iOS Reverse Engineering

GovTech Brownbag

Deailing With Swift

By Max Chee

Developers may think that swift is more secured compared to objectivc 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
★★★☆☆	**20 Minutes**

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

```
crazys-MacBook-Pro:DVIA-v2.app crazy
0x00100173d40 _$s7DVIA_v232Jailbreak
0x0010037b138 _$s7DVIA_v232Jailbreak
0x0010017496c _$s7DVIA_v232Jailbreak
0x0010037b140 _$s7DVIA_v232Jailbreak
0x001001721e4 _$s7DVIA_v232Jailbreak
0x0010037b118 _$s7DVIA_v232Jailbreak
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_v232JailbreakDetectionViewControllerC12isJailbrokenSbyF
```

Application Name

_$**s7DVIA_v2**32JailbreakDetectionViewControllerC12isJailbrokenSbyF

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

Module Name

_$s7DVIA_v2**32JailbreakDetectionViewController**C12isJailbrokenSbyF

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

Module Name

_$s7DVIA_v232JailbreakDetectionViewController**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

```
%hook ViewController
    static Boolean (*ori

    Boolean hook_ViewCon
        NSLog(@“We have 
        return false;
    }
%end

%ctor {
    %init(ViewController
    MSHookFunction(MSFin
        (void*)hook_View
        (void**)&orig_Vi
}
```

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!

Tweak Me!

Jailbreak Implementation for Jailbreak Test 1 in DVIA-v2

```
func isJailbroken() -> B
    if FileManager.defau
        return true
    }
    else if FileManager.
        return true
    }
    else if FileManager.
        return true
    }
    else if FileManager.
        return true
    }
    else if FileManager.
        return true
    }
    //All checks have fa
        return false
}
```