README' and 'CC0-1.0 license'. Below that is a 'Table of Contents' section with the following items:

- ▶ Working with the Flat APIs
- ▶ Working with Projects
- ▶ Working with Programs
- ▶ Working with Functions
- ▶ Working with Basic Blocks
- ▶ Working with Instructions
- ▶ Working with Variables
- ▶ Working with the Decompiler
- ▶ Working with Comments
- ▶ Working with PCode
- ▶ Working with Graphs
- ▶ Miscellaneous

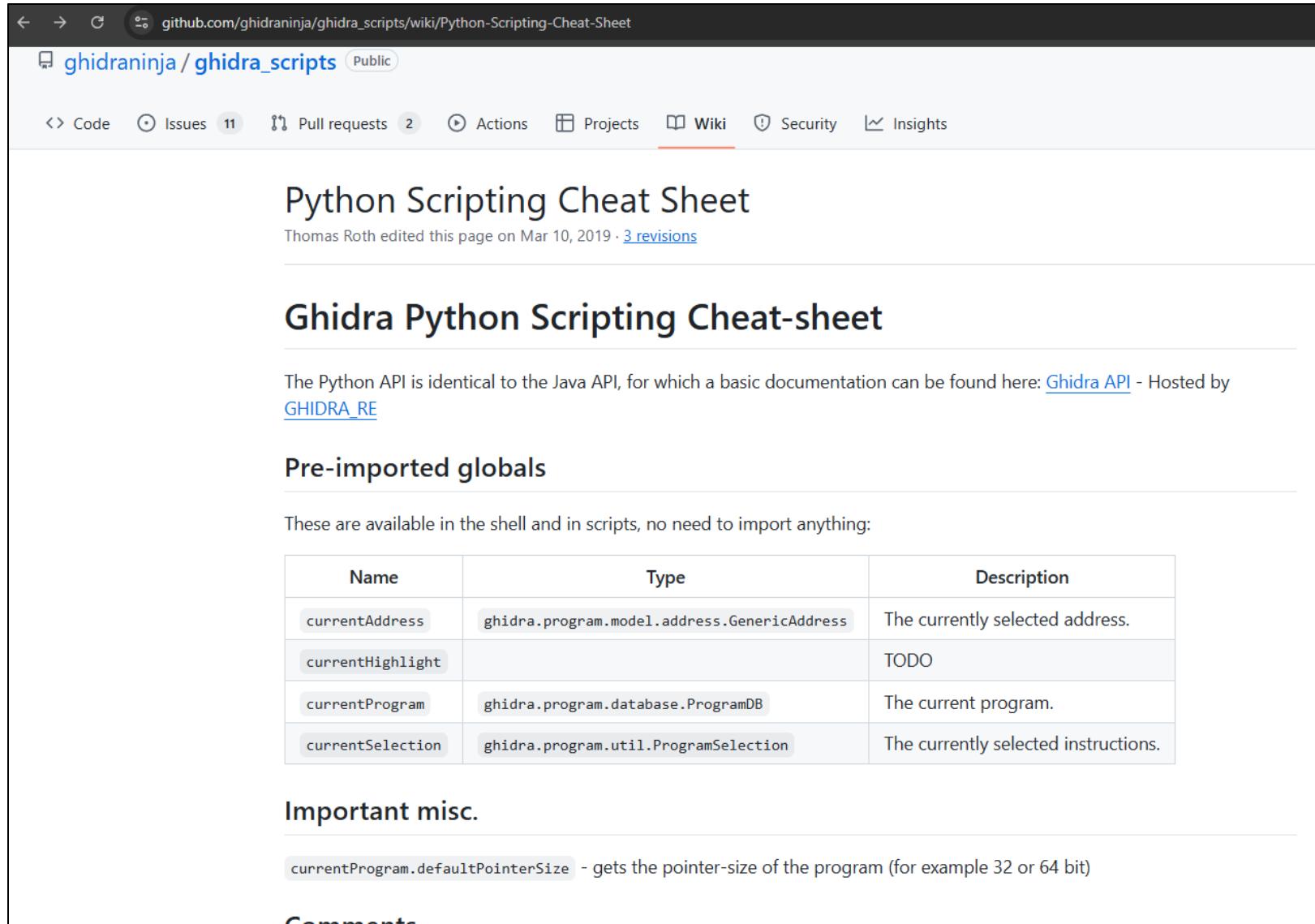
Below the table of contents is a section titled 'Working with the Flat APIs' with the following text:

As stated in the introduction, the simplest method of using Ghidra's API is via the Flat APIs. As of Ghidra 10.2 this includes FlatProgramAPI, FlatDecompilerAPI, and the most recent FlatDebuggerAPI. The Flat APIs are designed to offer a stable API interface to some of Ghidra's high level functionality. The simplicity and convenience offered by the Flat APIs can be quickly eclipsed by their limited functionality. However, if what you need to do is offered via the Flat APIs, it's highly recommended you use them.

At the bottom of the page, there is a link to 'Using the FlatProgramAPI'.

Writing Ghidra Scripts

https://github.com/ghidraninja/ghidra_scripts/wiki/Python-Scripting-Cheat-Sheet



The screenshot shows a GitHub wiki page titled "Python Scripting Cheat Sheet" under the "ghidraninja / ghidra_scripts" repository. The page has a "Public" status and a "Wiki" tab selected. It includes sections on "Ghidra Python Scripting Cheat-sheet", "Pre-imported globals", and "Important misc.". A table lists pre-imported globals with their types and descriptions. A note at the bottom explains how to get the pointer-size of the program.

Python Scripting Cheat Sheet

Thomas Roth edited this page on Mar 10, 2019 · [3 revisions](#)

Ghidra Python Scripting Cheat-sheet

The Python API is identical to the Java API, for which a basic documentation can be found here: [Ghidra API](#) - Hosted by [GHIDRA RE](#)

Pre-imported globals

These are available in the shell and in scripts, no need to import anything:

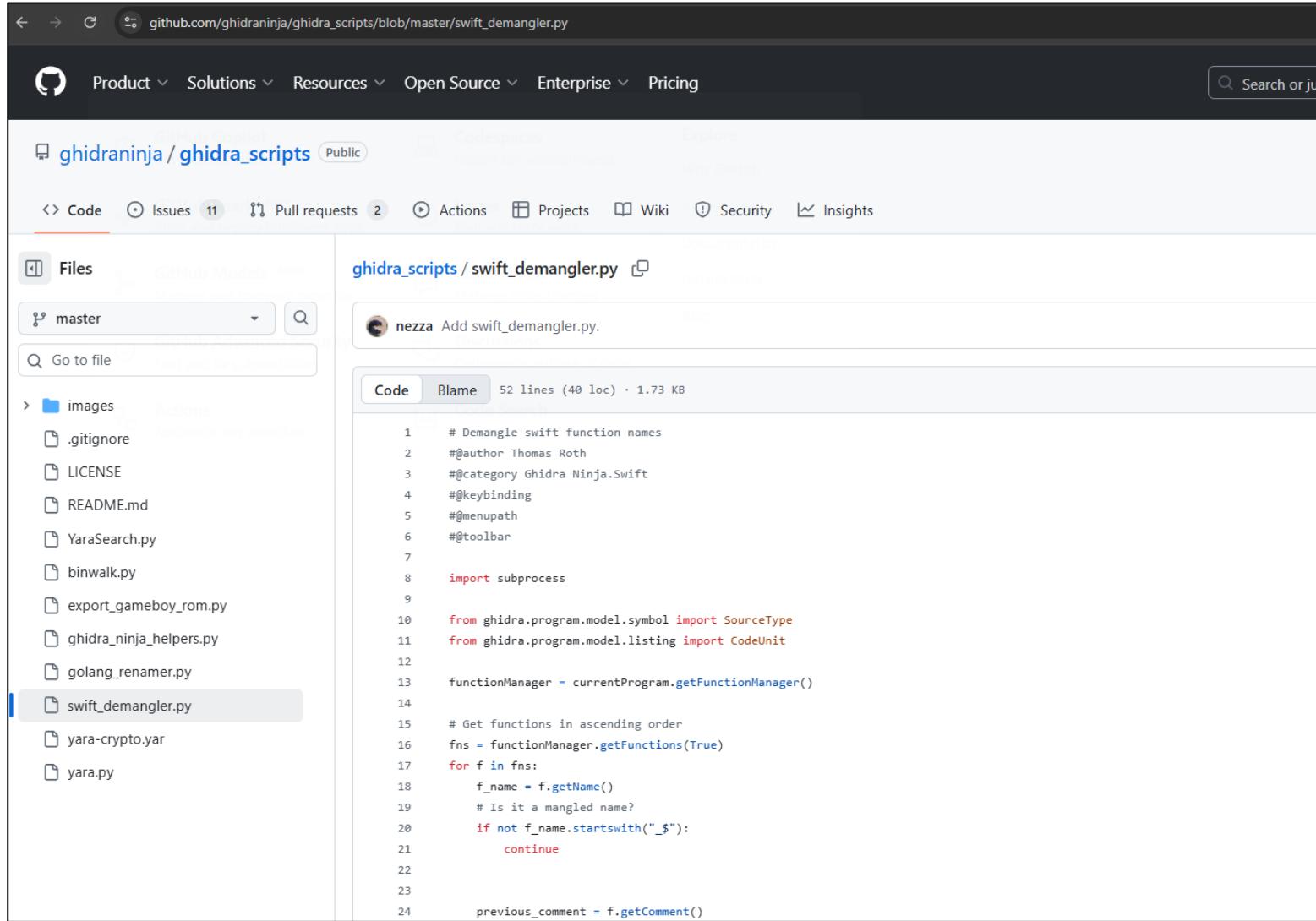
Name	Type	Description
<code>currentAddress</code>	<code>ghidra.program.model.address.GenericAddress</code>	The currently selected address.
<code>currentHighlight</code>		TODO
<code>currentProgram</code>	<code>ghidra.program.database.ProgramDB</code>	The current program.
<code>currentSelection</code>	<code>ghidra.program.util.ProgramSelection</code>	The currently selected instructions.

Important misc.

`currentProgram.defaultPointerSize` - gets the pointer-size of the program (for example 32 or 64 bit)

Writing Ghidra Scripts

https://github.com/ghidraninja/ghidra_scripts/blob/master/swift_demangler.py



The screenshot shows a GitHub repository page for 'ghidra_scripts'. The repository has 11 issues and 2 pull requests. The 'Code' tab is selected, displaying the contents of the 'swift_demangler.py' file. The code is a Python script for demangling Swift function names. It imports subprocess, SourceType, and CodeUnit from the Ghidra program model. It retrieves functions in ascending order and checks if their names start with an underscore. If they do, it continues to the next function. Otherwise, it gets the comment for the previous function.

```
# Demangle swift function names
#@author Thomas Roth
#@category Ghidra Ninja.Swift
#@keybinding
#@menupath
#@toolbar

import subprocess

from ghidra.program.model.symbol import SourceType
from ghidra.program.model.listing import CodeUnit

functionManager = currentProgram.getFunctionManager()

# Get functions in ascending order
fns = functionManager.getFunctions(True)
for f in fns:
    f_name = f.getName()
    # Is it a mangled name?
    if not f_name.startswith("_$"):
        continue

    previous_comment = f.getComment()
```

Ghidra Script Walkthrough

Pre-imported global(s)



```
#@author Thomas Roth
[... Snipped Comments ...]
```

Get all Functions



```
import subprocess

from ghidra.program.model.symbol import SourceType
from ghidra.program.model.listing import CodeUnit
```

Look for Swift Functions



```
functionManager = currentProgram.getFunctionManager()

# Get functions in ascending order
fns = functionManager.getFunctions(True)
for f in fns:

```

Run separate process to demangle Swift Functions



```
previous_comment = f.getComment()
if not previous_comment:
    previous_comment = ""

signature_full = subprocess.check_output(["swift", "demangle", "-compact", f_name])[:-1]
signature = signature_full

# [... Snipped Comments ...]
func_description = signature.split("(")[0]
if func_description.count(" ") > 1:
    signature = subprocess.check_output(["swift", "demangle", "-compact", "-simplified", f_name])[:-1]

# Replace characters we can't have in our name
signature = signature.split("(")[0]
signature = signature.replace(" ", "_")
```

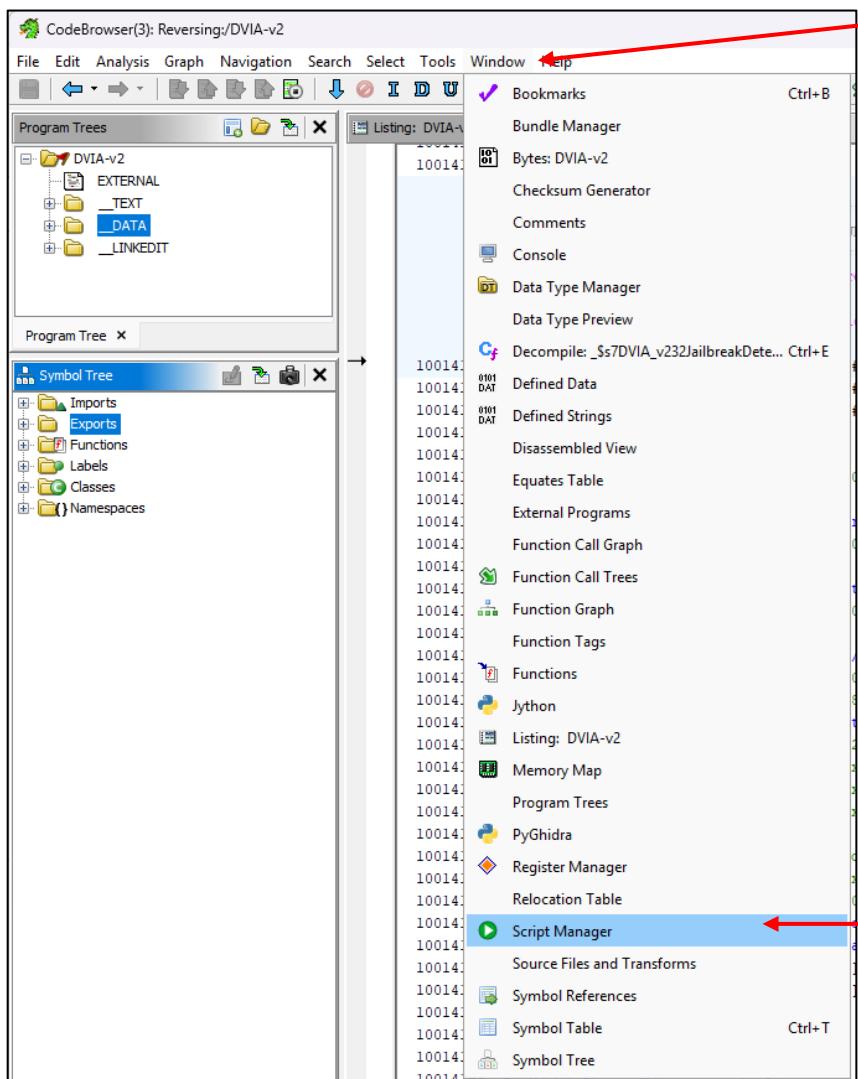
Update functions' name and comments



```
# Add newlines into full comment (maximum comment len = 58, afterwards truncated)
lines = len(signature_full) / 58
for l in range(1, lines+1):
    signature_full = signature_full[:l*58+(l-1)] + "\n" + signature_full[l*58+(l-1):]

f.setComment(previous_comment + "\n" + signature_full)
f.setName(signature, SourceType.ANALYSIS)
```

Execute Ghidra Scripts

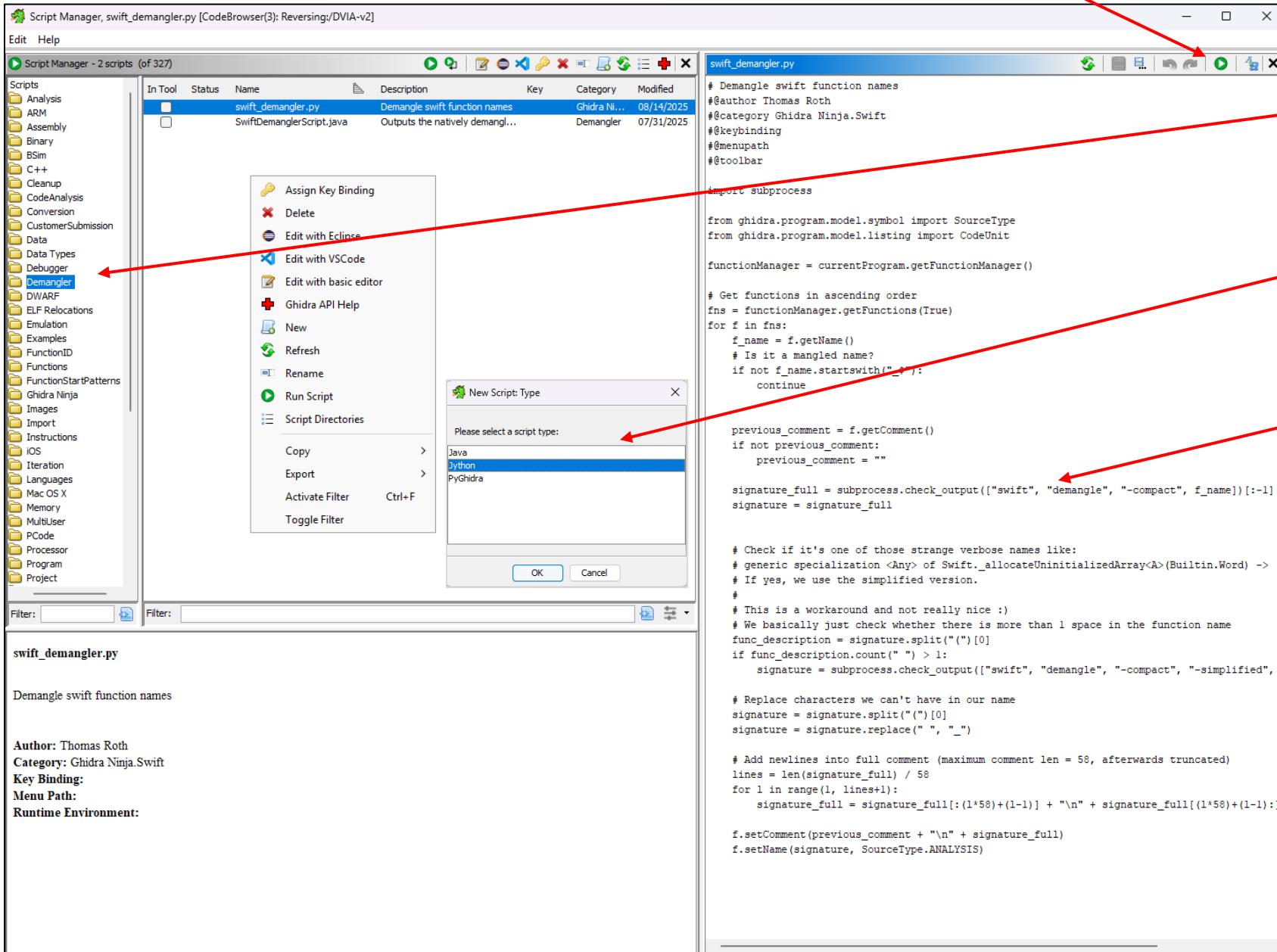


Window's

Script Manager

Execute Ghidra Scripts

Run script.



On the left select a folder which best matches the script's use case
Right click and select new to create a new script.

Copy and paste the script into the built in editor on the right.

Select the script's type (Java, Jython or PyGhidra).

Reversing

Strategy

Static Analysis

Static analysis involves analyzing the software's code without executing the program.



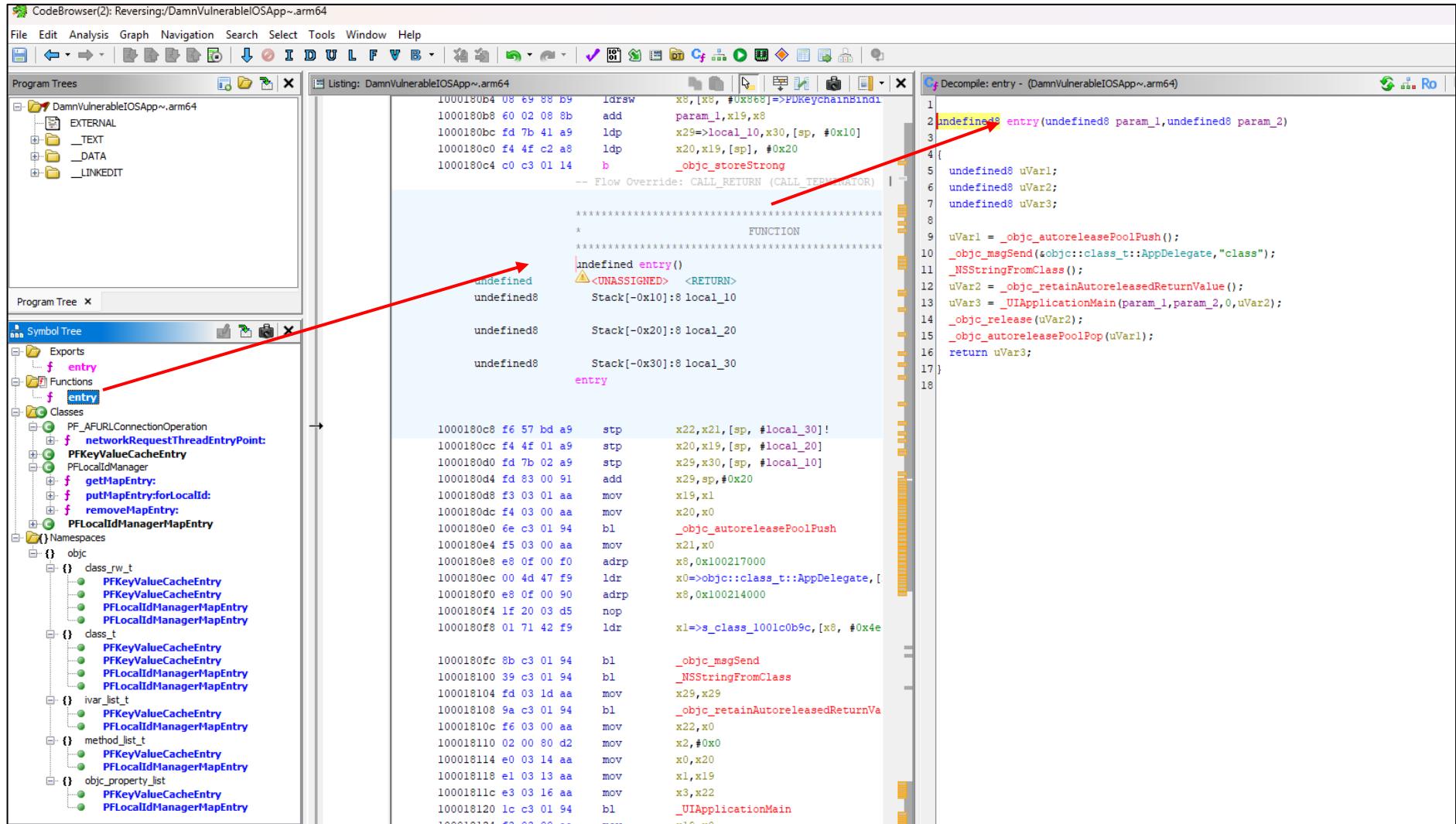
Dynamic Analysis

Dynamic analysis is a technique that analyzes the software while executing the program.



Static Analysis

Entry() Function



The screenshot shows the CodeBrowser interface with three main panes:

- Program Trees:** Shows the file structure of the DamnVulnerableIOSApp~.arm64 project.
- Listing:** Displays the assembly code for the entry point. The assembly code includes instructions like `ldrsw`, `add`, `ldp`, and `b`, along with comments indicating flow overrides and function boundaries.
- Decompile:** Shows the decompiled C-like pseudocode for the entry function. The code involves Objective-C runtime calls such as `_objc_msgSend`, `_NSStringFromClass`, and `_UIApplicationMain`.

```
CodeBrowser(2): Reversing/DamnVulnerableIOSApp~.arm64
File Edit Analysis Graph Navigation Search Select Tools Window Help
Program Trees Listing: DamnVulnerableIOSApp~.arm64 Decompile: entry - (DamnVulnerableIOSApp~.arm64)
EXTERNAL TEXT DATA LINKEDIT
damnVulnerableIOSApp~.arm64
undefined8 entry(undefined8 param_1,undefined8 param_2)
{
    undefined8 uVar1;
    undefined8 uVar2;
    undefined8 uVar3;
    uVar1 = _objc_autoreleasePoolPush();
    _objc_msgSend(&objc::class_t::AppDelegate,"class");
    _NSStringFromClass();
    uVar2 = _objc_retainAutoreleasedReturnValue();
    uVar3 = _UIApplicationMain(param_1,param_2,0,uVar2);
    _objc_release(uVar2);
    _objc_autoreleasePoolPop(uVar1);
    return uVar3;
}
FUNCTION
undefined entry()
<UNDEFINED> <RETURN>
Stack[-0x10]:8 local_10
Stack[-0x20]:8 local_20
Stack[-0x30]:8 local_30
entry

1000180b4 08 69 88 b9  ldrsw  x8,[x8, #0x868] >=PFKeychainBindi
1000180b8 60 02 08 b8  add   param_1,x19,x8
1000180bc fd 7b 41 a9  ldp   x29=>local_10,x30,[sp, #0x10]
1000180c0 f4 4f c2 a8  ldp   x20,x19,[sp], #0x20
1000180c4 c0 c3 01 14  b    _objc_storeStrong
-- Flow Override: CALL_RETURN (CALL TERMINATOR)

1000180c8 f6 57 bd a9  stp   x22,x21,[sp, #local_30]!
1000180cc f4 4f 01 a9  stp   x20,x19,[sp, #local_20]
1000180d0 fd 7b 02 a9  stp   x29,x30,[sp, #local_10]
1000180d4 fd 83 00 91  add   x29,sp, #0x20
1000180d8 f3 03 01 aa  mov   x19,x1
1000180dc f4 03 00 aa  mov   x20,x0
1000180e0 6e c3 01 94  bl    _objc_autoreleasePoolPush
1000180e4 f5 03 00 aa  mov   x21,x0
1000180e8 e8 0f 00 f0  adrp  x8,0x100217000
1000180ec 00 4d 47 f9  ldr   x0=>objc::class_t::AppDelegate,[x8, #0x100214000]
1000180f0 e8 0f 00 f0  adrp  x8,0x100214000
1000180f4 lf 20 03 d5  nop
1000180f8 01 71 42 f9  ldr   x1=>s_class_1001c0b9c,[x8, #0x4e
1000180fc 8b c3 01 94  bl    _objc_msgSend
100018100 39 c3 01 94  bl    _NSStringFromClass
100018104 fd 03 1d aa  mov   x29,x29
100018108 9a c3 01 94  bl    _objc_retainAutoreleasedReturnValue
10001810c fd 03 00 aa  mov   x22,x0
100018110 02 00 80 d2  mov   x2,#0x0
100018114 e0 03 14 aa  mov   x0,x20
100018118 e1 03 13 aa  mov   x1,x19
10001811c e3 03 16 aa  mov   x3,x22
100018120 1c c3 01 94  bl    _UIApplicationMain
100018124 #3 03 00 aa  mov   w19,w0
```

Static Analysis

Display Function Graph

The image shows a static analysis environment with three main windows:

- Decompiler View (Left):** Shows assembly code for a function named "entry - (Sample)". A red box highlights the "Function Graph" icon in the toolbar.
- Sync View (Center):** A red arrow labeled "Sync View" points from the decompiler view to the function graph. It displays the assembly code for the function, with some instructions highlighted in yellow.
- Function Graph View (Right):** Shows the control flow graph (CFG) for the function. Nodes represent assembly labels like 1018771fc, LAB_10187721c, and 10187720c. Edges represent jumps between nodes. A yellow box highlights a specific node, and a green box highlights another node. A message at the bottom right says: "Location is not in a defined or undefined function '105a561f8'".

```

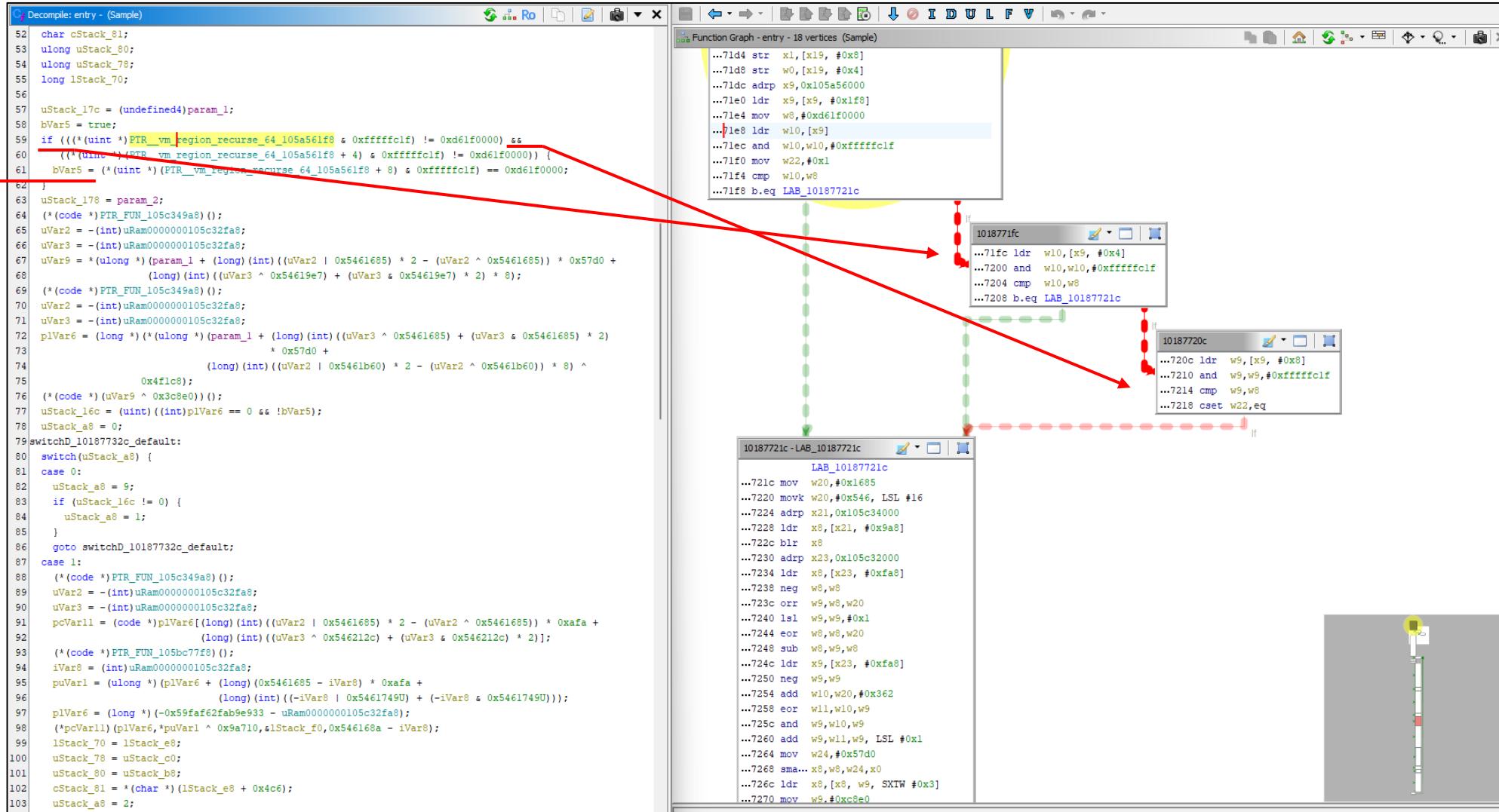
52 char cStack_81;
53 ulong uStack_80;
54 ulong uStack_78;
55 long lStack_70;
56
57 uStack_17c = (undefined4)param_1;
58 bVar5 = true;
59 if (((uint *)PTR__vm_region_recurse_64_105a561f8 & 0xfffffc1f) != 0xd61f0000) &&
60 (((uint *)PTR__vm_region_recurse_64_105a561f8 + 4) & 0xfffffc1f) != 0xd61f0000) {
61 bVar5 = (*uint *)PTR__vm_region_recurse_64_105a561f8 + 8) & 0xfffffc1f == 0xd61f0000;
62 }
63 uStack_178 = param_2;
64 (*code *)PTR_DAT_105c349a8();
65 uVar2 = -(int)uRam00000000105c32fa8;
66 uVar3 = -(int)uRam00000000105c32fa8;
67 uVar9 = *(ulong *)param_1 + (long)(int)((uVar2 | 0x5461685) * 2 - (uVar2 ^ 0x5461685)) * 0x57d0 +
68 (long)(int)((uVar3 ^ 0x54619e7) + (uVar3 & 0x54619e7) * 2) * 8);
69 (*code *)PTR_DAT_105c349a8();
70 uVar2 = -(int)uRam00000000105c32fa8;
71 uVar3 = -(int)uRam00000000105c32fa8;
72 plVar6 = (long *)*(ulong *)param_1 + (long)(int)((uVar3 ^ 0x5461685) + (uVar3 & 0x5461685) * 2)
73 * 0x57d0 +
74 (long)(int)((uVar2 | 0x5461b60) * 2 - (uVar2 ^ 0x5461b60)) * 8) ^
75 0x4fc1c8;
76 (*code *) (uVar9 ^ 0x3c8e0));
77 uStack_16c = (uint)((int)plVar6 == 0 && !bVar5);
78 uStack_a8 = 0;
79 LAB_101877314:
80 switch(uStack_a8) {
81 case 0:
82 uStack_a8 = 9;
83 if (uStack_16c != 0) {
84 uStack_a8 = 1;
85 }
86 goto LAB_101877314;
87 case 1:
88 (*code *)PTR_DAT_105c349a8();
89 uVar2 = -(int)uRam00000000105c32fa8;
90 uVar3 = -(int)uRam00000000105c32fa8;
91 pcVar11 = (code *)plVar6[(long)(int)((uVar2 | 0x5461685) * 2 - (uVar2 ^ 0x5461685)) * 0xfa +
92 (long)(int)((uVar3 ^ 0x546212c) + (uVar3 & 0x546212c) * 2)];
93 (*code *)PTR_DAT_105bc77f8();
94 iVar8 = (int)uRam0000000105c32fa8;
95 plVar1 = (ulong *) (plVar6 + (long)(0x5461685 - iVar8) * 0xfa +
96 (long)(int)((-iVar8 | 0x5461749U) + (-iVar8 & 0x5461749U)));
97 plVar6 = (long *)(-0x59faf62fab9e933 - uRam0000000105c32fa8);
98 (*pcVar11)(plVar6,*puVar1 ^ 0x9a710,&lStack_f0,0x546168a - iVar8);
99 lStack_70 = lStack_e8;
100 uStack_78 = uStack_c0;
101 uStack_80 = uStack_b8;
102 cStack_81 = *(char *) (lStack_e8 + 0x4c6);

```

Static Analysis

Interpreting the pseudocode alongside the execution flow graph to understand the program's behavior.

Boolean
bVar5



The screenshot displays two windows from a static analysis tool. On the left is a decompiler window titled 'Decompiled entry - (Sample)' showing C-like pseudocode. On the right is a 'Function Graph' window titled 'Function Graph - entry - 18 vertices (Sample)' showing a flow graph with nodes and assembly code. A red arrow points from the 'bVar5' variable in the pseudocode to its corresponding node in the flow graph. Another red arrow points from the flow graph node to the assembly code for that node.

```

52 char cStack_81;
53 ulong uStack_80;
54 ulong uStack_78;
55 long lStack_70;
56
57 uStack_17c = (*undefined4)param_1;
58 bVar5 = true;
59 if (((*uint *)PTR__vm_region_recurse_64_105a561f8 & 0xfffffcff) != 0xd61f0000) &&
60 (((*uint *)PTR__vm_region_recurse_64_105a561f8 + 4) & 0xfffffcff) != 0xd61f0000)) {
61     bVar5 = ((*uint *)PTR__vm_region_recurse_64_105a561f8 + 8) & 0xfffffcff == 0xd61f0000;
62 }
63 uStack_178 = param_2;
64 (*code *)PTR_FUN_105c349a8();
65 uVar2 = -(int)uRam0000000105c32fa8;
66 uVar3 = -(int)uRam0000000105c32fa8;
67 uVar9 = *(ulong *)param_1 + (long)(int)((uVar2 | 0x5461685) * 2 - (uVar2 ^ 0x5461685)) * 0x57d0 +
68 (long)(int)((uVar3 ^ 0x54619e7) + (uVar3 & 0x54619e7) * 2) * 8);
69 (*code *)PTR_FUN_105c349a8();
70 uVar2 = -(int)uRam0000000105c32fa8;
71 uVar3 = -(int)uRam0000000105c32fa8;
72 p1Var6 = (long *)*(ulong *)param_1 + (long)(int)((uVar3 ^ 0x5461685) + (uVar3 & 0x5461685) * 2
73             * 0x57d0 +
74             (long)(int)((uVar2 | 0x5461b60) * 2 - (uVar2 ^ 0x5461b60)) * 8) ^
75             0x4f1c8;
76 (*code *) (uVar9 ^ 0x3c8e0) ();
77 uStack_16c = (uint)((int)p1Var6 == 0 && !bVar5);
78 uStack_a8 = 0;
79 switchD_10187732c_default:
80 switch(uStack_a8) {
81 case 0:
82     uStack_a8 = 9;
83     if (uStack_16c != 0) {
84         uStack_a8 = 1;
85     }
86     goto switchD_10187732c_default;
87 case 1:
88     (*code *)PTR_FUN_105c349a8();
89     uVar2 = -(int)uRam0000000105c32fa8;
90     uVar3 = -(int)uRam0000000105c32fa8;
91     pcVar11 = (code *)p1Var6[(long)(int)((uVar2 | 0x5461685) * 2 - (uVar2 ^ 0x5461685)) * 0xfa +
92     (long)(int)((uVar3 ^ 0x546212c) + (uVar3 & 0x546212c) * 2)];
93     (*code *)PTR_FUN_105bc77f8();
94     iVar8 = (int)uRam0000000105c32fa8;
95     puVar1 = (ulong *) (p1Var6 + (long)(int)((-iVar8 | 0x5461749U) + (-iVar8 & 0x5461749U)));
96     p1Var6 = (long *)(-0x59faf62fab9e933 - uRam0000000105c32fa8);
97     (*pcVar11)(p1Var6, puVar1 ^ 0x9a710, &lStack_f0, 0x546168a - iVar8);
98     lStack_70 = lStack_e8;
99     uStack_78 = uStack_c0;
100    uStack_80 = uStack_b8;
101    cStack_81 = *(char *) (lStack_e8 + 0x4c6);
102    uStack_a8 = 2;
103}

```

The flow graph shows 18 vertices. The top vertex has assembly code:

```

...71d4 str x1,[x19, #0x8]
...71d8 str w0,[x19, #0x4]
...71dc adrp x9,0x105a56000
...71e0 ldr x9,[x9, #0x1f8]
...71e4 mov w8,#0xd61f0000
...71e8 ldr w10,[x9]
...71ec and w10,w10,#0xfffffcff
...71f0 mov w22,#0x1
...71f4 cmp w10,w8
...71f8 b.eq LAB_10187721c

```

The bottom vertex has assembly code:

```

1018771fc
...71fc ldr w10,[x9, #0x4]
...7200 and w10,w10,#0xfffffcff
...7204 cmp w10,w8
...7208 b.eq LAB_10187721c

```

The middle vertex has assembly code:

```

10187720c
...720c ldr w9,[x9, #0x8]
...7210 and w9,w9,#0xfffffcff
...7214 cmp w9,w8
...7218 cset w22,eq

```

The bottom-most vertex has assembly code:

```

10187721c-LAB_10187721c
LAB_10187721c
...721c mov w20,#0x1685
...7220 movk w20,#0x546, LSL #16
...7224 adrp x21,0x105c34000
...7228 ldr x8,[x21, #0xa8]
...722c blr x8
...7230 adrp x23,0x105c32000
...7234 ldr x8,[x23, #0xfa8]
...7238 neg w8,w8
...723c orr w9,w8,w20
...7240 lsl w9,w9,#0x1
...7244 eor w8,w8,w20
...7248 sub w8,w9,w8
...724c ldr x9,[x23, #0xfa8]
...7250 neg w9,w9
...7254 add w10,w20,#0x362
...7258 eor w11,w10,w9
...725c and w9,w10,w9
...7260 add w9,w11,w9, LSL #0x1
...7264 mov w24,#0x57d0
...7268 sma... x8,w8,w24,x0
...726c ldr x8,[x8, w9, SXTW #0x3]
...7270 mov w9,#0xc8e0

```

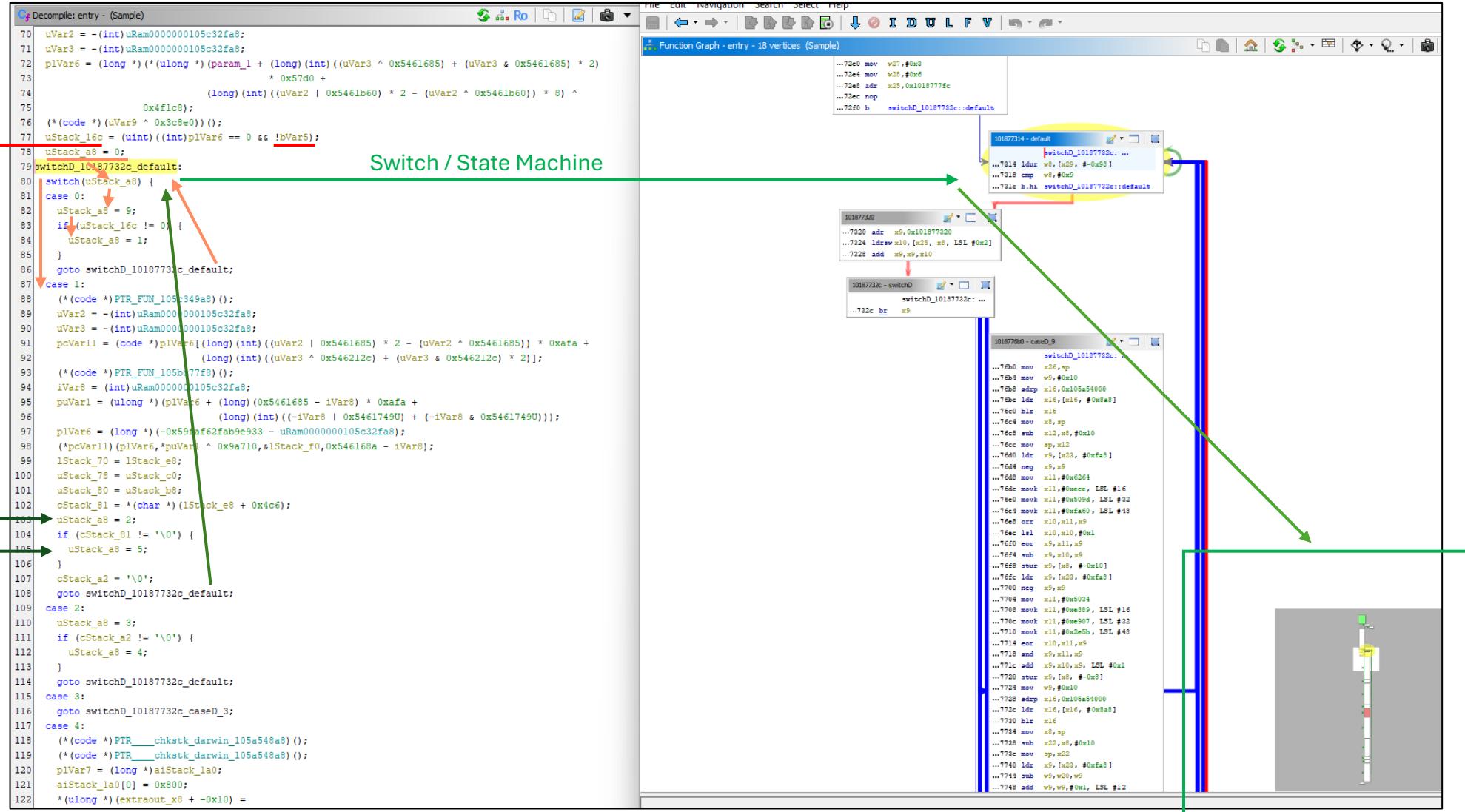
Static Analysis

Interpreting the pseudocode alongside the execution flow graph to understand the program's behavior.

Boolean Variable

Switch / State Machine

State Change



```

C# Decompile: entry - (Sample)
FILE Edit Navigation Search Select Help
Function Graph - entry - 18 vertices (Sample)
101877314 - default
...72e0 mov w27,#0x3
...72e1 mov w28,#0xd
...72e2 adr x25,[0x1018777fc]
...72e3 nop
...72e6 b switchD_10187732c::default

101877314 - default
...7314 ldrw w8,[x25, #0x98]
...7315 cmp w8,$09
...7316 b_hi switchD_10187732c::default

101877320
...7320 adr x9,[0x101877720]
...7324 ldrw x10,[x25, w8, LSL #0x2]
...7328 add x9,x10

10187732c - switchD
switchD_10187732c:: ...
...732c br x9

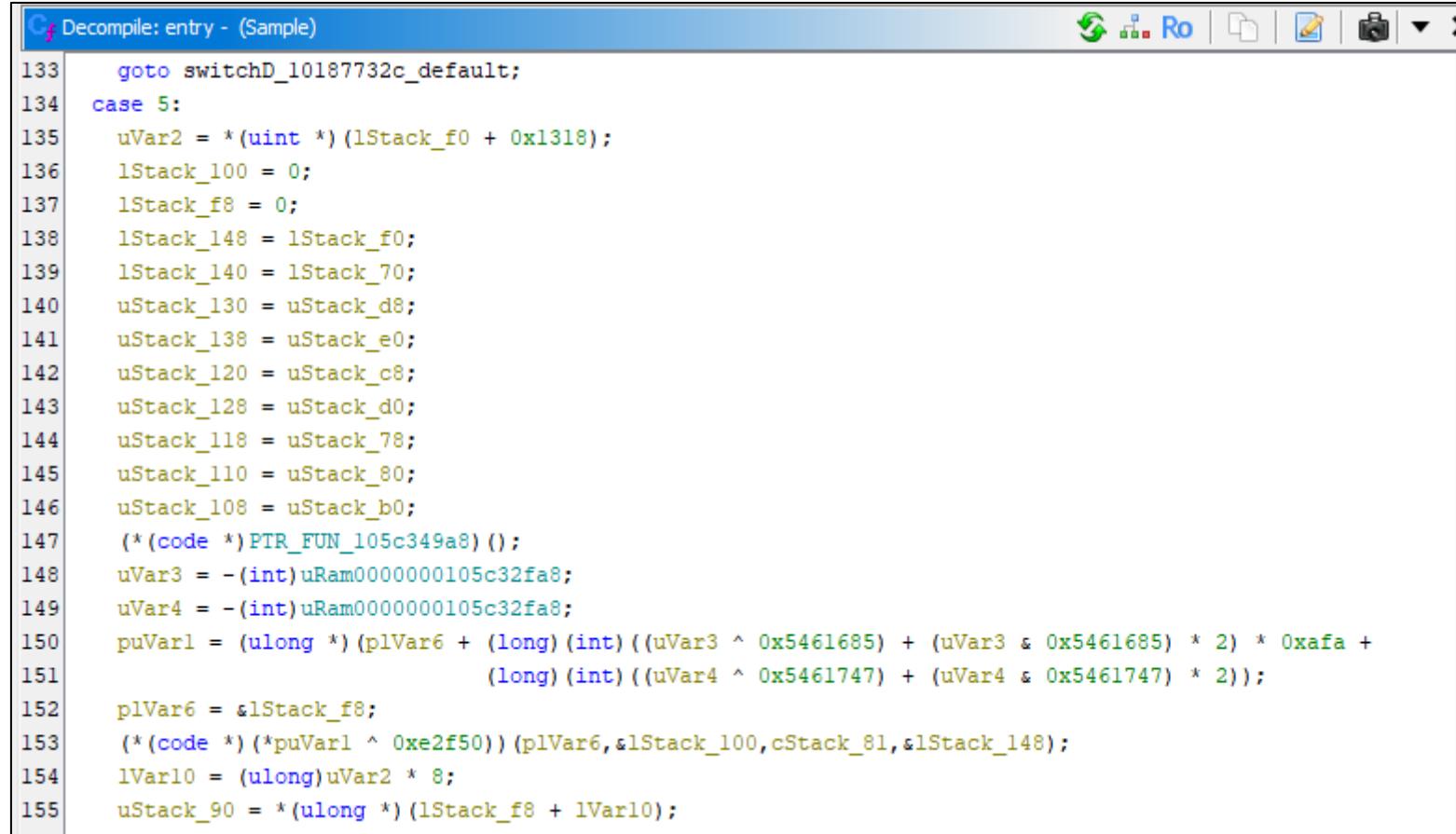
101877320 - caseD.9
switchD_10187732c:: ...
...76b0 mov x26,sp
...76b1 mov w9,#0x10
...76b2 adrpx x16,[0x105a54000
...76b3 ldr x16,[x16, #0x9a8]
...76c1 blr x16
...76c4 mov w8,sp
...76c8 sub x12,x8,#0x10
...76cc mov sp,x12
...76d0 ldr x9,[x23, #0xfa8]
...76d4 neg w9,w9
...76d8 mov x11,#0xd2d4
...76dc movx x11,#0xexce, LSL #16
...76e0 movx x11,#0xu0d4, LSL #32
...76e1 movx x11,#0xu0d0, LSL #48
...76e4 orr x10,x11,w9
...76e5 lsr x10,x10,#0x1
...76e9 eor x9,x11,w9
...76f4 sub x9,x10,w9
...76f8 ldr x9,[x23, #0xfa8]
...7700 neg w9,w9
...7704 mov x11,#0xu5024
...7708 movx x11,#0xex89, LSL #16
...770c movx x11,#0xex90, LSL #32
...7710 movx x11,#0xu25b, LSL #48
...7714 eor x10,x11,w9
...7718 and x9,x11,w9
...771c add x9,x10,w9, LSL #0x1
...7720 sturx x9,[w8, #-0x8]
...7724 mov w9,#0x10
...7728 adrpx x16,[0x105a54000
...7732 ldr x16,[x16, #0x9a8]
...7734 blr x16
...7738 sub x22,x8,#0x10
...773c mov sp,x22
...7740 ldr x9,[x23, #0xfa8]
...7744 sub w9,w20,w9
...7748 add w9,w9,#0x1, LSL #12

```

70 uVar2 = -(int)uRam000000105c32fa8;
71 uVar3 = -(int)uRam000000105c32fa8;
72 plVar6 = (long *) (ulng *) (param_1 + (long) (int) ((uVar3 ^ 0x5461685) + (uVar3 & 0x5461685) * 2) ^
 * 0x57d0 +
 (long) (int) ((uVar2 | 0x5461b60) * 2 - (uVar2 ^ 0x5461b60)) * 8) ^
 0x4f1c8);
73 (*(code *) (uVar9 ^ 0x3c8e0)) ();
74 uStack_16c = (uint) ((int)plVar6 == 0 && !bVar5);
75
76 uStack_a8 = 0;
77 switchD_10187732c_default:
78 switch(uStack_a8) {
79 case 0:
80 uStack_a8 = 9;
81 if(uStack_16c != 0) {
82 uStack_a8 = 1;
83 }
84 goto switchD_10187732c_default;
85 case 1:
86 (*(code *)PTR_FUN_105a349a8)();
87 uVar2 = -(int)uRam000000105c32fa8;
88 uVar3 = -(int)uRam000000105c32fa8;
89 pcVar11 = (code *)plVar6 [(long) (int) ((uVar2 | 0x5461685) * 2 - (uVar2 ^ 0x5461685)) * 0xafa +
 (long) (int) ((uVar3 ^ 0x546212c) + (uVar3 & 0x546212c) * 2)];
90 (*(code *)PTR_FUN_105b7758)();
91 iVar8 = (int)uRam000000105c32fa8;
92 puVar1 = (ulng *) (plVar6 + (long) (0x5461685 - iVar8) * 0xafa +
 (long) (int) ((-iVar8 | 0x5461749U) + (-iVar8 & 0x5461749U)));
93 plVar6 = (long *) (-0x59a62fab9e933 - uRam000000105c32fa8);
94 (*pcVar11) (plVar6,*puVar1 ^ 0x9a710,slStack_f0,0x546168a - iVar8);
95 lStack_70 = lStack_e8;
96 uStack_78 = uStack_c0;
97 uStack_80 = uStack_b8;
98 cStack_81 = *(char *) (lStack_e8 + 0x4c6);
99 uStack_a8 = 2;
100 if(cStack_81 != '\0') {
101 uStack_a8 = 5;
102 }
103 cStack_a2 = '\0';
104 goto switchD_10187732c_default;
105 case 2:
106 uStack_a8 = 3;
107 if(cStack_a2 != '\0') {
108 uStack_a8 = 4;
109 }
110 goto switchD_10187732c_default;
111 case 3:
112 goto switchD_10187732c_caseD_3;
113 case 4:
114 (*(code *)PTR____chkstk_darwin_105a548a8)();
115 (*(code *)PTR____chkstk_darwin_105a548a8)();
116 plVar7 = (long *)aiStack_la0;
117 aiStack_la0[0] = 0x800;
118 *(ulng *) (extraout_x8 + -0x10) =
119
120
121
122

Static Analysis

Representation of a Function Call in Pseudocode.

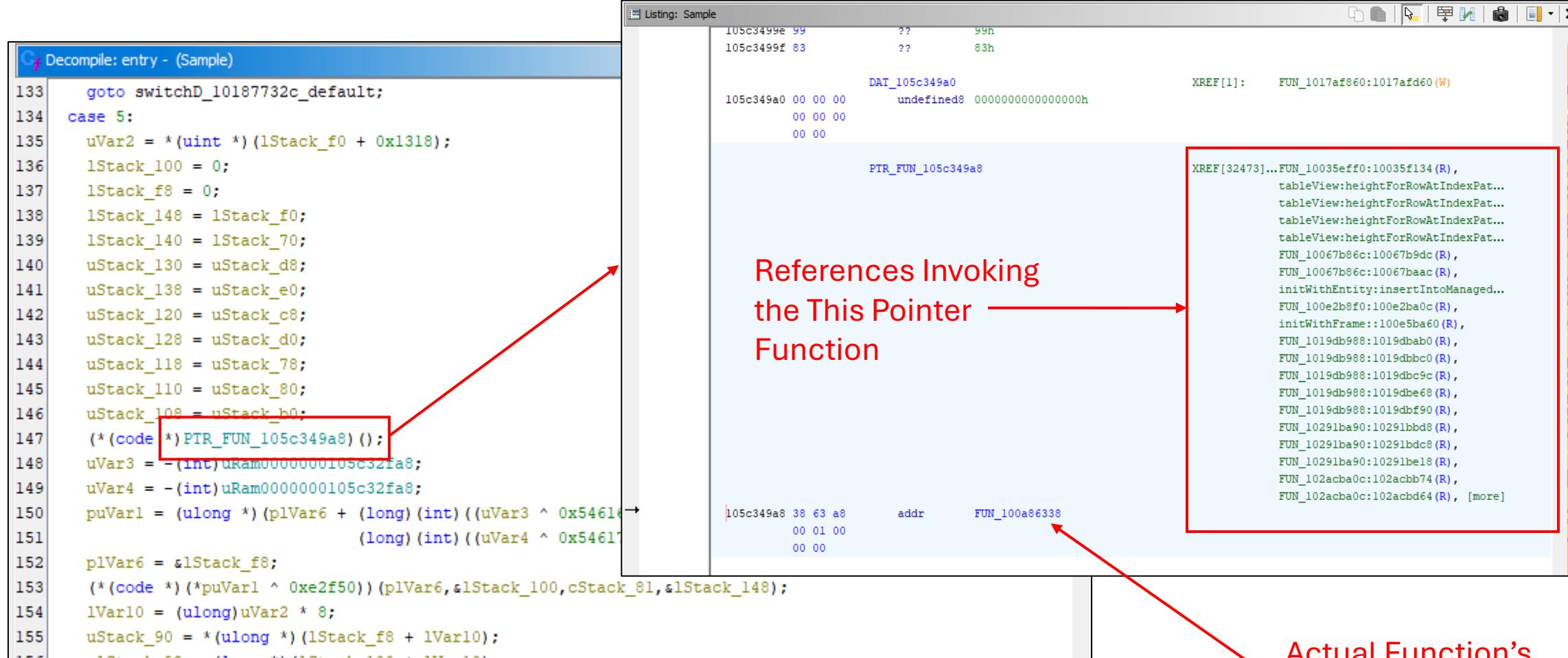


The screenshot shows a static analysis tool interface with a blue header bar containing icons for file operations and a search function. The main window displays a block of pseudocode. The code is numbered from 133 to 155. It includes declarations for variables like `uVar2`, `lStack_f0`, `lStack_100`, etc., and performs calculations involving bitwise operations and pointer arithmetic. A notable line is `puVar1 = (ulong *)(plVar6 + (long)(int)((uVar3 ^ 0x5461685) + (uVar3 & 0x5461685) * 2) * 0xfa + (long)(int)((uVar4 ^ 0x5461747) + (uVar4 & 0x5461747) * 2));`.

```
Decompile: entry - (Sample)
133    goto switchD_10187732c_default;
134    case 5:
135        uVar2 = *(uint *) (lStack_f0 + 0x1318);
136        lStack_100 = 0;
137        lStack_f8 = 0;
138        lStack_148 = lStack_f0;
139        lStack_140 = lStack_70;
140        uStack_130 = uStack_d8;
141        uStack_138 = uStack_e0;
142        uStack_120 = uStack_c8;
143        uStack_128 = uStack_d0;
144        uStack_118 = uStack_78;
145        uStack_110 = uStack_80;
146        uStack_108 = uStack_b0;
147        (*(code *)PTR_FUN_105c349a8)();
148        uVar3 = -(int)uRam00000000105c32fa8;
149        uVar4 = -(int)uRam00000000105c32fa8;
150        puVar1 = (ulong *) (plVar6 + (long)(int)((uVar3 ^ 0x5461685) + (uVar3 & 0x5461685) * 2) * 0xfa +
151                                (long)(int)((uVar4 ^ 0x5461747) + (uVar4 & 0x5461747) * 2));
152        plVar6 = &lStack_f8;
153        (*(code *)(*puVar1 ^ 0xe2f50)) (plVar6, &lStack_100, cStack_81, &lStack_148);
154        lVar10 = (ulong)uVar2 * 8;
155        uStack_90 = *(ulong *) (lStack_f8 + lVar10);
```

Static Analysis

Representation of a Function Call in Pseudocode.



Decompile: entry - (Sample)

```
133     goto switchD_10187732c_default;
134 case 5:
135     uVar2 = *(uint *) (lStack_f0 + 0x1318);
136     lStack_100 = 0;
137     lStack_f8 = 0;
138     lStack_148 = lStack_f0;
139     lStack_140 = lStack_70;
140     uStack_130 = uStack_d8;
141     uStack_138 = uStack_e0;
142     uStack_120 = uStack_c8;
143     uStack_128 = uStack_d0;
144     uStack_118 = uStack_78;
145     uStack_110 = uStack_80;
146     uStack_108 = uStack_b0;
147     (*(code *) PTR_FUN_105c349a8) ();
148     uVar3 = -(int)uRam00000000105c32fa8;
149     uVar4 = -(int)uRam00000000105c32fa8;
150     puVar1 = (ulong *) (plVar6 + (long)(int)((uVar3 ^ 0x5461e000) & 0xffffffff));
151                 (long)(int)((uVar4 ^ 0x5461e000) & 0xffffffff);
152     plVar6 = &lStack_f8;
153     (*(code *)(*puVar1 ^ 0xe2f50)) (plVar6, &lStack_100, cStack_81, &lStack_148);
154     lVar10 = (ulong)uVar2 * 8;
155     uStack_90 = *(ulong *) (lStack_f8 + lVar10);
```

Listing: Sample

105c3499e	99	??	99n
105c3499f	83	??	83h
DAT_105c349a0			
105c349a0	00 00 00	undefined8	0000000000000000h
	00 00 00		
	00 00		
PTR_FUN_105c349a8			
105c349a8	38 63 a8	addr	FUN_100a86338
	00 01 00		
	00 00		

References Invoking the This Pointer Function

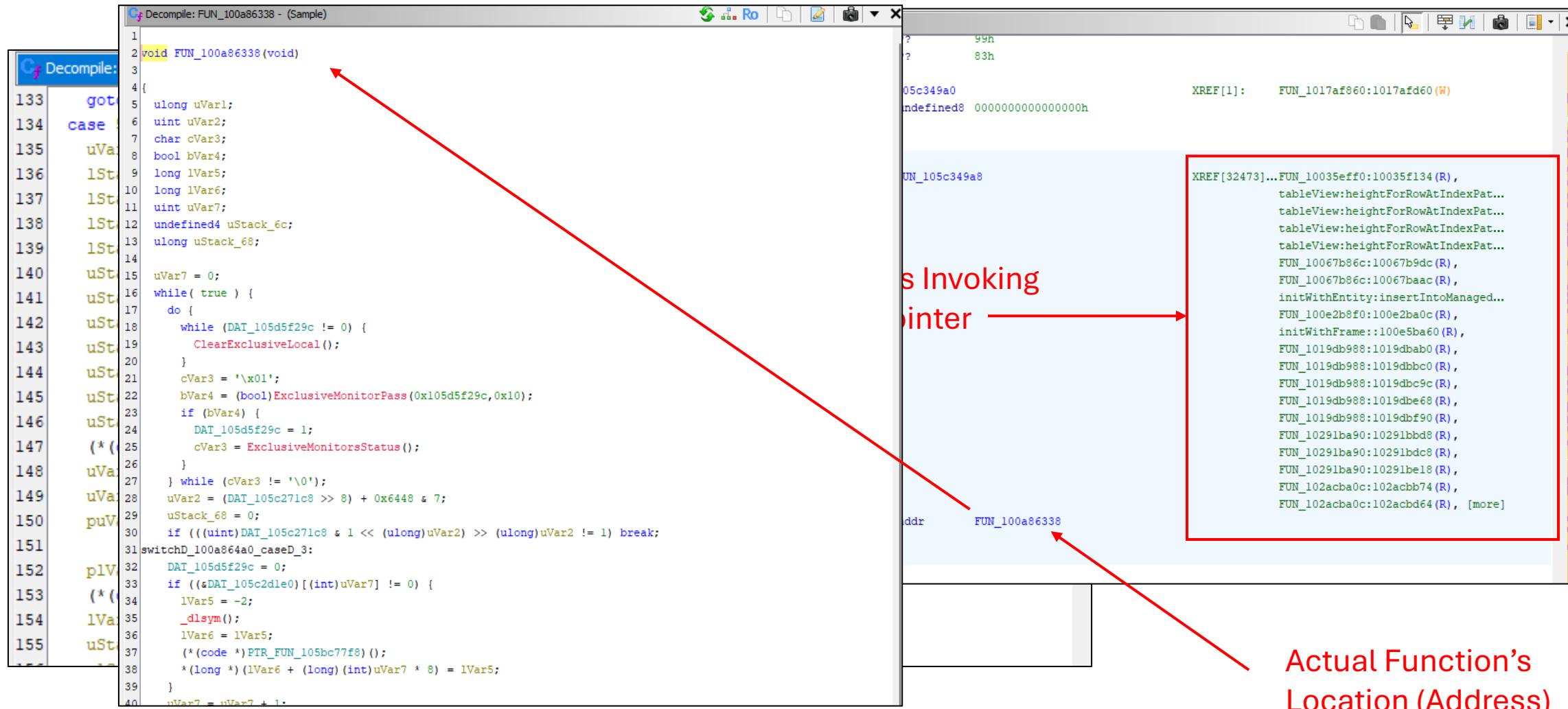
XREF[1]: FUN_1017af860:1017af860 (W)

XREF[32473]...FUN_10035eff0:10035f134 (R),
tableViewheightForRowAtIndexPathPat...
tableViewheightForRowAtIndexPathPat...
tableViewheightForRowAtIndexPathPat...
tableViewheightForRowAtIndexPathPat...
FUN_10067b86c:10067b9dc (R),
FUN_10067b86c:10067baac (R),
initWithEntity:insertIntoManaged...
FUN_100e2b8f0:100e2ba0c (R),
initWithFrame::100e5ba60 (R),
FUN_1019db988:1019dbab0 (R),
FUN_1019db988:1019dbbc0 (R),
FUN_1019db988:1019dbc9c (R),
FUN_1019db988:1019dbe68 (R),
FUN_1019db988:1019dbf90 (R),
FUN_10291ba90:10291bbd8 (R),
FUN_10291ba90:10291bdc8 (R),
FUN_10291ba90:10291be18 (R),
FUN_102aca0c:102acbb74 (R),
FUN_102aca0c:102acbd64 (R), [more]

Actual Function's Location (Address)

Static Analysis

Representation of a Function Call in Pseudocode.



The diagram illustrates the analysis of a function call. On the left, a screenshot of a debugger's decompiler window shows the pseudocode for `FUN_100a86338`. The code includes variable declarations, a loop, and a conditional break. A red arrow points from the label `got` at the top of the pseudocode to the assembly view on the right. In the assembly view, the address `00c349a0` is highlighted with a red box. A red arrow points from this box to the text "Actual Function's Location (Address)" at the bottom right. Another red arrow points from the label `got` in the pseudocode to the assembly address `00c349a0`. The assembly window also shows other references to the function and its code.

```

Decompile: FUN_100a86338 - (Sample)

1 void FUN_100a86338(void)
2
3 {
4     ulong uVar1;
5     uint uVar2;
6     char cVar3;
7     bool bVar4;
8     long lVar5;
9     long lVar6;
10    uint uVar7;
11    undefined4 uStack_6c;
12    ulong uStack_68;
13
14    uSt
15    uSt
16    while( true ) {
17        do {
18            while (DAT_105d5f29c != 0) {
19                ClearExclusiveLocal();
20            }
21            cVar3 = '\x01';
22            bVar4 = (bool)ExclusiveMonitorPass(0x105d5f29c,0x10);
23            if (bVar4) {
24                DAT_105d5f29c = 1;
25                cVar3 = ExclusiveMonitorsStatus();
26            }
27        } while (cVar3 != '\0');
28        uVar2 = (DAT_105c271c8 >> 8) + 0x6448 & 7;
29        uStack_68 = 0;
30        if (((uint)DAT_105c271c8 & 1 << (ulong)uVar2) >> (ulong)uVar2 != 1) break;
31    switchD_100a864a0_caseD_3:
32        DAT_105d5f29c = 0;
33        if ((sDAT_105c2d1e0)[(int)uVar7] != 0) {
34            lVar5 = -2;
35            _dlsym();
36            lVar6 = lVar5;
37            (*code *)PTR_FUN_105bc77f8();
38            *(long *) (lVar6 + (long)(int)uVar7 * 8) = lVar5;
39        }
40        uVar7 = uVar7 + 1;

```

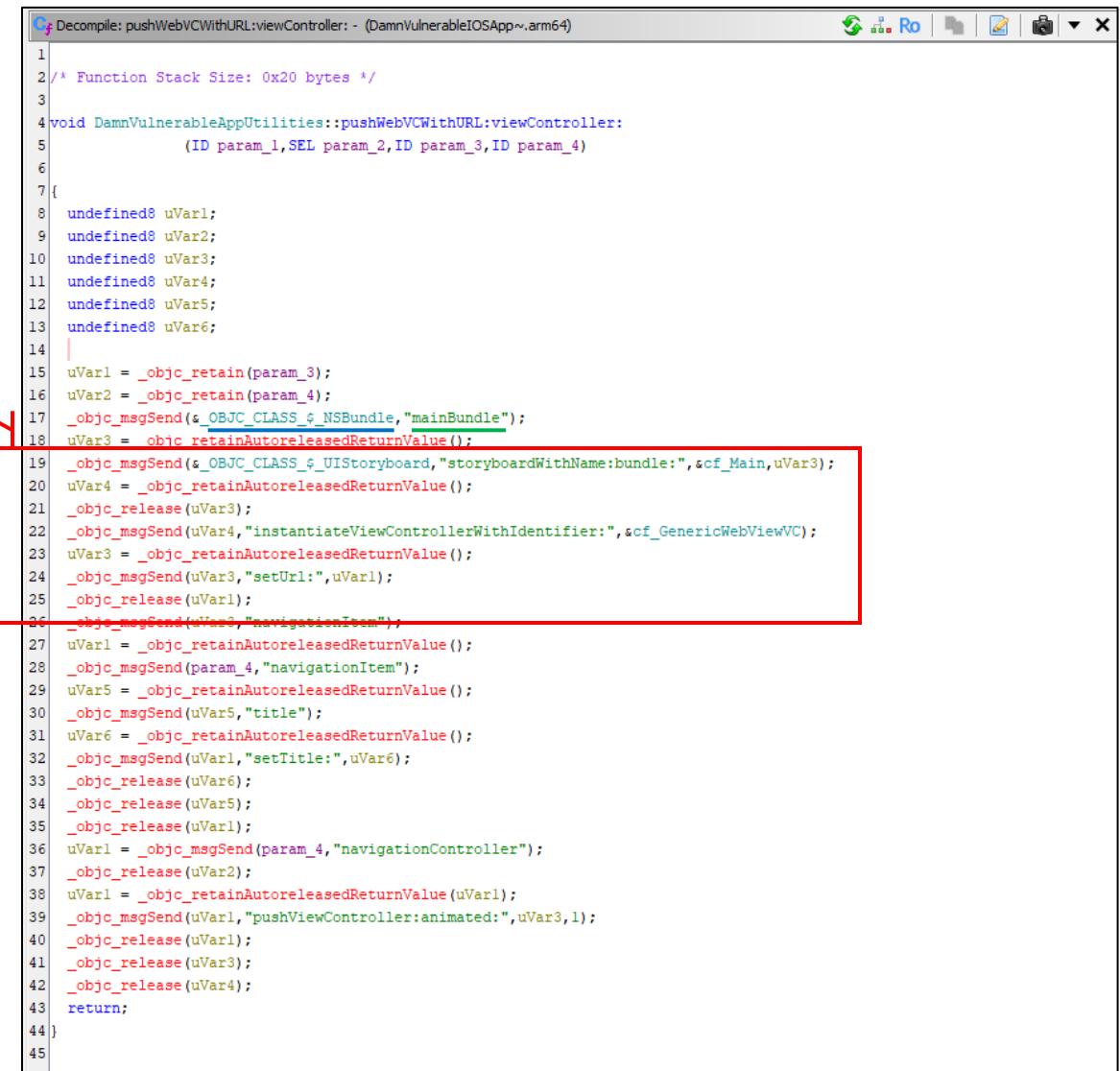
Actual Function's Location (Address)

Static Analysis

Representation of a Function Call in Pseudocode (msgSend).

```
// Get the app's main bundle  
uVar3 NSBundle *main = [NSBundle mainBundle];
```

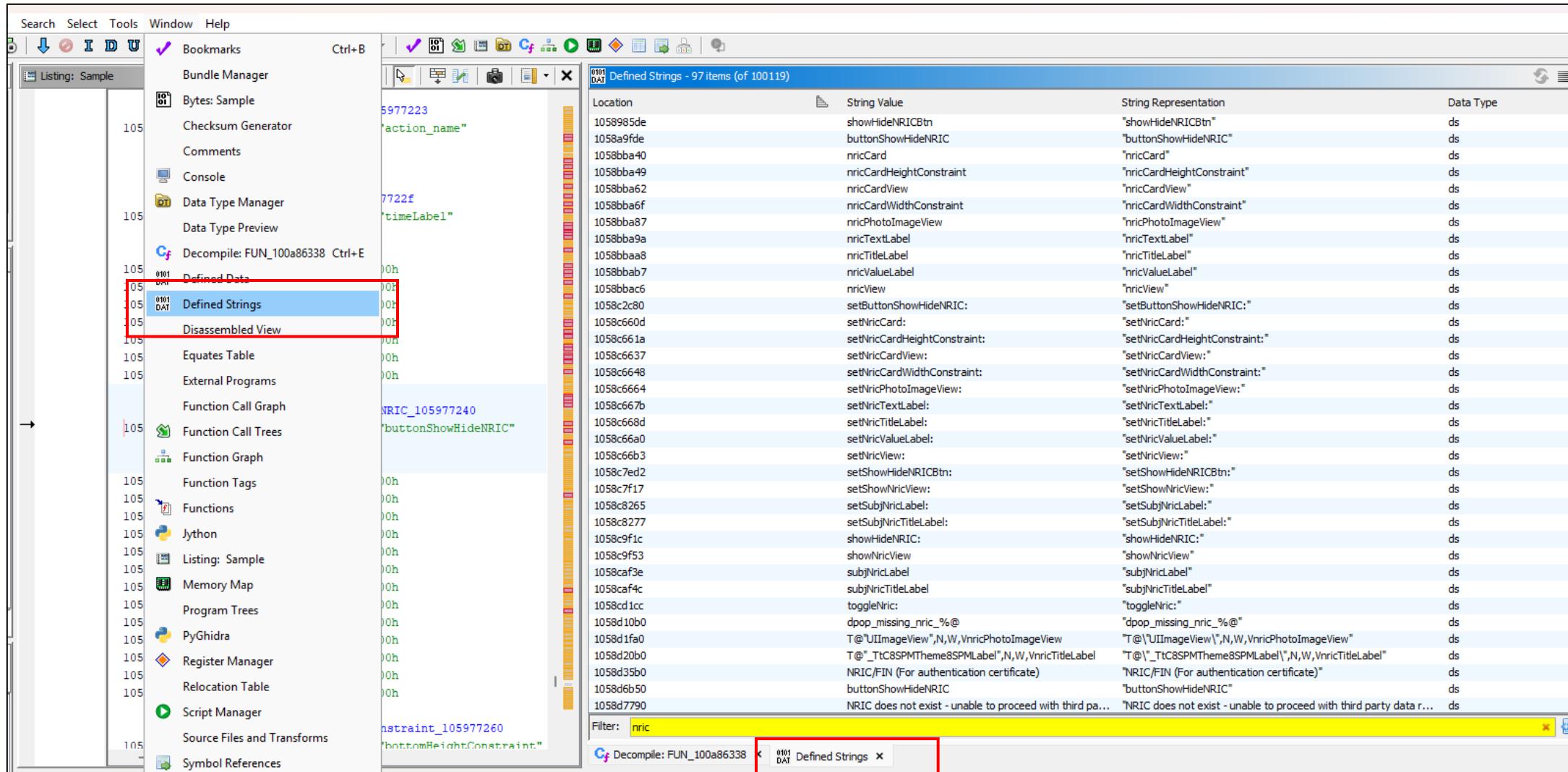
```
19 [UIStoryboard storyboardWithName:bundle:]  
20 Sets the returned UIStoryboard pointer reference to uVar4  
21 [UIStoryboard instantiateViewControllerWithIdentifier:]  
22 Sets the returned UIViewController pointer reference to uVar3  
23 [UIViewController setUrl:]
```



```
C# Decompile: pushWebVCWithURL:viewController: - (DamnVulnerableIOSApp~.arm64)  
1  
2 /* Function Stack Size: 0x20 bytes */  
3  
4 void DamnVulnerableAppUtilities::pushWebVCWithURL:viewController:  
5 (ID param_1,SEL param_2,ID param_3,ID param_4)  
6  
7 {  
8     undefined8 uVar1;  
9     undefined8 uVar2;  
10    undefined8 uVar3;  
11    undefined8 uVar4;  
12    undefined8 uVar5;  
13    undefined8 uVar6;  
14  
15    uVar1 = _objc_retain(param_3);  
16    uVar2 = _objc_retain(param_4);  
17    _objc_msgSend(&_OBJC_CLASS_$_NSBundle,"mainBundle");  
18    uVar3 = _objc_retainAutoreleasedReturnValue();  
19    _objc_msgSend(&_OBJC_CLASS_$_UIStoryboard,"storyboardWithName:bundle:",&cf_Main,uVar3);  
20    uVar4 = _objc_retainAutoreleasedReturnValue();  
21    _objc_release(uVar3);  
22    _objc_msgSend(uVar4,"instantiateViewControllerWithIdentifier:",&cf_GenericWebViewVC);  
23    uVar3 = _objc_retainAutoreleasedReturnValue();  
24    _objc_msgSend(uVar3,"setUrl:",uVar1);  
25    _objc_release(uVar1);  
26    _objc_msgSend(uVar3,"navigationItem");  
27    uVar1 = _objc_retainAutoreleasedReturnValue();  
28    _objc_msgSend(param_4,"navigationItem");  
29    uVar5 = _objc_retainAutoreleasedReturnValue();  
30    _objc_msgSend(uVar5,"title");  
31    uVar6 = _objc_retainAutoreleasedReturnValue();  
32    _objc_msgSend(uVar1,"setTitle:",uVar6);  
33    _objc_release(uVar6);  
34    _objc_release(uVar5);  
35    _objc_release(uVar1);  
36    uVar1 = _objc_msgSend(param_4,"navigationController");  
37    _objc_release(uVar2);  
38    uVar1 = _objc_retainAutoreleasedReturnValue(uVar1);  
39    _objc_msgSend(uVar1,"pushViewController:animated:",uVar3,1);  
40    _objc_release(uVar1);  
41    _objc_release(uVar3);  
42    _objc_release(uVar4);  
43    return;  
44}  
45
```

Static Analysis

Strings (Windows > Defined Strings)



Location	String Value	String Representation	Data Type
105 001 DAT	showHideNRICBtn	"showHideNRICBtn"	ds
105 001 DAT	buttonShowHideNRIC	"buttonShowHideNRIC"	ds
105 001 DAT	nricCard	"nricCard"	ds
105 001 DAT	nricCardHeightConstraint	"nricCardHeightConstraint"	ds
105 001 DAT	nricCardView	"nricCardView"	ds
105 001 DAT	nricCardWidthConstraint	"nricCardWidthConstraint"	ds
105 001 DAT	nricPhotoImageView	"nricPhotoImageView"	ds
105 001 DAT	nricTextLabel	"nricTextLabel"	ds
105 001 DAT	nricTitleLabel	"nricTitleLabel"	ds
105 001 DAT	nricValueLabel	"nricValueLabel"	ds
105 001 DAT	nricView	"nricView"	ds
105 001 DAT	setButtonShowHideNRIC:	"setButtonShowHideNRIC:"	ds
105 001 DAT	setNricCard:	"setNricCard:"	ds
105 001 DAT	setNricCardHeightConstraint:	"setNricCardHeightConstraint:"	ds
105 001 DAT	setNricCardView:	"setNricCardView:"	ds
105 001 DAT	setNricCardWidthConstraint:	"setNricCardWidthConstraint:"	ds
105 001 DAT	setNricPhotoImageView:	"setNricPhotoImageView:"	ds
105 001 DAT	setNricTextLabel:	"setNricTextLabel:"	ds
105 001 DAT	setNricTitleLabel:	"setNricTitleLabel:"	ds
105 001 DAT	setNricValueLabel:	"setNricValueLabel:"	ds
105 001 DAT	setNricView:	"setNricView:"	ds
105 001 DAT	setShowHideNRICBtn:	"setShowHideNRICBtn:"	ds
105 001 DAT	setShowNricView:	"setShowNricView:"	ds
105 001 DAT	setSubNricLabel:	"setSubNricLabel:"	ds
105 001 DAT	setSubjNrictitleLabel:	"setSubjNrictitleLabel:"	ds
105 001 DAT	showHideNRIC:	"showHideNRIC:"	ds
105 001 DAT	showNricView	"showNricView"	ds
105 001 DAT	subjNricLabel	"subjNricLabel"	ds
105 001 DAT	subjNrictitleLabel	"subjNrictitleLabel"	ds
105 001 DAT	toggleNric:	"toggleNric:"	ds
105 001 DAT	dpop_missing_nric_%@	"dpop_missing_nric_%@"	ds
105 001 DAT	T@\"UIImageview\",N,W,VnricPhotoImageView	"T@\"UIImageview\",N,W,VnricPhotoImageView"	ds
105 001 DAT	T@\"_TtC8SPMTheme8SPMLabel\",N,W,VnricTitleLabel	"T@\"_TtC8SPMTheme8SPMLabel\",N,W,VnricTitleLabel"	ds
105 001 DAT	NRIC/FIN (For authentication certificate)	"NRIC/FIN (For authentication certificate)"	ds
105 001 DAT	buttonShowHideNRIC	"buttonShowHideNRIC"	ds
105 001 DAT	NRIC does not exist - unable to proceed with third pa...	"NRIC does not exist - unable to proceed with third party data r..."	ds

Static Analysis

Ghidra + GhidraMCP + Claude Sonnet 4

N in my ghidra open project, if i want to bypass the jailbreak detection, which function(s) should i look out for?

I can help you analyze functions related to jailbreak detection in your Ghidra project. Let me first understand what we're working with by exploring the binary.

G List functions

The function list is quite large. Let me search for functions that commonly relate to jailbreak detection:

G Search functions by name

Great! I found many jailbreak detection common patterns:

G Search functions by name

G Search functions by name

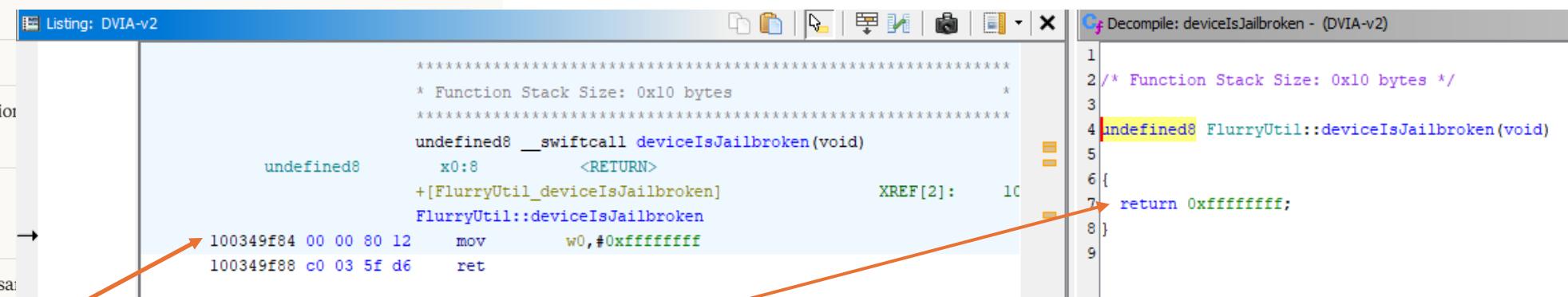
Let me also search for "device" and "sa

G Search functions by name

Perfect! Now let me look at some specific key functions. The `deviceIsJailbroken` function at address `100349f84` looks like a key function. Let me also check the main jailbreak detection function:

G Decompile function

Interesting - that one just returns false. Let me check the main jailbreak detection function:



The screenshot shows the Ghidra interface with two windows. On the left is the 'Listing' window for 'DVIA-v2', showing assembly code for a function at address 100349f84. The assembly code is:

```
*****  
* Function Stack Size: 0x10 bytes  
*****  
undefined8 __swiftcall deviceIsJailbroken(void)  
    x0:8      <RETURN>  
+[FlurryUtil_deviceIsJailbroken]  
FlurryUtil::deviceIsJailbroken  
100349f84 00 00 80 12  mov    w0,#0xffffffff  
100349f88 c0 03 5f d6  ret
```

The right window is the 'Decompile' window for the same function, showing C-like pseudocode:

```
/* Function Stack Size: 0x10 bytes */  
  
undefined8 FlurryUtil::deviceIsJailbroken(void)  
{  
    return 0xffffffff;  
}
```

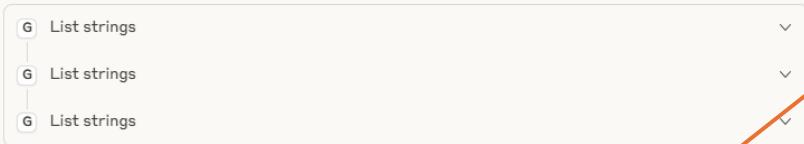
An orange arrow points from the assembly instruction `mov w0, #0xffffffff` in the listing window to the corresponding `return 0xffffffff;` line in the decompiled code window.

<https://github.com/LaurieWired/GhidraMCP>

Static Analysis

Ghidra + GhidraMCP + Claude Sonnet 4

Let me also look for common jailbreak files:



Perfect! Now I have a comprehensive understanding of the jailbreak detection mechanisms.

Let me create a detailed analysis:



Based on my analysis of your Ghidra project, I've identified the key jailbreak detection functions and created a comprehensive bypass guide.

The main functions you should focus on are:

1. `deviceIsJailbroken` at address `100349f84` - This is already returning false
2. `isJailbroken` at `10012127c` - The main Objective-C detection method
3. **Main Swift detection function** at `100141e18` - Core detection logic that checks multiple indicators

The app uses several detection techniques:

- **File system checks** for `/Applications/Cydia.app`, `/Library/MobileSubstrate/MobileSubstrate.dylib`, `/bin/bash`, `/usr/sbin/sshd`, and `/etc/apt`
- **URL scheme detection** testing `cydia://` URLs
- **File write tests** attempting to write to `/private/jailbreak.txt`

The most effective bypass approach would be to hook the

`NSFileManager.fileExistsAtPath:` method to return false for jailbreak-related paths, and hook `UIApplication.canOpenURL:` to block Cydia URL schemes.

Would you like me to help you implement any specific bypass technique, or would you like me to analyze any other particular functions in more detail?

Jailbreak Detection Analysis & Bypass Guide

Based on my analysis of your Ghidra project, I've identified the key jailbreak detection mechanisms and functions you should target for bypassing. This appears to be the DVIA-v2 (Damn Vulnerable iOS Application) which implements multiple jailbreak detection techniques.

Key Functions to Target

1. Main Detection Functions

Primary Functions:

- `deviceIsJailbroken` at `100349f84` - Currently returns `0xffffffff` (false)
- `isJailbroken` at `10012127c` - Objective-C method for jailbreak detection
- `$_s7DVIA_v2$2JailbreakDetectionViewControllerC1isJailbrokenSbyF` at `100141e18` - Main Swift detection function

Test Functions:

- `jailbreakTest3yyF` at `1001427bc` - File-based detection method
- `jailbreakTest1Tapped` through `jailbreakTest5Tapped` - Individual test methods

2. Detection Methods Used

The analysis reveals several detection techniques:

File System Checks

The application checks for the existence of these jailbreak-related files:

- `/Applications/Cydia.app` - Cydia package manager
- `/Library/MobileSubstrate/MobileSubstrate.dylib` - Substrate hooking framework
- `/bin/bash` - Unix shell
- `/usr/sbin/sshd` - SSH daemon
- `/etc/apt` - APT package manager directory
- `/private/jailbreak.txt` - Test file creation

URL Scheme Detection

- `cydia://package/com.example.package` - Tests if Cydia URL scheme can be opened

File Write Test

- Attempts to write "This is a test." to `/private/jailbreak.txt`
- If successful, indicates elevated privileges typical of jailbroken devices

Bypass Strategies

1. Function Hooking (Recommended)

Hook the main detection functions:

```
objc
// For Objective-C method
%hook JailbreakDetection
+ (BOOL)isJailbroken {
    return NO;
}
%end

// For individual test methods
%hook JailbreakDetectionViewController
- (void)jailbreakTest1Tapped:(id)sender { /* Do nothing */ }
- (void)jailbreakTest2Tapped:(id)sender { /* Do nothing */ }
- (void)jailbreakTest3Tapped:(id)sender { /* Do nothing */ }
- (void)jailbreakTest4Tapped:(id)sender { /* Do nothing */ }
- (void)jailbreakTest5Tapped:(id)sender { /* Do nothing */ }
%end
```

2. Function Patching

Target addresses for binary patching:

- `0x100141e18` - Main `isJailbrokenSbyF` function
- `0x1001427bc` - `jailbreakTest3yyF` function
- `0x100349f84` - `deviceIsJailbroken` function

Simple patches:

- Replace function starts with `mov w0, #0; ret` (ARM64) to always return false
- NOP out conditional branches that check for jailbreak indicators

3. File System Manipulation

Hide jailbreak files:

- Use substrate to hook `fileExistsAtPath:` in `NSFileManager`
- Return `NO` for paths containing jailbreak indicators
- Implement file system overlays to hide jailbreak artifacts

4. URL Scheme Blocking

Hook URL handling:

- Hook `canOpenURL:` in `UIApplication`
- Return `NO` for Cydia and other jailbreak-related URL schemes

Static Analysis

Ghidra + GhidraMCP + Claude Sonnet 4

Listing: DVIA-v2

```

undefined4 Stack[-0xac]:4 local_ac 100121520(R)
undefined8 Stack[-0xb8]:8 local_b8 XREF[2]: 100121584(W),
                                100121594(R)
undefined8 Stack[-0xc0]:8 local_c0 XREF[2]: 100121548(W),
                                100121570(R),
                                10012158c(R)
undefined8 Stack[-0xe8]:8 local_c8 XREF[2]: 1001215e0(W),
                                1001215e8(R)

/*[JailbreakDetection_isJailbroken]
JailbreakDetection::isJailbroken
10012127c ff 43 03 d1 sub sp,sp,#0xd0
100121280 fd 7b 0c a9 stp x29,x30,[sp, #local_10]
100121284 fd 03 03 91 add x29,sp,#0xc0
100121288 e8 03 01 aa mov x8,param_2
10012128c a0 03 01 f8 stur param_1,[x29, #local_20]
100121290 a8 83 le f8 stur x8,[x29, #local_28]
100121294 68 1d 00 adrp x8,0x1004cf000
100121298 00 d1 43 f9 ldr param_1=>_OBJC_CLASS_6_NSFileManager,[x8, #0x7... = ???
= 1007f4808
10012129c 9a 3d 09 94 bl _objc_msgSenddefaultManager
1001212a0 fd 03 1d aa mov x29,x29 undefined _objc_msgSenddefaultM...
1001212a4 d4 25 09 94 bl _objc_retainAutoreleasedReturnValue
1001212a8 a1 83 5b f8 ldur param_2,[x29, #local_58]
1001212ac a0 03 1c f8 stur param_1,[x29, #local_50]
1001212b0 c2 18 00 b0 adrp x2,0x10043a00
1001212b4 42 c0 17 91 add x2>>cf/_Applications/Cydia.app,x2,#0x5f0
1001212b8 03 42 09 94 bl _objc_msgSendfileExistsAtPath:
1001212bc e8 03 00 aa mov x8,param_1
1001212c0 a0 03 5c f8 ldur param_1,[x29, #local_50]
1001212c4 a8 83 1c b8 stur w8,[x29, #local_48]
1001212c8 92 25 09 94 bl _objc_release
1001212cc a0 83 5c b8 ldur param_1,[x29, #local_48]
1001212d0 e0 00 00 36 tbz param_1,#0x0,LAB_1001212ec
1001212d4 01 00 00 14 b LAB_1001212d8

LAB_1001212d8 XREF[1]: 1001212d4(j)
1001212d8 28 00 80 52 mov w8,#0x1
1001212dc 08 01 00 12 and w8,w8,#0x1
1001212e0 08 01 00 12 and w8,w8,#0x1
1001212e4 a8 3f 1f 38 sturb w8,[x29, #local_11]
1001212e8 c4 00 00 14 b LAB_1001215f8

LAB_1001212ec XREF[1]: 1001212d0(j)
1001212ec a1 83 5b f8 ldur param_2,[x29, #local_58]
1001212f0 68 1d 00 d0 adrp x8,0x1004cf000

```

```

Decompile: isJailbroken - (DVIA-v2)

1 /* Function Stack Size: 0x10 bytes */
2
3 /**
4  * JailbreakDetection::isJailbroken(undefined8 param_1,
5  *
6  * @param1
7  * long lVar1;
8  * undefined *puVar2;
9  * undefined *puVar3;
10 * undefined *puVar4;
11 * cfstringStruct *pcVar5;
12 * SEL local_58;
13 * ID local_38;
14 * long local_30;
15 * undefined8 local_28;
16 * undefined8 local_20;
17 * byte local_11;
18
19 * puVar2 = &_OBJC_CLASS_6_NSFileManager;
20 * local_28 = param_2;
21 * local_20 = param_1;
22 * _objc_msgSenddefaultManager(&_OBJC_CLASS_6_NSFileManager,
23 * _objc_retainAutoreleasedReturnValue());
24 * puVar3 = puVar2;
25 * _objc_msgSendfileExistsAtPath:(puVar2);
26 * _objc_release(puVar2);
27 * if (((ulong)puVar3 & 1) == 0) {
28 *     puVar2 = &_OBJC_CLASS_6_NSFileManager;
29 *     _objc_msgSenddefaultManager(&_OBJC_CLASS_6_NSFileManager,
30 *     _objc_retainAutoreleasedReturnValue());
31 *     puVar3 = puVar2;
32 *     _objc_msgSendfileExistsAtPath:(puVar2);
33 *     _objc_release(puVar2);
34 *     if (((ulong)puVar3 & 1) == 0) {
35 *         puVar2 = &_OBJC_CLASS_6_NSFileManager;
36 *         _objc_msgSenddefaultManager(&_OBJC_CLASS_6_NSFileManager,
37 *         _objc_retainAutoreleasedReturnValue());
38 *         puVar3 = puVar2;
39 *         _objc_msgSendfileExistsAtPath:(puVar2);
40 *         _objc_release(puVar2);
41 *         if (((ulong)puVar3 & 1) == 0) {
42 *             puVar2 = &_OBJC_CLASS_6_NSFileManager;
43 *             _objc_msgSenddefaultManager(&_OBJC_CLASS_6_NSFileManager,
44 *             _objc_retainAutoreleasedReturnValue());
45 *             puVar3 = puVar2;
46 *             _objc_msgSendfileExistsAtPath:(puVar2);
47 *             _objc_release(puVar2);

```

N i don't understand `isJailbroken` at `10012127c`. go through the function and tell me what it does

Let me decompile and analyze that specific function for you:

G Decompile function by address

The decompilation is a bit hard to read because Ghidra couldn't fully resolve the string constants. Let me check what strings are being referenced and get a better understanding:

G Get xrefs to

Let me also look for the actual function signature to understand it better:

G Search functions by name

Let me look at the disassembly to better understand what's happening:

G Disassemble function

The disassembly is clearer. Let me check the string references to understand what file paths are being checked:

G List data items

Let me check the specific memory locations that are being referenced in the function:

G List strings

Now I understand! The function is checking for several specific jailbreak-related files. Looking at the assembly code and the strings I've found, let me create a comprehensive analysis:

Analysis of `isJailbroken` Function at `0x10012127c`
Document



Now I understand the `isJailbroken` function at `0x10012127c` completely! Here's what it does:

Static Analysis

Ghidra + GhidraMCP + Claude Sonnet 4

Function Breakdown

This is an Objective-C class method `+[JailbreakDetection isJailbroken]` that uses a **multi-layered detection approach**:

1. File Existence Checks (5 sequential checks)

It checks for these jailbreak-related files **in order**:

- `/Applications/Cydia.app` - Cydia package manager
- `/Library/MobileSubstrate/MobileSubstrate.dylib` - Hooking framework
- `/bin/bash` - Unix shell (not on stock iOS)
- `/usr/sbin/sshd` - SSH daemon
- `/etc/apt` - APT package manager

If ANY of these files exist → immediately returns TRUE (jailbroken)

2. Privilege Escalation Test

If no files are found, it tests write privileges:

- Tries to write "This is a test." to `/private/jailbreak.txt`
- If write fails** → returns FALSE (not jailbroken)
- If write succeeds** → continues to next test (indicates elevated privileges)

3. URL Scheme Test

As a final check:

- Tests if `cydia://package/com.example.package` can be opened
- If YES** → returns TRUE (Cydia installed)
- If NO** → returns FALSE (not jailbroken)

Why This Function is Important

This is the **core detection logic** - it's more sophisticated than the other functions because it:

- Uses multiple detection vectors
- Tests actual privilege escalation (not just file presence)
- Has a fallback URL scheme test
- Uses early return optimization (stops at first positive detection)

Bypass Strategies

1. Hook NSFileManager (Most Effective)

```
objc
%hook NSFileManager
- (BOOL)fileExistsAtPath:(NSString *)path {
    if ([path isEqualToString:@"Cydia"] ||
        [path isEqualToString:@"MobileSubstrate"] ||
        [path isEqualToString:@"/bin/bash"] ||
        [path isEqualToString:@"/usr/sbin/sshd"] ||
        [path isEqualToString:@"/etc/apt"]){
        return NO;
    }
    return %orig;
}
%end
```

2. Hook String Write Method

```
objc
%hook NSString
- (BOOL)writeToFile:(NSString *)path
    atomically:(BOOL)atomically
    encoding:(NSStringEncoding)encoding
    error:(NSError **)error {
    if ([path isEqualToString:@"jailbreak.txt"]){
        // Simulate write failure
        if (error) {
            *error = [NSError errorWithDomain:NSCocoaErrorDomain
                                         code:NSFileWriteNoPermissionError
                                         userInfo:nil];
        }
        return NO;
    }
    return %orig;
}
%end
```

3. Hook UIApplication URL Test

```
objc
%hook UIApplication
- (BOOL)canOpenURL:(NSURL *)url {
    if ([url.scheme isEqualToString:@"cydia"]){
        return NO;
    }
    return %orig;
}
%end
```

4. Direct Function Hook (Simplest)

```
objc
%hook JailbreakDetection
+ (BOOL)isJailbroken {
    return NO;
}
%end
```

Key Insights

- This function uses a **cascading detection approach** - if any early check fails, it returns immediately
- The **file write test** is particularly clever as it tests actual privilege escalation rather than just file presence
- The **URL scheme test** acts as a secondary confirmation for Cydia presence
- Return value:** 1 (TRUE) = jailbroken, 0 (FALSE) = not jailbroken

Recommendation

The most effective bypass is to **hook this function directly** since it's a central point of failure. Alternatively, hook `NSFileManager.fileExistsAtPath:` to handle all file-based detection attempts across the application.

Static Analysis

Ghidra Decompiled Code + ChatGPT 5 (Thinking)

assuming you are a professional reverse engineer, what does this function does?

Thought for 46s

Short version: it's a **jailbreak-detection routine**. It returns **1/true** if it finds common jailbreak indicators; otherwise **0/false**.

What it checks (in order)

1. Presence of well-known jailbreak files/paths via `NSFileManager fileExistsAtPath:`
The decompile shows repeated `defaultManager` → `fileExistsAtPath:` checks. One constant visible is `"/etc/apt"`; the others are the usual suspects (e.g., `/Applications/Cydia.app`, `/Library/MobileSubstrate/MobileSubstrate.dylib`, `/bin/bash`, `/usr/sbin/sshd`, `/private/var/lib/apt`), and if any of them exist it immediately flags as jailbroken.
2. Sandbox escape (write test)
It builds an `NSString` and calls `writeToFile:atomically:encoding:error:` to a path outside the app sandbox (classic trick like `/private/jb.txt`).
 - If the write succeeds (no `NSError`), it treats that as jailbroken (because a normal sandboxed app shouldn't be able to write there).
 - If the write fails, it proceeds, and if it did create a file it cleans it up with `removeItemAtPath:error:`.
3. Cydia URL scheme probe
It constructs an `NSURL` and asks `[[UIApplication sharedApplication] canOpenURL:url]`. If the app can open a `cydia://...` URL, it flags as jailbroken.

Return value

- Any positive hit above sets `local_11 = 1` (jailbroken).
- If none hit, it returns `0` (not jailbroken).

Static Analysis

ChatGPT 5 (Thinking) Pseudocode vs Source Code

Readable pseudocode (what it's trying to do)

```
objc
BOOL isJailbroken(void) {
    NSArray *paths = @[@"~/Applications/Cydia.app",
                      @"~/Library/MobileSubstrate/MobileSubstrate.dylib",
                      @"~/bin/bash",
                      @"~/usr/sbin/sshd",
                      @"~/etc/apt"];
    for (NSString *p in paths) {
        if ([[NSFileManager defaultManager] fileExistsAtPath:p]) return YES;
    }

    NSError *err = nil;
    BOOL wrote = [@"test" writeToFile:@"~/private/jb.txt"
                           atomically:YES
                             encoding:NSUTF8StringEncoding
                               error:&err];
    if (wrote && !err) {
        [[NSFileManager defaultManager] removeItemAtPath:@"~/private/jb.txt" error:nil];
        return YES;
    }

    NSURL *u = [NSURL URLWithString:@"cydia://package/com.example"];
    if ([[UIApplication sharedApplication] canOpenURL:u]) return YES;

    return NO;
}
```

Bottom line: this function implements a standard, older-style **multi-indicator jailbreak check** and returns true if **any** indicator is present. It's fine for demos/training, but trivial to evade and noisy on modern iOS.

```
16  #import <Foundation/Foundation.h>
17  #import "JailbreakDetection.h"
18
19  @implementation JailbreakDetection
20
21  +(BOOL)isJailbroken{
22
23      #if !(TARGET_IPHONE_SIMULATOR)
24
25      if ([[NSFileManager defaultManager] fileExistsAtPath:@"~/Applications/Cydia.app"]){
26          return YES;
27      }else if([[NSFileManager defaultManager] fileExistsAtPath:@"~/Library/MobileSubstrate/MobileSubstrate.dylib"]){
28          return YES;
29      }else if([[NSFileManager defaultManager] fileExistsAtPath:@"~/bin/bash"]){
30          return YES;
31      }else if([[NSFileManager defaultManager] fileExistsAtPath:@"~/usr/sbin/sshd"]){
32          return YES;
33      }else if([[NSFileManager defaultManager] fileExistsAtPath:@"~/etc/apt"]){
34          return YES;
35      }
36
37      NSError *error;
38      NSString *stringToBeWritten = @"This is a test.";
39      [stringToBeWritten writeToFile:@"~/private/jailbreak.txt" atomically:YES
40                           encoding:NSUTF8StringEncoding error:&error];
41      if(error==nil){
42          //Device is jailbroken
43          return YES;
44      } else {
45          [[NSFileManager defaultManager] removeItemAtPath:@"~/private/jailbreak.txt" error:nil];
46      }
47
48      if([[UIApplication sharedApplication] canOpenURL:[NSURL URLWithString:@"cydia://package/com.example.package"]]){
49          //Device is jailbroken
50          return YES;
51      }
52  #endif
53
54      //All checks have failed. Most probably, the device is not jailbroken
55      return NO;
56  }
57
58
59  @end
```

Static Analysis

Static analysis involves analyzing the software's code without executing the program.



Dynamic Analysis

Dynamic analysis is a technique that analyzes the software while executing the program.

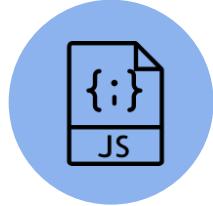


Dynamic Analysis

Dynamic analysis is a technique that analyzes the software while executing the program.



In this section, we include basic iOS Frida script templates that which you can use immediately in your test.



Next, we include Frida code snippets that worked for us for our tests.
(Version before Frida 17)

References:

- <https://learnfrida.info/macOS/>
- <https://github.com/iddoeldor/frida-snippets>
- <https://awakened1712.github.io/hacking/hacking-frida/>

Dynamic Analysis

Quick Note:

A major difference for Frida version before and after v17.

stackoverflow.com/questions/79700740/frida-17-module-getexportbyname-typeerror-not-a-function

stackoverflow About Products For Teams Search... Sorted by: Highest score (default)

1 Answer

Frida 17 introduced some breaking changes which includes renaming some functions.
To migrate such scripts you need to apply these changes:

Frida command before 17	Frida command for 17+
Module.getGlobalExportByName(null, "open");	Module.getExportByName("open");
Module.findExportByName(null, "open");	Module.getExportByName("open");
Module.getSymbolByName(null, 'open')	Module.getGlobalExportByName('open')
Module.getExportByName('libc.so', 'open')	Process.getModuleByName('libc.so').getExportByName
Module.getBaseAddress("libc.so")	Process.getModuleByName('libc.so').base
Memory.readCString(somePtr)	somePtr.readCString()
Memory.readUtf8String(somePtr)	somePtr.readUtf8String()
Memory.readUtf16String(somePtr)	somePtr.readUtf16String()
Memory.readAnsiString(somePtr)	somePtr.readAnsiString()
Memory.readInt(somePtr)	somePtr.readInt()
Memory.writeUInt(somePtr)	somePtr.writeUInt()

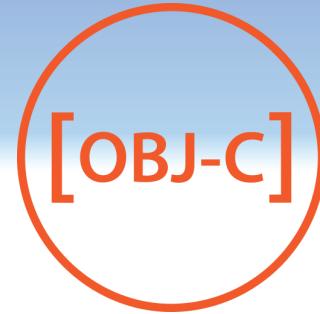
See also <https://frida.re/news/2025/05/17/frida-17-0-0-released/> and <https://frida.re/docs/javascript-api/>

Share Improve this answer Follow edited Jul 15 at 12:33 answered Jul 14 at 10:31

Robert 43.4k ● 18 ▪ 112 ▪ 178 Add a comment

Frida Basic Templates

Hooking Objective-C Functions Template



Target Class: -[NSTaggedPointerString isEqualToString:]

```
Interceptor.attach(ObjC.classes.NSTaggedPointerString['- isEqualToString:'].implementation, {
    onEnter: function (args) {
        var str = new ObjC.Object(ptr(args[2])).toString()
        console.log('[+] Hooked NSTaggedPointerString[- isEqualToString:] ->' , str);
    },
    onLeave: function (retval) {
        console.log('[+] Returned NSTaggedPointerString[- isEqualToString:] ->' , str);
    },
});
```

Target Class: -[NSDictionary objectForKey:]

```
Interceptor.attach(ObjC.classesNSDictionary['- objectForKey:'].implementation, {
    onEnter: function (args) {
        var str1 = new ObjC.Object(ptr(args[1])).toString();
        var str2 = new ObjC.Object(ptr(args[2])).toString();
        console.log('[+] Hooked -[NSDictionary objectForKey:] ->' , str1 + " " + str2);
    },
});
```

Frida Basic Templates

Hooking Swift Functions Template

<https://learnfrida.info/macos/#swiftstring>



```
const swiftFunctionPtr = Module.getExportByName(null, '$s7DVIA_v213DVIAUtilitiesC11loadWebView7withURL14viewControllerySS_So06UIViewJ0CtFZ');
console.log(swiftFunctionPtr)
Interceptor.attach(swiftFunctionPtr, {
    onEnter(args) {
        // Note that Swift String below <=15 Bytes can be printed like this
        const tmpAddr = args[1].and('0xFFFFFFFFFF').toString(16);
        const lsbAddr = ptr(new UInt64('0x' + tmpAddr));
        console.log(lsbAddr.add(32).readCString());

        //Longer Swift String > 15 Bytes require a different method to print.
    }
});
```

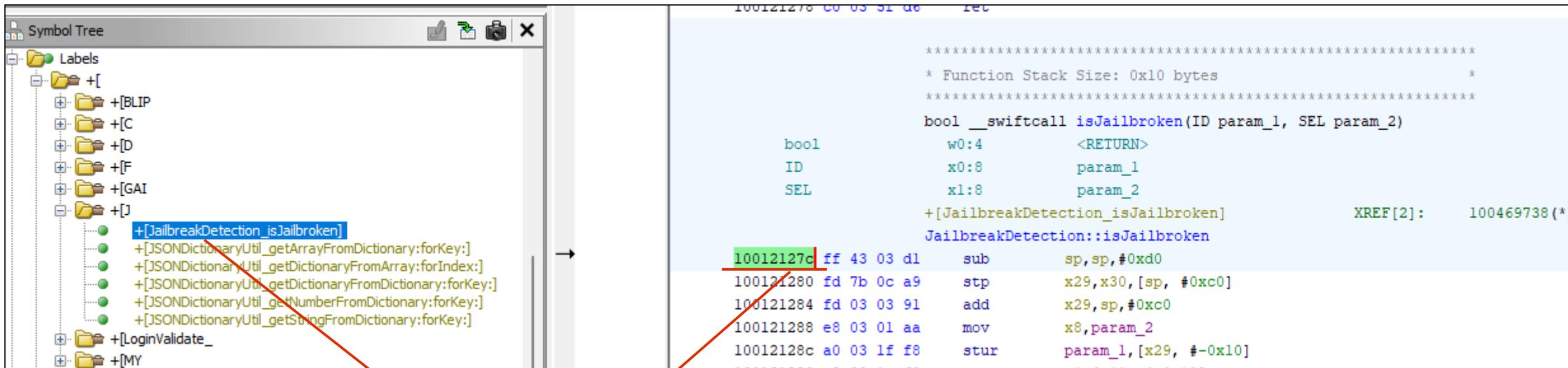
!

- ***Note:** For Swift strings, the logic for printing their values depends on the string length. If the length exceeds 15 bytes, a different approach is required compared to shorter strings.

Frida Basic Templates

Hooking Functions By Address Offset Template

Target Class: +[JailbreakDetection isJailbroken]



```
const objC_fun = ObjC.classes.JailbreakDetection['+ isJailbroken'].implementation
console.log ("Address Objective-C Syntax: " + objC_fun)
```

```
const app_base = Module.findBaseAddress("DVIA-v2");
const add_addr_fun = app_base.add('0x12127c')
console.log ("Address by Adding to Base: " + add_addr_fun)
```

Frida Basic Templates

Hooking Functions By Address Offset Template

Target Class: +[JailbreakDetection isJailbroken]

```
const objC_fun = ObjC.classes.JailbreakDetection['+ isJailbroken'].implementation
console.log ("Address Objective-C Syntax: " + objC_fun)

const app_base = Module.findBaseAddress("DVIAswiftv2");
const add_addr_fun = app_base.add('0x12127c')
console.log ("Address by Adding to Base: " + add_addr_fun)
```

Output:

```
Spawning `com.highlatitudehacks.DVIAswiftv2`...
Address Objective-C Syntax: 0x100ec927c
Address by Adding to Base: 0x100ec927c
```



Result in the same address pointer

Frida Basic Templates

Hooking Functions By Address Offset Template

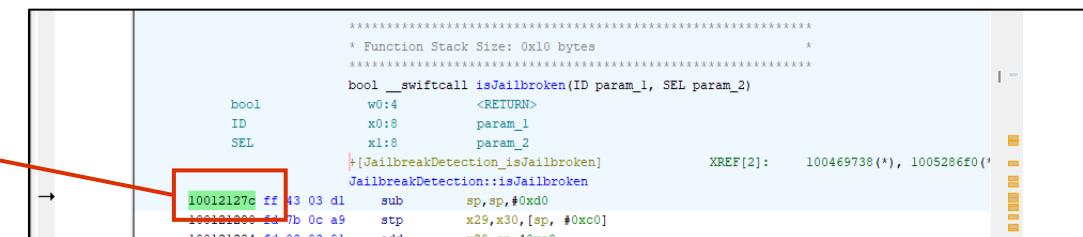
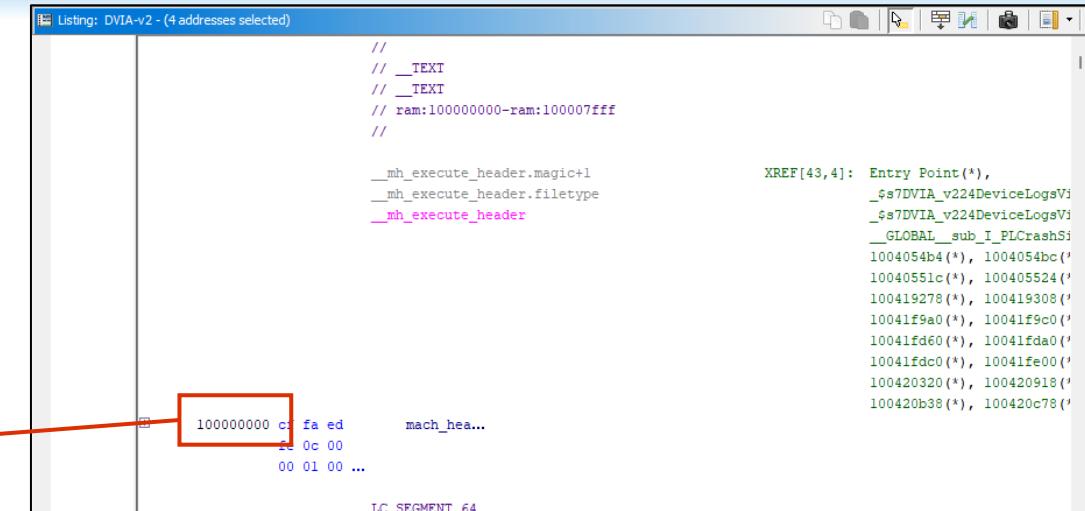
Target Class: +[JailbreakDetection isJailbroken]

Calculate Offset:

- Scroll to the top of the disassembly window.
- Record the base address.
- Record the function address shown.
- Offset = function address – base address

```
const app_base = Module.findBaseAddress("DVIA-v2");

Interceptor.attach(app_base.add('0x12127c'), {
    onEnter: function (args) {
        /*** Do your stuff here ***/
    },
    onLeave: function (retval) {
        /*** Do your stuff here ***/
    },
});
```



Frida Basic Templates

Hooking Functions By Address Offset Template

.dylib Libraries

```
const app_base = Module.findBaseAddress("target.dylib");

Interceptor.attach(app_base.add('0x12127c'), {
    onEnter: function (args) {
        /*** Do your stuff here ***/
    },
    onLeave: function (retval) {
        /*** Do your stuff here ***/
    },
});
```

.so Libraries

```
const app_base = Module.findBaseAddress("target.so");

Interceptor.attach(app_base.add('0x12127c'), {
    onEnter: function (args) {
        /*** Do your stuff here ***/
    },
    onLeave: function (retval) {
        /*** Do your stuff here ***/
    },
});
```

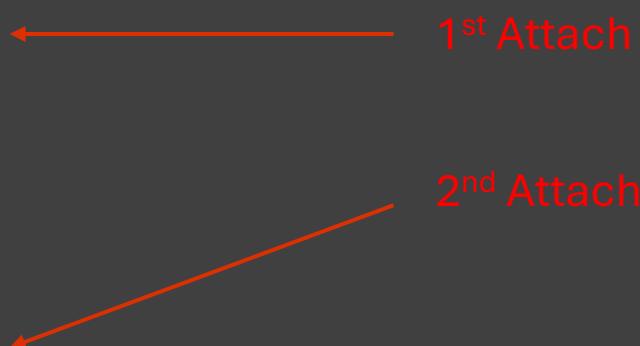
Frida Basic Templates

Early Hook Template (Trampoline)

<https://github.com/iddoeldor/frida-snippets/blob/master/README.md#hook-jni-by-address>

```
var moduleName = "libfoo.so";
var nativeFuncAddr = 0x1234;

Interceptor.attach(Module.findExportByName(null, "dlopen"), {
    onEnter: function(args) {
        this.lib = Memory.readUtf8String(args[0]);
        console.log("dlopen called with: " + this.lib);
    },
    onLeave: function(retval) {
        if (this.lib.endsWith(moduleName)) {
            console.log("ret: " + retval);
            var baseAddr = Module.findBaseAddress(moduleName);
            Interceptor.attach(baseAddr.add(nativeFuncAddr), {
                onEnter: function(args) {
                    console.log("[-] hook invoked");
                    console.log(JSON.stringify({
                        a1: args[1].toInt32(),
                        a2: Memory.readUtf8String(Memory.readPointer(args[2])),
                        a3: Boolean(args[3])
                    }, null, '\t'));
                }
            });
        }
    }
});
```



Frida Basic Templates

Overwriting a iOS Function

```

Decompile: decryptData:publicKey: -
1
2 /* WARNING: Unknown calling convention -- yet parameter storage is locked */
3 /* Function Stack Size: 0x20 bytes */
4
5 ID RSA::decryptData:publicKey:(ID param_1,SEL param_2,ID param_3,ID param_4)
6
7 {
8     long lVar1;
9     ID IVar2;
10    undefined8 uVar3;
11    undefined1 auVar4 [16];
12
13    auVar4 = _objc_retain(param_3);
14    lVar1 = auVar4._0_8_;
15    uVar3 = 0;
16    if ((lVar1 != 0) && (param_4 != 0)) {
17        auVar4 = FUN_000f4780(&objc::class_t::RSA,auVar4._0_8_,param_4);
18        if (auVar4._0_8_ == 0) {
19            uVar3 = 0;

```

ID is a pointer
SEL is a pointer

Method Parameters

```

var decryptData = eval('ObjC.classes.RSA["+ decryptData:publicKey:"']');

Interceptor.replace(decryptData.implementation, new NativeCallback(function(self1,self2,id1,id2) {
    console.warn('[decryptData:publicKey]: Replaced! DecryptedLicense for Device [' + deviceSerial + ']');
    console.log(ObjC.Object(id2).toString());
}

var strObj = nsstr(deviceSerial+ "1111111111111111" + "23000223000000");
var nsDataObj = nsstr2nsdata(strObj);

return ptr(nsDataObj);
}, 'pointer', ['pointer','pointer','pointer','pointer']));

```

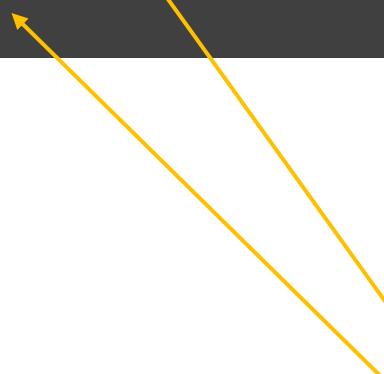
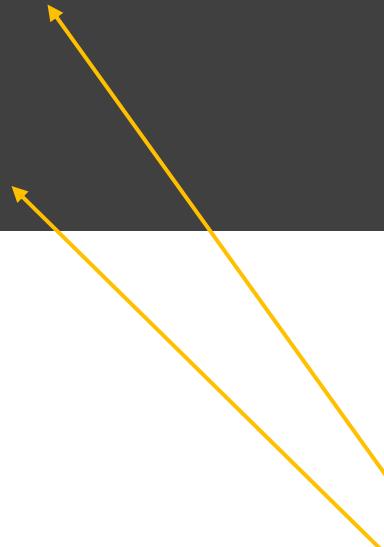
Method Parameters Type

Return type

Frida Basic Templates

Getting Argument and Return Types (Frida API)

```
[Apple iPad::com.highlatitudehacks.DVIAswiftv2 ]-> ObjC.classes.JailbreakDetection.isJailbroken
function
[Apple iPad::com.highlatitudehacks.DVIAswiftv2 ]-> ObjC.classes.JailbreakDetection.isJailbroken.argumentTypes
[
    "pointer",
    "pointer"
]
[Apple iPad::com.highlatitudehacks.DVIAswiftv2 ]-> ObjC.classes.JailbreakDetection.isJailbroken.returnType
"bool"
```



Frida API

Frida Basic Templates

Dynamic method invocation from Frida

```
var WebServerManager = ObjC.chooseSync(ObjC.classes.WebServerManager)[0];

function startWeb(){
    WebServerManager['- startWebServer']();
}

function stopWeb(){
    WebServerManager['- stopWebServer']();
}

startWeb();
stopWeb();
```

Invoke a libc function (<https://awakened1712.github.io/hacking/hacking-frida/>)

```
var openPtr = Module.findExportByName("libc.so", "open");
var open = new NativeFunction(openPtr, 'int', ['pointer', 'int']);
var fd = open(Memory.allocUtf8String('/tmp/test.txt'), 0);
```

Frida Snippets

Trigger a backtrace

```
console.warn(`\t[+] Backtrace:\n\t` + Thread.backtrace(this.context, Backtracer.ACCURATE).map(DebugSymbol.fromAddress).join(`\n\t`));
```

Provides additional visibility on application's execution calls.

Point of interest

```
[CFStringCreateWithBytes] len=5 enc=1536 preview="cydia"  
[+] Backtrace:  
0x183394e84 Foundation!-[NSString _newSubstringWithRange:zone:]  
0x181c659f0 CoreFoundation!_retainedComponentString  
0x181c77580 CoreFoundation!CFURLCopyScheme  
0x183395564 Foundation!-[NSURL(NSURL) scheme]  
0x1833c94b8 Foundation!-[NSURL(NSURL) description]  
0x183354304 Foundation!_NSDescriptionWithLocaleFunc  
0x181c2f968 CoreFoundation!__CFStringAppendFormatCore  
0x181c4d72c CoreFoundation!_CFStringCreateWithFormatAndArgumentsReturningMetadata  
0x181c6de08 CoreFoundation!_CFLogvEx2Predicate  
0x181c88ef8 CoreFoundation!_CFLogvEx3  
0x18337e550 Foundation!_NSLogv  
0x183376250 Foundation!NSLog  
0x184df9a0c UIKitCore!-[UIApplication _canOpenURL:publicURLsOnly:]  
0x184dfa57c UIKitCore!-[UIApplication canOpenURL:]  
0x10148822c DVIA-v2!JailbreakDetectionViewController.jailbreakTest5Tapped(_:) ← Point of interest  
0x101488860 DVIA-v2!objc JailbreakDetectionViewController.jailbreakTest5Tapped(_:) ← Trace up the call stack
```

Trace up the call stack

Frida Snippets

Hooking Objective-C String Compare Functions

```
function hookMethod(cls,meth) {
    try {
        const impl=cls[meth].implementation;
        Interceptor.attach(impl,{
            onEnter(args){
                try {
                    this.a=new ObjC.Object(args[0]).toString();
                    this.b=new ObjC.Object(args[2]).toString();
                    this.cls=cls.$className;
                    this.meth=meth;
                } catch (e) {console.error(e);}
            },
            onLeave(ret){
                try{
                    console.log(`[ObjC] -[${this.cls} ${this.meth}] a="${this.a}" b="${this.b}" -> ${ret}`);
                } catch (e) {console.error(e);}
            }
        });
        console.log('[+] hooked',cls.$className,meth);
    } catch (e) {console.error(e);}
}

const NSString = ObjC.classes.NSString;
const methods = ['- isEqualToString:', '- compare:', '- caseInsensitiveCompare:', '- localizedCompare:', '- localizedCaseInsensitiveCompare:', '- localizedStandardCompare:', '- hasPrefix:', '- hasSuffix:', '- containsString:', '- rangeOfString:'];
methods.forEach(meth=>
    hookMethod(NSString,meth)
);
console.log('[+] ObjC NSString hooks active.');
```

Frida Snippets

Hooking C String Compare Functions

```
function readCString(p){
    try{
        if (p.isNull()) { return 'NULL'; }
        return Memory.readUtf8String(p);
    } catch(_){
        try{
            const buf=Memory.readByteArray(p,32);
            return Array.from(new Uint8Array(buf)).map(b=>('0'+b.toString(16)).slice(-2)).join(' ');
        } catch(_){
            return '<unreadable>';
        }
    }
}

const exp_methods = ['strcmp','strncmp','strcasecmp','strncasecmp','memcmp']
exp_methods.forEach(meth=>{
    const p=Module.findExportByName(null,meth);
    if(p) {
        Interceptor.attach(p,{
            onEnter(args) {
                this.a=readCString(args[0]);
                this.b=readCString(args[1]);
                this.n=(meth.includes('ncmp')||meth==='memcmp')
            },
            onLeave(ret){
                console.log(`[C] ${meth}("${this.a}", "${this.b}"${this.n!==null?'n='+this.n:'}) -> ${ret}`);
            }
        }));
    }
});
console.log('[*] C string compare hooks active');
```

Frida Snippets

Hooking String Declaration Functions (NSString)

```
function hook(clsName, sel, argIndex){  
    try {  
        const cls=ObjC.classes[clsName];  
        if(!cls||!cls[sel]) return;  
        const impl=cls[sel].implementation;  
        Interceptor.attach(impl,{  
            onEnter(args){  
                this.tag=clsName+' '+sel;  
                try {  
                    if(sel.indexOf('Characters:length:')!==-1){  
                        const p=args[argIndex];  
                        this.msg='utf16="'+ Memory.readUtf16String(p);  
                    } else {  
                        const p=args[argIndex];  
                        this.msg='cstr="'+readCString(p);  
                    }  
                } catch(e){ this.msg=<parse err> + e; }  
            },  
            onLeave(ret){  
                try{  
                    const out=new ObjC.Object(ret).toString();  
                    console.log('ObjC',this.tag+' -> "'+out+'" [src '+this.msg+']);  
                } catch(_){  
                    console.log('ObjC',this.tag+' -> <non-objc return> [src '+this.msg+']);  
                }  
            }  
        });  
        console.log('[*] hooked',clsName,sel);  
    } catch(_){}  
}
```

```
hook('NSString','- initWithUTF8String:',2);  
hook('NSString','+ stringWithUTF8String:',2);  
hook('NSString','- initWithCString:encoding:',2);  
hook('NSString','+ stringWithCString:encoding:',2);  
hook('NSString','- initWithCharacters:length:',2);  
hook('NSString','+ stringWithCharacters:length:',2);  
hook('NSMutableString','- initWithUTF8String:',2);  
hook('NSMutableString','- initWithCString:encoding:',2);
```

Frida Snippets

Hooking Crypto Functions (Script 1 - libSystem.B.dylib)

Reference:

<https://github.com/federicodotta/Brida>

https://github.com/noobpk/frida-ios-hook/blob/master/frida-ios-hook/methods/intercept_crypto.js

```
Interceptor.attach(Module.findExportByName("libSystem.B.dylib", "CCCrypt"),
Interceptor.attach(Module.findExportByName("libSystem.B.dylib", "CCCryptorCreate"),
Interceptor.attach(Module.findExportByName("libSystem.B.dylib", "CCCryptorUpdate"),
Interceptor.attach(Module.findExportByName("libSystem.B.dylib", "CCCryptorFinal"),
Interceptor.attach(Module.findExportByName("libSystem.B.dylib", "CC_SHA1_Init"),
```

Frida Snippets

Hooking Crypto Functions (Script 1 - libSystem.B.dylib)

Reference:

<https://github.com/federicodotta/Brida>

https://github.com/noobpk/frida-ios-hook/blob/master/frida-ios-hook/methods/intercept_crypto.js

Output:

```
[+] Data out CCUpdate:  
eyJwcm9maWxlIjp7ImF1dG9RdwI0SW50ZXJ2YWwiOiI1IiwiYnRuTmV4dCI6Ik5leHQiLCJidG5PayI6Ik9LIIwiYnRuUXVpdCI6IlF1aXQiLCJjbGlwYm9h  
cmQiOjub25lIiwiY29tcGFueUlkIjoiMTAyNSIsImRhGFTeW5jQ2hLY2tJbnRlcNzhbCI6IjE0NDAiLCJkaXNhYmxlVGltZSI6IjYwIiwiZW5hYmxlZCI6  
IjAiLCJnZW9GZW5jZSI6IjAiLCJnZW9GZW5jZUxhdCI6IjEuMzUyMDgiLCJnZW9GZW5jZUxuZyI6IjEwMy44MiIsImdlb0ZlbmNlUmFkIjoiMTYwOTMuMCIs  
Im1vZGlmawVkjIjoiMjAyNC0xMS0yOCAwMToyMj01NyIsIm1zZ0FsZXJ0IjoiWW91ciBkZXZpY2UgYXBwZWfycyB0byBiZSBqYwlSvnJva2VuIG9yIHJvb3Rl  
ZC4gRm9yIHlvdXIgc2VjdXJpdHksIHdIHN0cm9uZ2x5IHJlY29tbWVuZCB0aGF0IHlvdSBkbyBub3QgdXNlIHRoaXMgZGV2aWNLIHBlcmZvcm0gYW55  
IGJhbmtpbmcgYWN0aXZpdGllcy5cclxuXHJcbkZyb20gTm92IDIwMjEsIHRoaXMgYXBwIHdpbGwgbm90IHdvcmsgb24gamFpbGJyb2tlbiBvcibyb290ZWQg  
ZGV2aWNlc4iLCJtc2dBcHBsaWNhdGlvkZpbGVzSW5RdwVzdGlvbii6I1lvdXIgYXBwbGljYXRpb24gYXBwZWfycyB0byBiZSBoYXZLIGJlZW4gdGftcGVy  
ZWQgd2l0aCwgYW5kIGNvdWxkIGJlIHVuZGVyIGF0dGFjay4gRm9yIHlvdXIgc2VjdXJpdHksIHBsZWfzZSBzS1kb3dubG9hZCB0aGUgYXBwbGljYXRpb24g  
ZnJvbSB0aGUgQXBwIFN0b3JlIC8gUGxheSBTdG9yZS4iLCJtc2dCbGt0ZXR3b3JrIjoiICIsIm1zZ0Rpc2FibGUI0iJuaGUgZm9sbG93aW5nIGNyaXRpY2Fs  
IHRocmVhdHMgaGF2ZSBiZWVuIGZvdW5kIG9uIHlvdXIgbW9iaWxlIGRldmljZS4gRm9yIHlvdXIgc2VjdXJpdHksIHRoaXMgYXBwbGljYXRpb24gaGFzIGJl  
ZW4gdGVtcG9yYXJpbHkgZGlzYWJsZWQuXHJcbiR7dGhyZWFOc30iLCJtc2dFbmFibGVMb2NhdGlvblNldHRpbmdzIjoiWW91IG1heSBvbx5IHVzZSB0aGlz  
IGFwcGxpY2F0aW9uIHdpdGhpbiB0aGUgYXV0aG9yaXNLZCBwcmVtaXNlc4gUGxlyXNlIGVuYWJsZSB0aGUgR1BTIGZvcIB0aGlzIGFwcGxpY2F0aW9uIHRv  
IHZlcmlmeSB5b3VyIGxvY2F0aW9uLiIsIm1zZ0V4cGlyeURhdGUI0iJZb3UgbWF5IG5vIGxvbmdlciB1c2UgdGhpcyBhcHBsaWNhdGlvbibhcyBpdCBoYXMg  
ZXhwaxJlZCBvbiAke2V4cGlyeV9kYXRlfS4iLCJtc2dGaW5hbE1lc3NhZ2Ui0iJGb3IgeW91ciBzZWN1cml0eSwgdGhpcyBhcHBsaWNhdGlvbib3aWxsIG5v  
dyBydwI0IiT<Im1zZ0d1b0Z1bmNlTjoiWW91TG1heSBvbx5IHVzZSB0aGlzIGFwcGxpY2F0aW9uTHdndGhpbiB0aGUgYXV0aG9vaXNlZCBwcmVtaXNlc4li
```

Frida Snippets

Hooking Crypto Functions (Script 2 - libcommonCrypto.dylib)

Reference:

<https://github.com/dpnishant/appmon/>

```
Interceptor.attach(Module.findExportByName("libcommonCrypto.dylib","CCHmac"),  
  
Interceptor.attach(Module.findExportByName("libcommonCrypto.dylib","CC_MD5"),  
  
Interceptor.attach(Module.findExportByName("libcommonCrypto.dylib","CC_SHA1"),
```

Output:

```
0200a2030447312d2da80302fa032038303136453635394535433734432354231464336413839434131333354136ba04053139393131c00400", "a  
rgSeq":2}, {"name": "MD5 Digest", "value": "9cc7[REDACTED]98a", "argSeq":3}]} } data: None  
message: {'type': 'send', 'payload': {'time': "2025-08-31T00:40:05.037Z", "txnType": "MD5 Hash", "lib": "libcommonCrypto.dylib", "method": "CC_MD5", "artifact": [{"name": "MD5 Data", "value": "08011a0e08fed6b488833312035f666920011a1408fed6b488833312045f666f74208090968982221a0f08e9e6badc822212045f6e706120001a0f0881e7badc822212045f6e746520011a1308f0d2e1cd9f3312045f73696420cdmmc5061a0f08a9e0037216636f6d2e6462732e73672e6465433542414532444441303039464441303434441443139352d434542432d3444351f80106800200880200a2030447312d2da80302fa032038303136453635394535433734432354231464336413839434131333354136ba04053139393131c00400", "a  
rgSeq":2}, {"name": "MD5 Digest", "value": "9cc7[REDACTED]98a", "argSeq":3}]} } data: None  
message: {'type': 'send', 'payload': {'time': "2025-08-31T00:40:05.048Z", "txnType": "MD5 Hash", "lib": "libcommonCrypto.dylib", "method": "CC_MD5", "artifact": [{"name": "MD5 Data", "value": "08011a0e08fed6b488833312035f666920011a1408fed6b488833312045f666f74208090968983331a0f08e9e6badc833312045f6e706120001a0f0881e7badc833312045f6c746520011a1308f0d2e1cd8f3312045f73696420cdmmc5061a0f08ae0037216636f6d2e6462732e73672e6465433542414532444441303039464441303434441443139352d434542432d3444351f80106800200880200a2030447312d2da80302fa032038303136453635394535433734432354231464336413839434131333354136ba04053139393131c00400", "a  
rgSeq":2}, {"name": "MD5 Digest", "value": "9cc7[REDACTED]98a", "argSeq":3}]} } data: None  
message: {'type': 'send', 'payload': {'time': "2025-08-31T00:40:05.057Z", "txnType": "MD5 Hash", "lib": "libcommonCrypto.dylib", "method": "CC_MD5", "artifact": [{"name": "MD5 Data", "value": "08011a0e08fed6b488833312035f666920011a1408fed6b488833312045f666f74208090968983331a0f08e9e6badc833312045f6e706120001a0f0881e7badc833312045f6c746520011a1308f0d2e1cd8f3312045f73696420cdmmc5061a0f08ae0037216636f6d2e6462732e73672e6465433542414532444441303039464441303434441443139352d434542432d3444351f80106800200880200a2030447312d2da80302fa032038303136453635394535433734432354231464336413839434131333354136ba04053139393131c00400", "a  
rgSeq":2}, {"name": "MD5 Digest", "value": "9cc7[REDACTED]98a", "argSeq":3}]} }
```

Frida Snippets

Hooking NSLog Functions

```
function nsstr(p) {
    try { return p.isNull() ? '(null)' : ObjC.Object(p).toString(); }
    catch (_) { return '(not NSString)'; }
}

const NSLog=Module.findExportByName('Foundation','NSLog');
if(NSLog){
    Interceptor.attach(NSLog,{
        onEnter(args){
            try{
                console.log('[NSLog]',nsstr(args[0]));
            }catch(e){
                //console.error(e);
            }
        }()
    );
    console.log('[*] hooked NSLog @',NSLog);
}
const NSLogv=Module.findExportByName('Foundation','NSLogv');
if(NSLogv){
    Interceptor.attach(NSLogv,{
        onEnter(args){
            try{
                console.log('[NSLogv]',nsstr(args[0]),'(va_list)');
            }catch(e){
                //console.error(e);
            }
        }()
    );
    console.log('[*] hooked NSLogv @',NSLogv);
}
```

Frida Snippets

Hooking OS Log Functions (Part 1) – Generated by LLM

```
function hookOSLog() {
    const os_log = Module.findExportByName(null, 'os_log');
    const os_log_with_type = Module.findExportByName(null, 'os_log_with_type');
    if (os_log) {
        Interceptor.attach(os_log, {
            onEnter(args) {
                console.log(`[os_log] ${readCString(args[1])}`);
            }
        });
        console.log('[+] Hooked os_log');
    }
    if (os_log_with_type) {
        Interceptor.attach(os_log_with_type, {
            onEnter(args) {
                const type = args[1].toInt32();
                console.log(`[os_log_with_type] type=${type} ${readCString(args[2])}`);
            }
        });
        console.log('[+] Hooked os_log_with_type');
    }
}
```

Continue on next slide.

Frida Snippets

Hooking OS Log Functions (Part 2) - Generated by LLM

```
const names = ['__os_log_impl', '_os_log_impl'];
for (const n of names) {
    const p = Module.findExportByName('libsystem_trace.dylib', n)
        || Module.findExportByName(null, n);
    if (!p) continue;
    Interceptor.attach(p, {
        onEnter(args) {
            const type = args[2].toInt32 ? args[2].toInt32() : 0;
            const fmt = args[3];
            console.log(`[${n}] type=${type} fmt="${readCString(fmt)}"`);
        }
    });
    console.log(`[+] Hooked ${n}`);
}

const os_log_create = Module.findExportByName(null, 'os_log_create');
if (os_log_create) {
    Interceptor.attach(os_log_create, {
        onEnter(args) {
            this.subsystem = readCString(args[0]);
            this.category = readCString(args[1]);
        },
        onLeave(ret) {
            console.log(`[os_log_create] handle=${ret} subsystem="${this.subsystem}" category="${this.category}"`);
        }
    });
    console.log(`[+] Hooked os_log_create`);
}
```

Frida Execution Examples

Generated Sample Outputs

```
[CFStringCreateWithBytes] len=14 enc=134217984 preview="/usr/sbin/sshd"
[CFStringCreateWithBytes] len=8 enc=134217984 preview="/etc/apt"
[CFStringCreateWithBytes] len=13 enc=134217984 preview="77008hboggfhvg"
[CFStringCreateWithBytes] len=29 enc=134217984 preview="file:///private/jailbreak.txt"
[CFStringCreateWithCString] enc=134217984 "/private/var/mobile/Containers/Data/Application/2883397D-5F2F-4084-A8DC-05987997AF45/tmp/"
[CFStringCreateWithBytes] len=96 enc=134217984 preview="file:///private/var/mobile/Containers/Data/Application/2883397D-5F2F-4084-A8DC-05987997AF45/tmp/"
[CFStringCreateWithBytes] len=22 enc=134217984 preview="/private/jailbreak.txt"
[CFStringCreateWithBytes] len=9 enc=134217984 preview="/private/"
[CFStringCreateWithBytes] len=22 enc=134217984 preview="/private/jailbreak.txt"
[CFStringCreateWithBytes] len=5 enc=1536 preview="cydia"
[CFStringCreateWithBytes] len=5 enc=1536 preview="cydia"
[CFStringCreateWithBytes] len=5 enc=1536 preview="cydia"
[CFStringCreateWithBytes] len=5 enc=1536 preview="cydia"
[CFStringCreateWithBytes] len=0 enc=134217984 preview=""
[CFStringCreateWithBytes] len=2 enc=134217984 preview="OK"
[CFStringCreateWithBytes] len=4 enc=1536 preview="self"
[CFStringCreateWithBytes] len=12 enc=1536 preview="_contentView"
[CFStringCreateWithBytes] len=4 enc=1536 preview="self"
[CFStringCreateWithBytes] len=12 enc=1536 preview="_contentView"
[NSString stringWithUTF8String:] "FA5CD3DA-6AD8-4224-967E-23626A8EFC53"
[CFStringCreateWithBytes] len=36 enc=134217984 preview="FA5CD3DA-6AD8-4224-967E-23626A8EFC53"
[CFStringCreateWithBytes] len=9 enc=1536 preview="leftInset"
[CFStringCreateWithBytes] len=11 enc=1536 preview="contentView"
[CFStringCreateWithBytes] len=10 enc=1536 preview="rightInset"
[CFStringCreateWithBytes] len=11 enc=1536 preview="contentView"
[CFStringCreateWithCString] enc=134217984 "com.apple.datadetectorscore.cache.urlifier"
[CFStringCreateWithBytes] len=118 enc=134217984 preview="file:///System/Library/PrivateFrameworks/DataDetectorsCore.framework/com.apple.datadetectorscore.cache.urlifier.system"
[CFStringCreateWithCString] enc=1536 "en-SG"
[CFStringCreateWithCString] enc=134217984 "com.apple.datadetectorscore.cache.urlifier"
[CFStringCreateWithBytes] len=118 enc=134217984 preview="file:///System/Library/PrivateFrameworks/DataDetectorsCore.framework/com.apple.datadetectorscore.cache.urlifier.system"
[CFStringCreateWithCString] enc=1536 "en-SG"
[CFStringCreateWithCString] enc=1536 "en"
```

Scanning Through Logs

Generated Sample Logs

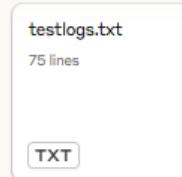
```
2025-08-08 16:33:03.471 PhotoStream[4521:88765] <Debug> [UserSettings] Loading user preferences from UserDefaults.  
2025-08-08 16:33:03.475 PhotoStream[4521:88765] <Info> [UserSettings] Preferences loaded: {notifications: true, theme: 'dark'}.  
2025-08-08 16:33:04.812 PhotoStream[4521:88765] <Default> [UI] User toggled notifications switch to: false.  
2025-08-08 16:33:04.818 PhotoStream[4521:88765] <Info> [UserSettings] Saving user preferences.  
2025-08-08 16:33:04.825 PhotoStream[4521:88765] <Debug> [PushNotifications] Unregistering for remote notifications.  
2025-08-08 16:33:04.950 PhotoStream[4521:88792] <Info> [Network] POST to /api/v2/settings successful.  
2025-08-08 16:33:06.111 PhotoStream[4521:88765] <Default> [UI] User tapped 'Log Out' button.  
2025-08-08 16:33:06.115 PhotoStream[4521:88765] <Info> [Authentication] Clearing user session and credentials.  
2025-08-08 16:33:06.120 PhotoStream[4521:88765] <Debug> [Keychain] Removing auth token from keychain.  
2025-08-08 16:33:06.128 PhotoStream[4521:88765] <Debug> [Database] Clearing user-specific cache.  
2025-08-08 16:33:06.135 PhotoStream[4521:88765] <Info> [UI] Presenting LoginViewController modally.  
2025-08-08 16:33:06.340 PhotoStream[4521:88765] <Debug> [UI] LoginViewController viewDidAppear.  
2025-09-09 16:33:10.954 PhotoStream[4521:88765] <Default> [UI] User tapped 'Login' button.  
2025-08-08 16:33:10.958 PhotoStream[4521:88765] <Info> [Authentication] Initiating login request.  
2025-08-08 16:33:10.965 PhotoStream[4521:88792] <Info> [Network] POST to /api/v2/login.  
2025-08-08 16:33:11.850 PhotoStream[4521:88792] <Info> [Network] Received 401 Unauthorized for /api/v2/login.  
2025-08-08 16:33:11.855 PhotoStream[4521:88765] <Debug> [Authentication] Failed for 'j.doe' , 'Wrongword123'.  
2025-08-08 16:33:15.301 PhotoStream[4521:88765] <Debug> [UI] Login error alert dismissed by user.  
2025-08-08 16:33:20.150 PhotoStream[4521:88765] <Default> [UI] User tapped 'Login' button.  
2025-08-08 16:33:20.153 PhotoStream[4521:88765] <Info> [Authentication] Initiating login request.  
2025-08-08 16:33:20.158 PhotoStream[4521:88792] <Info> [Network] POST to /api/v2/login.  
2025-08-08 16:33:20.888 PhotoStream[4521:88792] <Info> [Network] Received 200 OK for /api/v2/login.  
2025-08-08 16:33:20.895 PhotoStream[4521:88765] <Info> [Authentication] Login successful for user 'j.doe@example.com'.  
2025-08-08 16:33:20.901 PhotoStream[4521:88765] <Debug> [Keychain] Storing auth token.  
2025-08-08 16:33:20.915 PhotoStream[4521:88765] <Info> [UI] Dismissing LoginViewController.  
2025-08-08 16:33:21.241 PhotoStream[4521:88765] <Debug> [UI] MainFeedViewController  
2025-08-08 16:33:21.255 PhotoStream[4521:88792] <Info> [Network] Fetching main feed.  
2025-08-08 16:33:21.876 PhotoStream[4521:88792] <Info> [Network] Received 200 OK for /api/v2/mainfeed.  
2025-08-08 16:33:21.888 PhotoStream[4521:88765] <Debug> [UI] Reloading collection.  
2025-08-08 16:33:25.101 PhotoStream[4521:88765] <Default> [AppLifecycle] Application will resign active.  
2025-08-08 16:33:25.105 PhotoStream[4521:88765] <Default> [AppLifecycle] Application did enter background.  
2025-08-08 16:33:25.115 PhotoStream[4521:88765] <Info> [Database] Saving application context.  
2025-08-08 16:33:30.500 PhotoStream[4521:88765] <Default> [AppLifecycle] Application will enter foreground.  
2025-08-08 16:33:30.550 PhotoStream[4521:88765] <Default> [AppLifecycle] Application did become active.  
2025-08-08 16:33:30.558 PhotoStream[4521:88765] <Debug> [UI] Syncing UI state on foreground.  
2025-08-08 16:33:32.912 PhotoStream[4521:88765] <Fault> [Memory] Received low memory warning. Level: 1.  
2025-08-08 16:33:32.918 PhotoStream[4521:88765] <Info> [Cache] Clearing in-memory image cache due to memory pressure.
```

```
$ grep -rin pass testlogs.txt
```

```
$
```

Scanning Through Logs

Claude Sonnet 4



N scan for potential security issues such as sensitive data exposure in the attached iOS nslogs

I'll analyze the iOS log file for potential security issues and sensitive data exposure.

Analyzed data

View analysis ▾

Security Analysis Summary

I've identified several **critical security issues** in the iOS log file:

CRITICAL ISSUES

1. Password/Credential Exposure

- Line 46: Failed for 'j.doe' , 'Wrongword123' - This is a major security violation
 - The log explicitly contains a failed login attempt with both username (j.doe) and password (Wrongword123) in plaintext
 - This could allow anyone with log access to see user credentials

2. User Email Address Exposure

- Line 52: Login successful for user 'j.doe@example.com' - Exposes user's email address

HIGH RISK ISSUES

3. Authentication Token References

- Lines 38, 53: References to "auth token" operations
- While the actual token values aren't shown, these logs reveal authentication mechanisms

4. Username/User ID Exposure

- Line 25: 'testuser123' - Exposes internal username
- Various user interaction logs that could aid in user behavior analysis

MEDIUM RISK ISSUES

5. API Endpoint Disclosure

- Complete API structure exposed: /api/v2/feed, /api/v2/user/me, /api/v2/login, etc.
- Could help attackers understand the application architecture

6. Internal System Information

- CDN URLs: https://cdn.example.com/ structure
- Item IDs: 8A4F-23E1 format revelation

Recommendations

- Immediately sanitize logs - Remove any credential logging from authentication failures
- Implement log filtering - Mask sensitive data like emails, usernames, and passwords
- Review logging policies - Ensure no sensitive data is logged at any level
- Use log levels appropriately - Debug-level logs should never contain sensitive information
- Consider structured logging with automatic PII redaction
- Audit existing log storage for similar exposures

The most critical issue is the plaintext password logging in authentication failures - this needs immediate attention.

Blog Articles

<https://appledb.dev/firmware.html>
<https://www.romainthomas.fr/post/22-08-singpass-rasp-analysis/>
<https://learnfrida.info/macos/>
<https://github.com/iddoeldor/frida-snippets>
<https://awakened1712.github.io/hacking/hacking-frida/>
https://github.com/doronz88/swift_reversing
<https://github.com/swiftlang/swift/tree/main/docs/ABI>
https://github.com/ghidraninja/ghidra_scripts
<https://knight.sc/reverse%20engineering/2019/07/17/swift-metadata.html>
<https://github.com/federicodotta/Brida>
<https://github.com/dpnishant/appmon>

YouTube Reversing iOS

#OBTS v6.0: "iObfuscate Unraveling iOS Obfuscation Techniques" - Laurie Kirk
https://www.youtube.com/watch?v=5JuSp_7Xf1o

#OBTS v4.0: "iOS Reverse Engineering With Frida" - Christine Fossaceca
<https://www.youtube.com/watch?v=miSg0Km2V-w>

iOS Reverse Engineering :: Part I :: Dynamic Reversing and iOS Basics
<https://www.youtube.com/watch?v=h070-YZKOKE>

References



Contact Us



-  Email: hello@pastelops.io
-  GitHub Repo: <https://github.com/PastelOps/OWASPAppSec2025>

