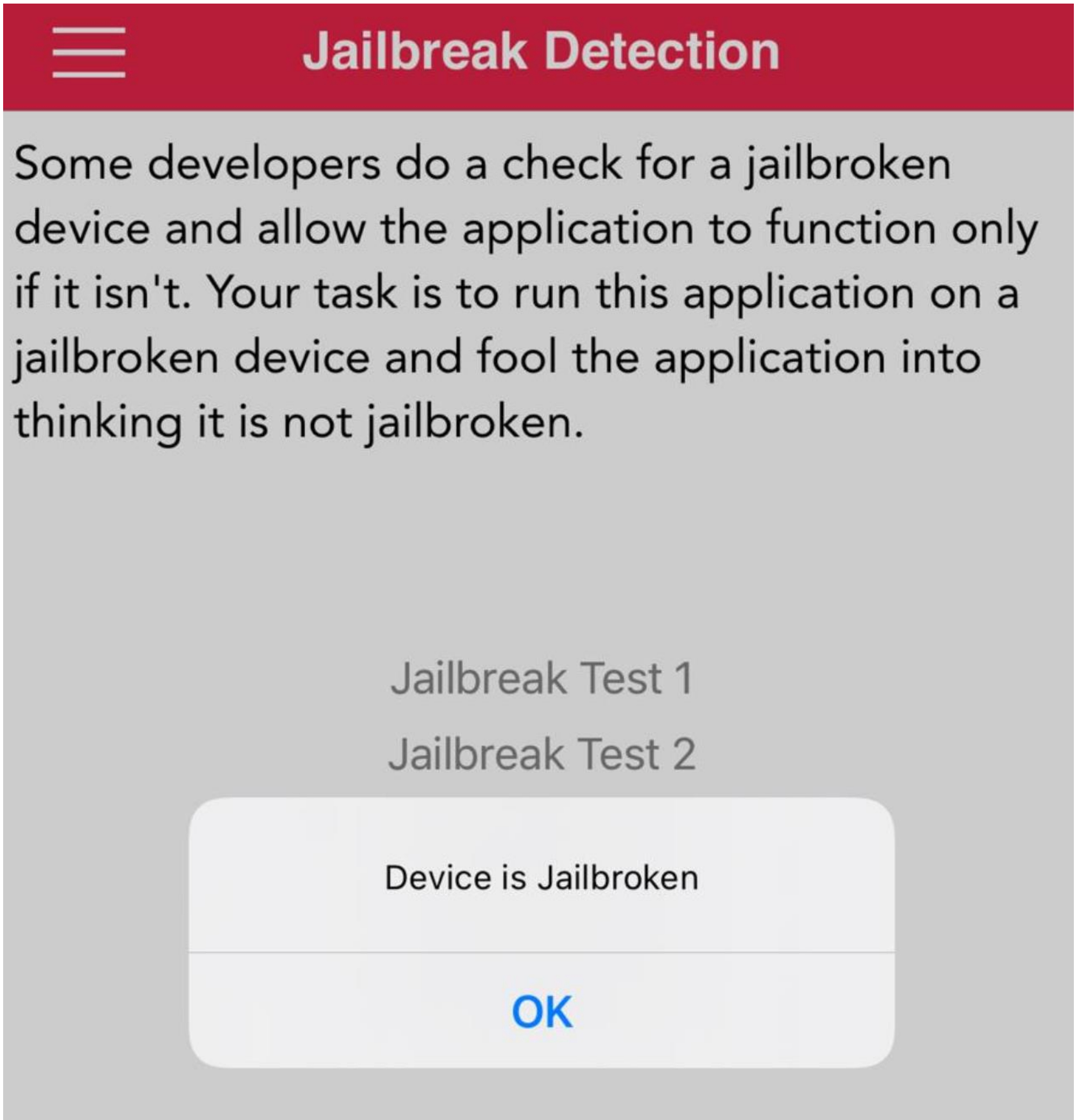
DVIA-V2

Jailbreak Test 1

It begins by searching for the string “Device is Jailbroken” as shown from the error message on the app itself.

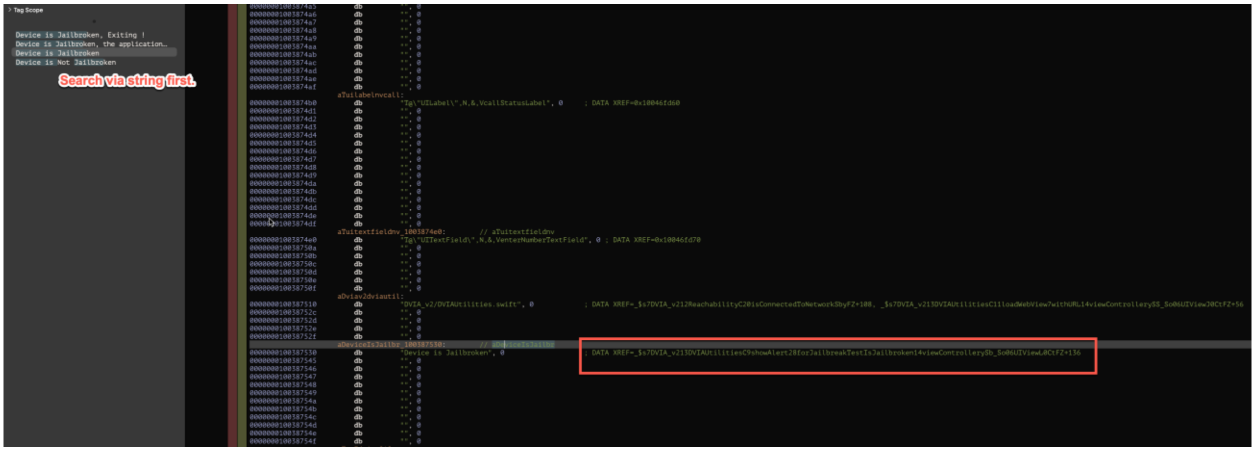


Jailbreak Detection

Some developers do a check for a jailbroken device and allow the application to function only if it isn't. Your task is to run this application on a jailbroken device and fool the application into thinking it is not jailbroken.

Device is Jailbroken

Subsequently, the app needs to be disassembled, and the relevant string(s) need to be searched.



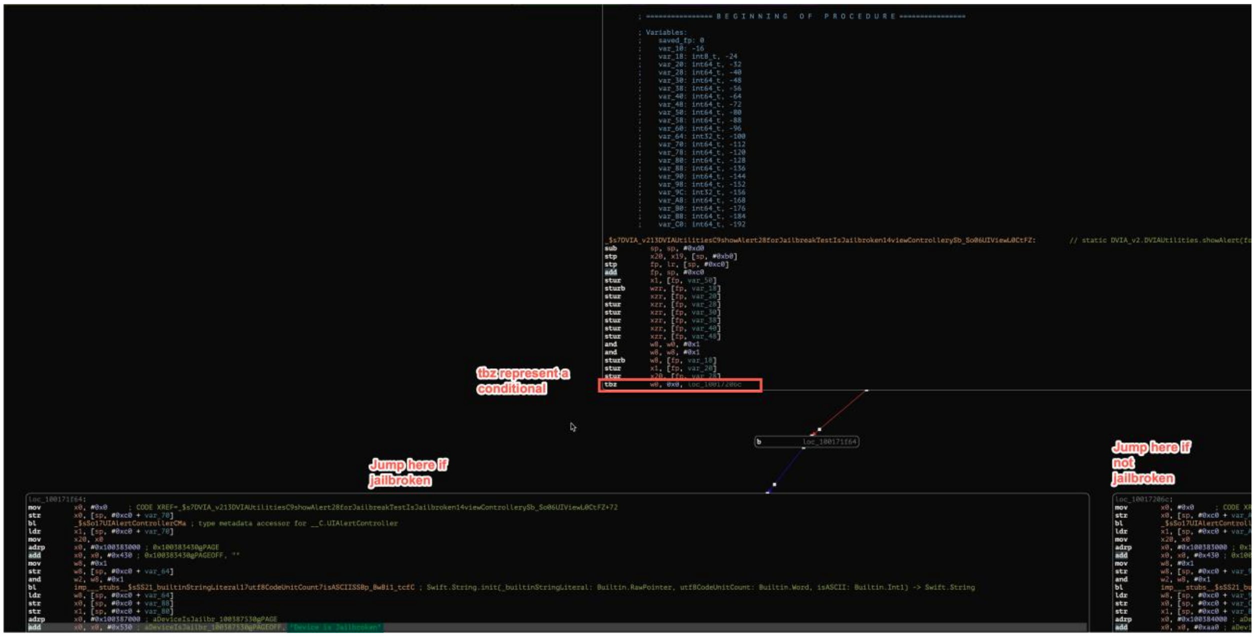
```
Device is Jailbroken, Exiting !
Device is Jailbroken, the application…
Device is Jailbroken
Device is Not Jailbroken
```

```
00000001003874a5        db        "", 0
00000001003874a6        db        "", 0
00000001003874a7        db        "", 0
00000001003874a8        db        "", 0
00000001003874a9        db        "", 0
00000001003874aa        db        "", 0
00000001003874ab        db        "", 0
00000001003874ac        db        "", 0
00000001003874ad        db        "", 0
00000001003874ae        db        "", 0
00000001003874af        db        "", 0
                  aTuilabelnvcall:
00000001003874b0        db        "T@\"UILabel\",N,&,VcallStatusLabel", 0      ; DATA XREF=0x10046fd60
00000001003874d1        db        "", 0
00000001003874d2        db        "", 0
00000001003874d3        db        "", 0
00000001003874d4        db        "", 0
00000001003874d5        db        "", 0
00000001003874d6        db        "", 0
00000001003874d7        db        "", 0
00000001003874d8        db        "", 0
00000001003874d9        db        "", 0
00000001003874da        db        "", 0
00000001003874db        db        "", 0
00000001003874dc        db        "", 0
00000001003874dd        db        "", 0
00000001003874de        db        "", 0
00000001003874df        db        "", 0
                  aTuitextfieldnv_1003874e0:        // aTuitextfieldnv
00000001003874e0        db        "T@\"UITextField\",N,&,VenterNumberTextField", 0 ; DATA XREF=0x10046fd70
000000010038750a        db        "", 0
000000010038750b        db        "", 0
000000010038750c        db        "", 0
000000010038750d        db        "", 0
000000010038750e        db        "", 0
000000010038750f        db        "", 0
                  aDviav2dviautil:
0000000100387510        db        "DVIA_v2/DVIAUtilities.swift", 0            ; DATA XREF=_$s7DVIA_v212ReachabilityC20isConnectedToNetworkSbyFZ+108, _$s7DVIA_v213DVIAUtilitiesC11loadWebView7withURL14viewControllerySS_So06UIViewJ0CtFZ+56
000000010038752c        db        "", 0
000000010038752d        db        "", 0
000000010038752e        db        "", 0
000000010038752f        db        "", 0
                  aDeviceIsJailbr_100387530:        // aDeviceIsJailbr
0000000100387530        db        "Device is Jailbroken", 0                  ; DATA XREF=_$s7DVIA_v213DVIAUtilitiesC9showAlert28forJailbreakTestIsJailbroken14viewControllerySb_So06UIViewL0CtFZ+136
0000000100387545        db        "", 0
0000000100387546        db        "", 0
0000000100387547        db        "", 0
0000000100387548        db        "", 0
0000000100387549        db        "", 0
000000010038754a        db        "", 0
000000010038754b        db        "", 0
000000010038754c        db        "", 0
000000010038754d        db        "", 0
000000010038754e        db        "", 0
000000010038754f        db        "", 0
```

By zeroing on the highlighted swift method and changing the view to code flow graph. There is a preceding code that determines whether a device is jailbroken.

This is the most important code here in arm assembly.

tbz w0, 0x0, loc_10017206c



```
; ================ B E G I N N I N G   O F   P R O C E D U R E ================

; Variables:
;    saved_fp: 0
;    var_10: -16
;    var_18: int8_t, -24
;    var_20: int64_t, -32
;    var_28: int64_t, -40
;    var_30: int64_t, -48
;    var_38: int64_t, -56
;    var_40: int64_t, -64
;    var_48: int64_t, -72
;    var_50: int64_t, -80
;    var_58: int64_t, -88
;    var_60: int64_t, -96
;    var_64: int32_t, -100
;    var_70: int64_t, -112
;    var_78: int64_t, -120
;    var_80: int64_t, -128
;    var_88: int64_t, -136
;    var_90: int64_t, -144
;    var_98: int64_t, -152
;    var_9C: int32_t, -156
;    var_A8: int64_t, -168
;    var_B0: int64_t, -176
;    var_B8: int64_t, -184
;    var_C0: int64_t, -192

_$s7DVIA_v213DVIAUtilitiesC9showAlert28forJailbreakTestIsJailbroken14viewControllerySb_So06UIViewL0CtFZ:       // static DVIA_v2.DVIAUtilities.showAlert(fo
sub        sp, sp, #0xd0
stp        x20, x19, [sp, #0xb0]
stp        fp, lr, [sp, #0xc0]
add        fp, sp, #0xc0
stur       x1, [fp, var_50]
sturb      wzr, [fp, var_18]
stur       xzr, [fp, var_20]
stur       xzr, [fp, var_28]
stur       xzr, [fp, var_30]
stur       xzr, [fp, var_38]
stur       xzr, [fp, var_40]
stur       xzr, [fp, var_48]
and        w8, w0, #0x1
and        w8, w8, #0x1
sturb      w8, [fp, var_18]
stur       x1, [fp, var_20]
stur       x20, [fp, var_28]
tbz        w0, 0x0, loc_10017206c
```

```
b          loc_100171f64
```

```
loc_100171f64:
mov        x0, #0x0      ; CODE XREF=_$s7DVIA_v213DVIAUtilitiesC9showAlert28forJailbreakTestIsJailbroken14viewControllerySb_So06UIViewL0CtFZ+72
str        x0, [sp, #0xc0 + var_70]
bl         _$sSo17UIAlertControllerCMa ; type metadata accessor for __C.UIAlertController
ldr        x1, [sp, #0xc0 + var_70]
mov        x20, x0
adrp       x0, #0x100383000 ; 0x100383430@PAGE
add        x0, x0, #0x430 ; 0x100383430@PAGEOFF, ""
mov        w8, #0x1
str        w8, [sp, #0xc0 + var_64]
and        w2, w8, #0x1
bl         imp___stubs__$sSS21_builtinStringLiteral17utf8CodeUnitCount7isASCIISSBp_BwBi1_tcfC ; Swift.String.init(_builtinStringLiteral: Builtin.RawPointer, utf8CodeUnitCount: Builtin.Word, isASCII: Builtin.Int1) -> Swift.String
ldr        w8, [sp, #0xc0 + var_64]
str        x0, [sp, #0xc0 + var_88]
str        x1, [sp, #0xc0 + var_80]
adrp       x0, #0x100387000 ; aDeviceIsJailbr_100387530@PAGE
add        x0, x0, #0x530 ; aDeviceIsJailbr_100387530@PAGEOFF, "Device is Jailbroken"
```

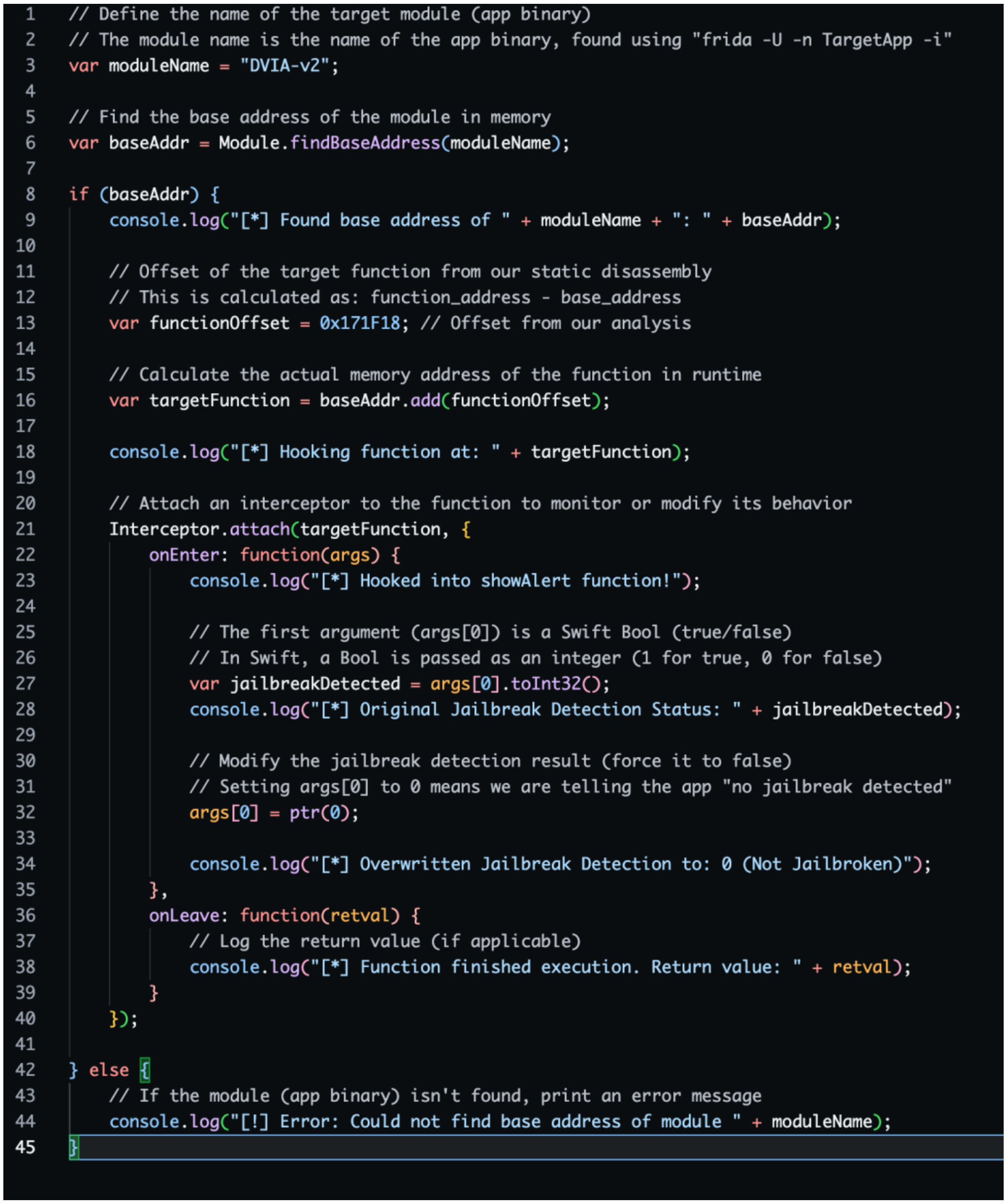
```
loc_10017206c:
mov        x0, #0x0      ; CODE XR
str        x0, [sp, #0xc0 + var_A
bl         _$sSo17UIAlertControll
ldr        x1, [sp, #0xc0 + var_A
mov        x20, x0
adrp       x0, #0x100383000 ; 0x1
add        x0, x0, #0x430 ; 0x100
mov        w8, #0x1
str        w8, [sp, #0xc0 + var_9
and        w2, w8, #0x1
bl         imp___stubs__$sSS21_bu
ldr        w8, [sp, #0xc0 + var_9
str        x0, [sp, #0xc0 + var_C
str        x1, [sp, #0xc0 + var_B
adrp       x0, #0x100384000 ; aDe
add        x0, x0, #0xaa0 ; aDevi
```

There are two ways to hijack this.

A. You find the offset for "tbz" and modify the register values.

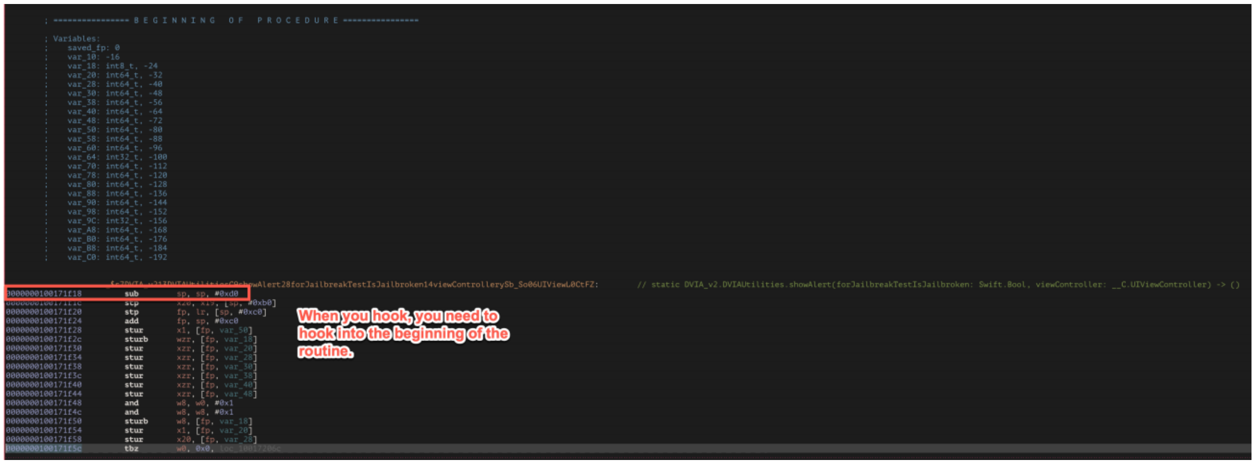
B. You just hook into the affected swift method and tamper with the method argument so that it appears to the app that the device is not jailbroken.

Enlisting GPT's help to draft up Frida's code. Here it is: ￼



```
// Define the name of the target module (app binary)
// The module name is the name of the app binary, found using "frida -U -n TargetApp -i"
var moduleName = "DVIA-v2";

// Find the base address of the module in memory
var baseAddr = Module.findBaseAddress(moduleName);

if (baseAddr) {
    console.log("[*] Found base address of " + moduleName + ": " + baseAddr);

    // Offset of the target function from our static disassembly
    // This is calculated as: function_address - base_address
    var functionOffset = 0x171F18; // Offset from our analysis

    // Calculate the actual memory address of the function in runtime
    var targetFunction = baseAddr.add(functionOffset);

    console.log("[*] Hooking function at: " + targetFunction);

    // Attach an interceptor to the function to monitor or modify its behavior
    Interceptor.attach(targetFunction, {
        onEnter: function(args) {
            console.log("[*] Hooked into showAlert function!");

            // The first argument (args[0]) is a Swift Bool (true/false)
            // In Swift, a Bool is passed as an integer (1 for true, 0 for false)
            var jailbreakDetected = args[0].toInt32();
            console.log("[*] Original Jailbreak Detection Status: " + jailbreakDetected);

            // Modify the jailbreak detection result (force it to false)
            // Setting args[0] to 0 means we are telling the app "no jailbreak detected"
            args[0] = ptr(0);

            console.log("[*] Overwritten Jailbreak Detection to: 0 (Not Jailbroken)");
        },
        onLeave: function(retval) {
            // Log the return value (if applicable)
            console.log("[*] Function finished execution. Return value: " + retval);
        }
    });

} else {
    // If the module (app binary) isn't found, print an error message
    console.log("[!] Error: Could not find base address of module " + moduleName);
}
```

How the offset is found is via hooking to the first instruction of the affected method. As ASLR is enabled, it means that on every program runtime, address will change so you need offset(s) to get the real address of the method during runtime.



```
        ; ================ B E G I N N I N G   O F   P R O C E D U R E ================

        ; Variables:
        ;    saved_fp: 0
        ;    var_10: -16
        ;    var_18: int8_t, -24
        ;    var_20: int64_t, -32
        ;    var_28: int64_t, -40
        ;    var_30: int64_t, -48
        ;    var_38: int64_t, -56
        ;    var_40: int64_t, -64
        ;    var_48: int64_t, -72
        ;    var_50: int64_t, -80
        ;    var_58: int64_t, -88
        ;    var_60: int64_t, -96
        ;    var_64: int32_t, -100
        ;    var_70: int64_t, -112
        ;    var_78: int64_t, -120
        ;    var_80: int64_t, -128
        ;    var_88: int64_t, -136
        ;    var_90: int64_t, -144
        ;    var_98: int64_t, -152
        ;    var_9C: int32_t, -156
        ;    var_A8: int64_t, -168
        ;    var_B0: int64_t, -176
        ;    var_B8: int64_t, -184
        ;    var_C0: int64_t, -192


                     $s7DVIA_v213DVIAUtilitiesC9showAlert28forJailbreakTestIsJailbroken14viewControllerySb_So06UIViewL0CtFZ:        // static DVIA_v2.DVIAUtilities.showAlert(forJailbreakTestIsJailbroken: Swift.Bool, viewController: __C.UIViewController) -> ()
0000000100171f18         sub        sp, sp, #0xd0
0000000100171f1c         stp        x20, x19, [sp, #0xb0]
0000000100171f20         stp        fp, lr, [sp, #0xc0]
0000000100171f24         add        fp, sp, #0xc0
0000000100171f28         stur       x1, [fp, var_50]
0000000100171f2c         sturb      wzr, [fp, var_18]
0000000100171f30         stur       xzr, [fp, var_20]
0000000100171f34         stur       xzr, [fp, var_28]
0000000100171f38         stur       xzr, [fp, var_30]
0000000100171f3c         stur       xzr, [fp, var_38]
0000000100171f40         stur       xzr, [fp, var_40]
0000000100171f44         stur       xzr, [fp, var_48]
0000000100171f48         and        w8, w0, #0x1
0000000100171f4c         and        w8, w8, #0x1
0000000100171f50         sturb      w8, [fp, var_18]
0000000100171f54         stur       x1, [fp, var_20]
0000000100171f58         stur       x20, [fp, var_28]
0000000100171f5c         tbz        w0, 0x0, loc_10017206c
```

When you hook, you need to hook into the beginning of the routine.