

iOS Reverse Engineering

GovTech Brownbag



Dynamic Analysis > Deailing With Swift

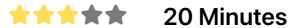
Deailing With Swift

By Max Chee

Developers may think that swift is more secured compared to objective C as there are limited tools used to reverse engineer application coded in swift. Explore how it is possible to modify swift applications.

DIFFICULTY

ESTIMATED TIME



Exercise Summary

- Execute class dump for swift apps
- Modify application behavior through runtime tampering tools
- Execute basic jailbreak bypass on Swift

Background

In this example, we are showing an open-sourced iOS application DVIA that is built using Swift. We will attempt to dump swift symbols and understand how are swift symbols structured and their difference from Objective-C.

Swift Symbols

Unlike Objective-C, we are not able to dump out the class headers for application hooking. Instead for Swift apps, we dump the symbol table which are then used to do application hooking.

Output for Symbol Table Dump using nm/dsdump

- 1 crazys-MacBook-Pro:DVIA-v2.app crazy
- 2 0x00100173d40 _\$s7DVIA_v232Jailbreak
- 3 0x0010037b138 _\$s7DVIA_v232Jailbreak
- 4 0x0010017496c _\$s7DVIA_v232Jailbreak
- 5 0x0010037b140 _\$s7DVIA_v232Jailbreak
- 6 0x001001721e4 _\$s7DVIA_v232Jailbreak
- 7 0x0010037b118 _\$s7DVIA_v232Jailbreak
- 8 0x00100172328 _\$s7DVIA_v232Jailbreak

Understanding Swift Symbols

The following is what will be seen when we dump the swift symbol table. It is very different from dumping Objective-C headers. We will disect the following dump to better understand swift symbols:

_\$s7DVIA_v232JailbreakDetectionVie wControllerC12isJailbrokenSbyF

Application Name

_\$<u>s7DVIA_v2</u>32JailbreakDetectionVie wControllerC12isJailbrokenSbyF

- s7DVIA_v2 can be broken into 3 parts,
 - o s Indicate Swift Stable Mangling
 - o 7 App Name Length
 - o DVIA_v2 App Name

Module Name

_\$s7DVIA_v2**32JailbreakDetectionVie** wControllerC12isJailbrokenSbyF

- 32JailbreakDetectionViewController can be broken into 2 parts,
 - o 32 Module Name Length
 - JailbreakDetectionViewController
 - Module Name

Module Name

_\$s7DVIA_v232JailbreakDetectionVie wController**C12isJailbroken**SbyF

• C12isJailbroken - can be broken into 3 parts,

Creating Theos Tweak

Now that we have a better understanding towards Swift symbols, we will attemp to create a theos tweak to modify application that is written using Swift.

Theos Jailbreak Evasion Script for Jailbreak Test 1

1 %hook ViewController 2 static Boolean (*orig 3 4 Boolean hook_ViewCon+ 5 NSLog(@"We have I 6 return false; 7 8 %end 9 10 %ctor { %init(ViewController 11 12 MSHookFunction(MSFind 13 (void*)hook_View (void**)&orig_Vic 14 15 }

We have included the application on the iPhone provided. To view the difference in application before and after the tweak, please run the application first (DVIA-v2) before clicking the tweak me button!

Jailbreak Implementation for Jailbreak Test 1 in DVIA-v2

```
func isJailbroken() -> Bo
1
2
       if FileManager.defau
3
           return true
4
5
       else if FileManager.
6
           return true
7
8
       else if FileManager.
9
           return true
10
11
       else if FileManager.
12
           return true
13
14
       else if FileManager.
15
           return true
16
17
       //All checks have fa
           return false
18
19 }
```

Tweak Me!