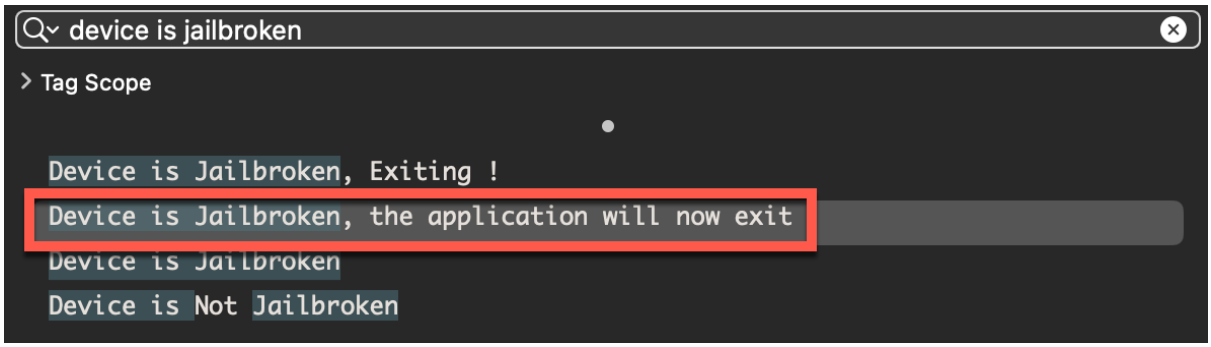
Methodology for Jailbreak 3

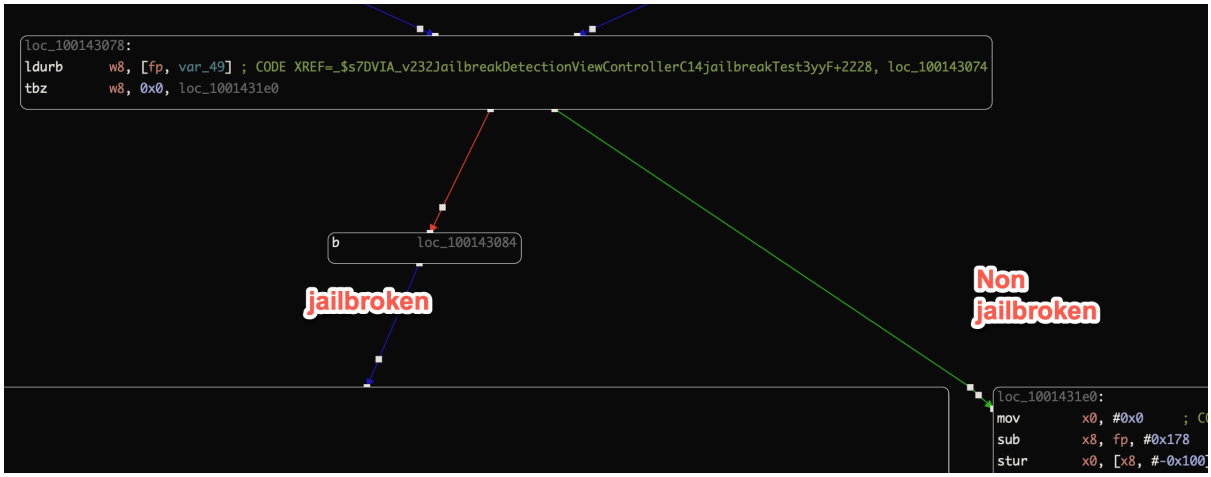
Searching using the highlighted string



Click xref

So, you need to focus on the non-jailbroken path

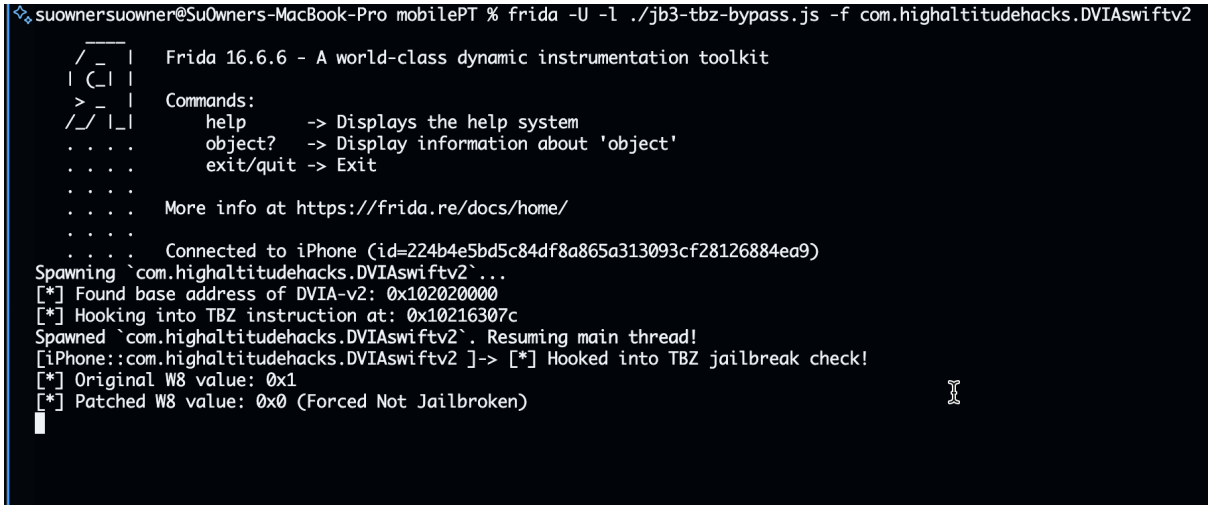
High level overview



Use Frida code to modify bit 0 of w8 to 0

```
/*
    🔍 Understanding the Jailbreak Check Mechanism

    1️⃣ The function loads a value into register `w8`:
       0000000100143078    ldurb      w8, [fp, var_49]  ; Load a byte into w8 (likely a jailbreak flag)

    2️⃣ The `TBZ` instruction checks **bit 0** of `w8`:
       000000010014307c    tbz        w8, 0x0, loc_1001431e0

       - If `bit 0 == 0` → Branch to `0x1431E0` (🚀 "Not Jailbroken" path)
       - Otherwise → Continue execution to `0x143084` (💀 "Jailbroken" path)

    3️⃣ If the check fails, execution jumps to `0x143084`:
       0000000100143080    b          loc_100143084  ; Default path leads to jailbreak detection

       - 🔴 `0x143084` → Displays "Device is Jailbroken" and **exits the app**.
       - 🟢 `0x1431E0` → Displays "Device is Not Jailbroken" and **continues normally**.

    🎯 Our Goal: **Modify `w8` so bit 0 is always `0`**, forcing the app to **always branch to `0x1431E0`**.
*/

// Define the target module (app binary)
var moduleName = "DVIA-v2";

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

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

    // 🛠️ Offset of the TBZ instruction from static disassembly
    var tbzOffset = 0x14307C; // TBZ instruction at 0x14307C

    // 🎯 Calculate the actual memory address at runtime (handling ASLR)
    var tbzInstruction = baseAddr.add(tbzOffset);

    console.log("[*] Hooking into TBZ instruction at: " + tbzInstruction);

    // 🔥 Attach an interceptor at the TBZ instruction
    Interceptor.attach(tbzInstruction, {
        onEnter: function(args) {
            console.log("[*] Hooked into TBZ jailbreak check!");

            // 🧐 Log the original value of W8 before modification
            console.log("[*] Original W8 value: " + this.context.x8);

            // 🛠️ Force bit 0 of W8 to 0, ensuring the TBZ instruction **always branches to 0x1431E0**
            this.context.x8 &= ~1;

            // ✅ Log the new W8 value after patching
            console.log("[*] Patched W8 value: " + this.context.x8 + " (Forced Not Jailbroken)");
        }
    });

} else {
    console.log("[!] Error: Could not find base address of module " + moduleName);
}
```

Output


suownersuowner@SuOwners-MacBook-Pro mobilePT % frida -U -l ./jb3-tbz-bypass.js -f com.highaltitudehacks.DVIAswiftv2
Frida 16.6.6 - A world-class dynamic instrumentation toolkit
Commands:
help -> Displays the help system
object? -> Display information about 'object'
exit/quit -> Exit
More info at https://frida.re/docs/home/
Connected to iPhone (id=224b4e5bd5c84df8a865a313093cf28126884ea9)
Spawning `com.highaltitudehacks.DVIAswiftv2`...
[*] Found base address of DVIA-v2: 0x102020000
[*] Hooking into TBZ instruction at: 0x10216307c
Spawned `com.highaltitudehacks.DVIAswiftv2`. Resuming main thread!
[iPhone::com.highaltitudehacks.DVIAswiftv2]-> [*] Hooked into TBZ jailbreak check!
[*] Original W8 value: 0x1
[*] Patched W8 value: 0x0 (Forced Not Jailbroken)