

Week 14

Mobile Application Hacking

Pete




Announcements

- Last meeting!
- Due to MetaCTF, no challenges



sigpwny{not_so_free_gems}



Clash of Clans Gems Generator



Last update: Monday, 29 Nov 2021
Online Users: 506
Gems Generated: 129,954,254

Type your username

Select your platform:

Select the amount of Gems (max.250,000 daily):





 10,000 Gems	 50,000 Gems	 100,000 Gems	 250,000 Gems
---	---	--	--



Table of Contents

- What is mobile app hacking?
- Mobile app tooling
- Why Android Reversing?
- Facets of Android Reversing
 - Unpacking & Repacking Applications
 - User Data Modifications
 - API Sniffing / Certificate Pinning
- Challenge Walkthrough



What is mobile app hacking?

- Given a compiled application, perform some modification in order to gain information
 - User personal information
 - Modify high-scores / in-app currency
 - Expose internal APIs
 - Circumvent security checks (i.e. PINs / license key checks)
- Apps are turned from source code (Objective-C or swift on iOS / Java or Kotlin on android) into app files
 - frameworks like flutter and react native are slightly different... (more on this later)
 - essentially a zip file
 - iOS - .ipa
 - android - .apk (~~literally a zip file~~)





Mobile app tooling



IDA / Ghidra

- Static analysis of program flow



Frida (covered in-depth later)

- Closest thing to a debugger - Hook into running app and modify it (like CheatEngine)
- Objection - Builds on top of frida
- MobSF - Builds on top of frida



Android Static Analysis

- jadx / dex2jar
 - apk to .java decompilation / resources
- apktool
 - apk to .smali / resources & repacking



Compiled Java Analysis

- Bytecode Viewer
 - .class -> .java decompilation using multiple different tools
- JD-GUI



Why Start with Android Reversing?

1. Improved Tooling
 - a. Better tooling makes it much easier to reverse
 - b. apk -> java source code helps immensely
2. Better resources
 - a. A lot more information is available online
3. Application reuse
 - a. Most apps are released on both iOS and Android
 - b. Similar code base = only have to reverse Android



Android Static Reversing Difficulty Scale

Program was written in Java or Kotlin

- Use JADX or (APKTool+java decompiler) to recover original .java files, analyze

Program was written in React Native

- Use APKTool to dump files, locate /assets/index.android.bundle (the bundled JS file)

Program was written with Native Libs

- Use APKTool to dump files
- Use JADX to figure out what native libraries were loaded (System.loadLibrary or System.Load)
- Use Ghidra to analyze native libraries

Program was written in Unity

- Use APKTool to dump files, locate compiled unity game (libil2cpp.so), decompile into DLLs with Il2CppDumper, analyze in dnSpy or Ghidra

Program was written with Flutter

- Cry because flutter is executed with the Dart VM, making it very hard to analyze (see the RE chall hell for why VMs are hard)
- Tools such as [Doldrums](#), [reFlutter](#), and [darter](#) can identify function signatures in libapp.so



User Data Modifications

Hacking crossy road!

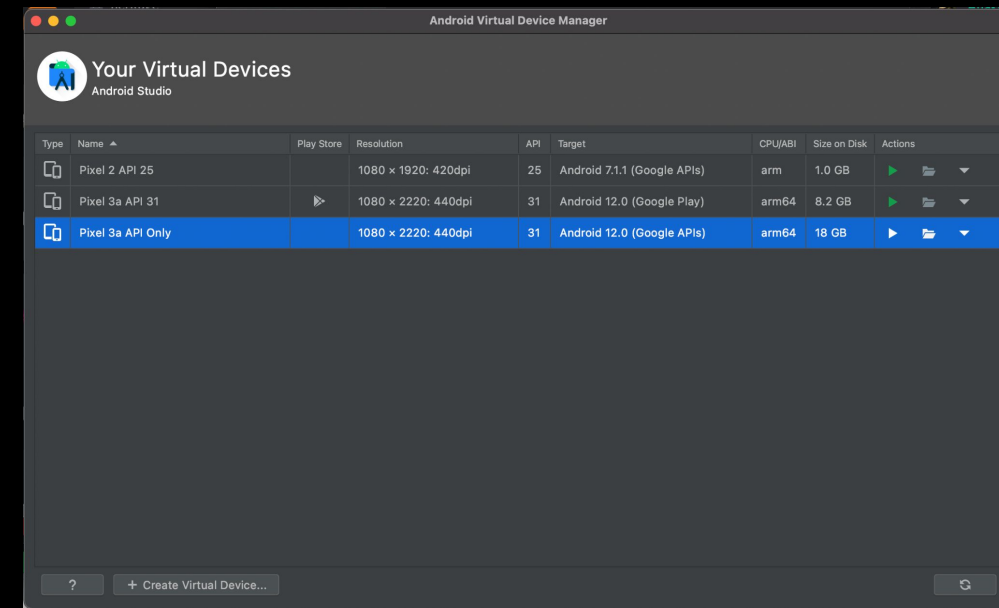


Setup

- Install Android Studio, make a AVD (Android Virtual Device) without the play store
 - Google doesn't allow you to root an emulator w/ play store

ADB - Android Debug Bridge

- Allows us to have full control of the emulator
- Download the Crossy Road APK
- Sideload the Crossy Road APK onto emulator
- Play the game to ensure everything is working



Analysis

Objective: Change High Score

Since Crossy Road works offline, we know game data is stored on-device

```
[(base) 🍏 ~/repos/crossy_road/ adb root
adb is already running as root
[(base) 🍏 ~/repos/crossy_road/ adb shell
emulator64_arm64:/ # cd /data/data/com.yodo1.crossyroad/
emulator64_arm64:/data/data/com.yodo1.crossyroad # ls
app_SafeDK_INTERSTITIAL  app_data  app_textures  app_webview  cache  code_cache  databases  files  no_backup  oat  shared_prefs
emulator64_arm64:/data/data/com.yodo1.crossyroad # exit
[(base) 🍏 ~/repos/crossy_road/ adb pull /data/data/com.yodo1.crossyroad .
/data/data/com.yodo1.crossyroad/: 161 files pulled, 0 skipped. 26.3 MB/s (7153536 bytes in 0.259s)
[(base) 🍏 ~/repos/crossy_road/ cd com.yodo1.crossyroad
[(base) 🍏 ~/repos/crossy_road/com.yodo1.crossyroad/ grep -ri 'score' .
./shared_prefs/com.yodo1.crossyroad.v2.playerprefs.xml: <int name="offlineHighScore" value="51" />
```

Locate file in /data/data/*.crossyroad containing score, and download

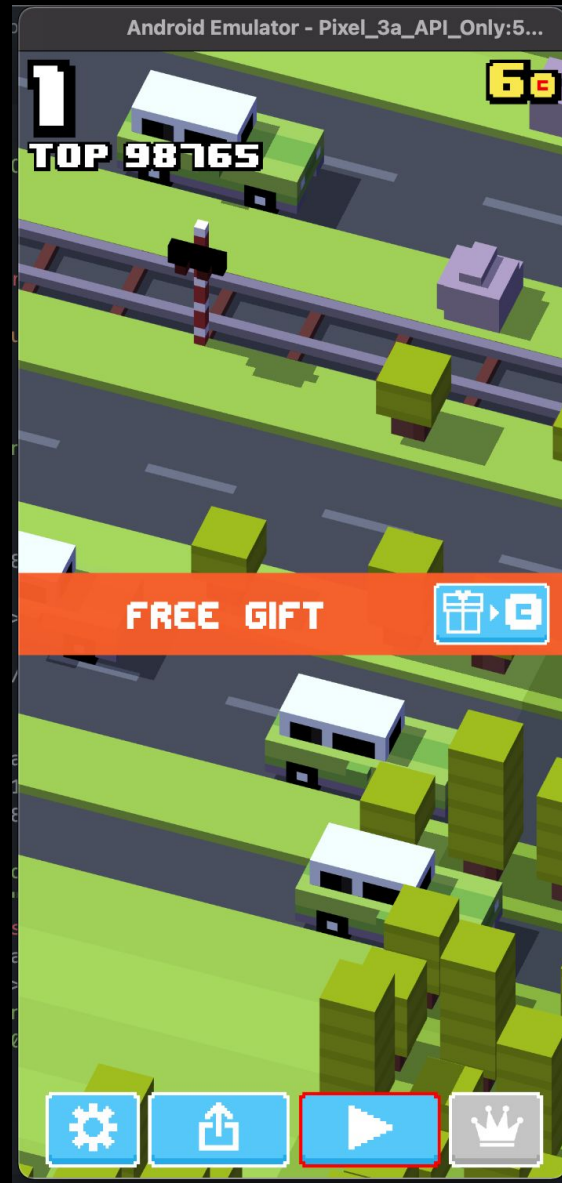


Execution

```
com.yodo1.crossyroad.v2.playerprefs.xml ×
com.yodo1.crossyroad.v2.playerprefs.xml
1  <?xml version='1.0' encoding='utf-8' standalone='yes' ?>
2  <map>
3      <string name="unity.player_sessionid">3286190662122065507</string>
4      <int name="hasPlayerBefore" value="1" />
5      <int name="offlineHighScore" value="98765" />
6      <int name="Screenmanager%20Resolution%20Width" value="1080" />
7      <int name="SH_DV" value="1" />
8      <string name="lastGAorientationSubmit">132828611644639560</string>
9      <string name="SH_U_0">019</string>
10     <int name="qd_bs" value="0" />
[100] emu4160104_81m07 /data/data/com.yodo1.crossyroad/shared_prefs π exit
[(base) ~ /repos/crossy_road/com.yodo1.crossyroad/ adb push shared_prefs/com.yodo1.crossyroad.v2.playerprefs.xml /data/data/com.yodo1.crossyroad/shared_prefs/
shared_prefs/com.yodo1.crossyroad.v2.playerprefs.xml: 1 file pushed, 0 skipped. 2.4 MB/s (2657 bytes in 0.001s)
```



Result



Frida Demo

FRIDA



Setup

- Decompile APK to java using JADX, get frida running

MainActivity.java

```
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setContentView((int) R.layout.activity_main);
    String property = System.getProperty("user.home");
    String str = System.getenv("USER");
    if (property == null || property.isEmpty() | !property.equals("Russia")) {
        a("Integrity Error", "This app can only run on Russian devices.");
    }
} else {
    a.a(this);
    startActivity(new Intent(this, LoginActivity.class));
}
}
```



Frida

main.py

```
import frida
device = frida.get_usb_device()
pid = device.spawn(["com.demo.demoapp"])
session = device.attach(pid)
script = session.create_script(open("hook.js").read())
script.load()
device.resume(pid)

# Prevent the script from terminating
input()
```

hook.js

```
console.log("My first frida script!");
```



Circumventing Check

```
Java.perform(function() {  
  // Retrieve the class with Java.use  
  const System = Java.use("java.lang.System");  
  // Select the correct overload of getProperty  
  const propertyMethod = System.getProperty.overload('java.lang.String');  
  // Modify the implementation  
  propertyMethod.implementation = function (prop) {  
    // Log the event  
    console.log("getProperty('" + prop.toString() + "') called" );  
    // Log the original value  
    const ret = propertyMethod.call(this, prop);  
    console.log("Value is: '" + ret + "'");  
    // Change the return value  
    return "Russia";  
  }  
});
```



Summary

Frida is like tampermonkey for iOS and android apps

- can “hook” functions, and
 - read values
 - modify outputs
 - change function / class implementation



APIs & Certificate Pinning



API Sniffing

- Many apps will use proprietary/unofficial APIs for their apps
- We can make a Man-In-The-Middle attack to figure out what endpoints they use
- Personal Recommendations:
 - mitmproxy - linux
 - Charles - \$2.99 on iOS (works on device, can export logs)
 - Burp Suite

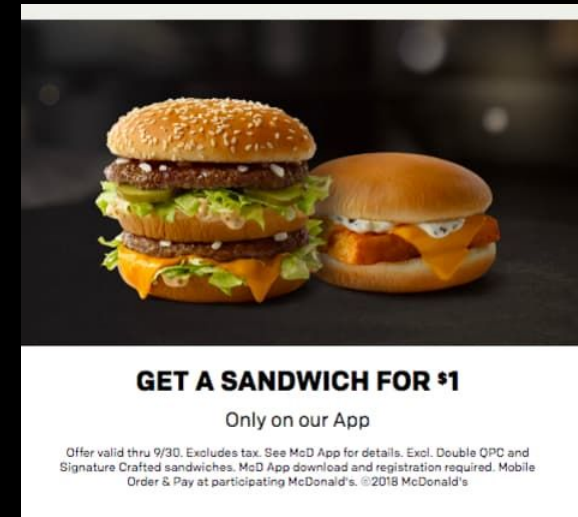


Personal Example

```
#!/usr/bin/env python3

client = Client("API_KEY")
client.sign_in("USERNAME","PASSWORD")
stores = client.find_stores(*client.lookup_zip())
menu = client.menu(stores[0])
items = client.order_picker(menu)
cards = client.cards()
client.order(cards[0], items)
total = client.get_price(items)
confirm = input("Pay {} and complete transaction? [y/N] > ".format(total))
if confirm == "y":
    order_number = client.pickup()
    print("Order #{}".format(order_number))
```

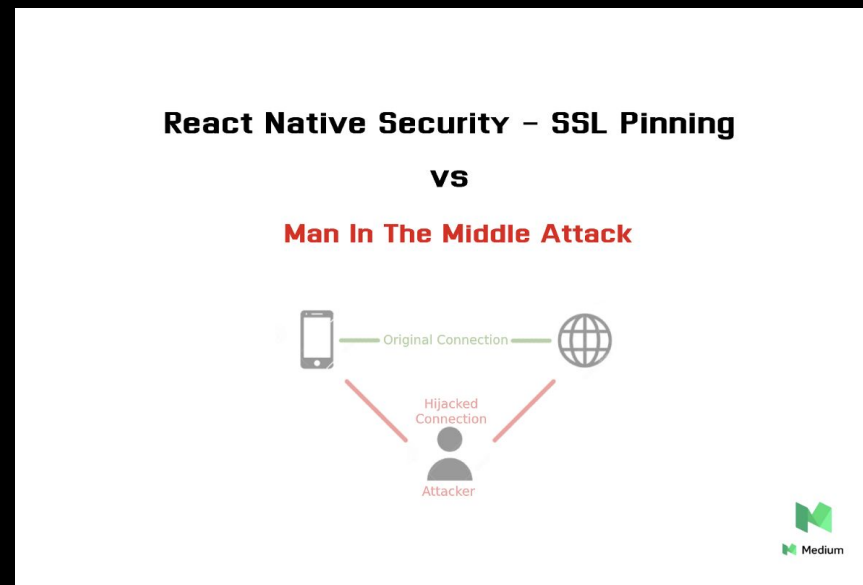
- McDonald's near my HS home didn't allow you to mobile order with McDonalds app until you arrive at location
- Sniff API Calls using Charles
 - Reverse-Engineer their ordering protocol, create python wrapper
- Create script to create account, order from home, and use \$1 big mac promotion every day for ~2 weeks
 - For some reason, they updated the API...
- How do apps protect against this?



Certificate Pinning

Nowadays, most apps pin their TLS Certificates

- Server sends over certificate
- Logic inside app verifies if certificate is valid
- If not valid, connection terminated!
 - Try using Netflix, TikTok, or Snapchat using a VPN! It may not work....



Cracking Certificate Pinning

The In-App check can be
Frida!!!

- Frida is a tool to trace, inspect inputs/outputs, and modify functions while an app is running

Modifying function returns is *extremely* powerful

- We can make the certificate check always return true!

“ It’s Greasemonkey for native apps, or, put in more technical terms, it’s a dynamic code instrumentation toolkit. It lets you inject snippets of JavaScript or your own library into native apps on Windows, macOS, GNU/Linux, iOS, Android, and QNX. Frida also provides you with some simple tools built on top of the Frida API. These can be used as-is, tweaked to your needs, or serve as examples of how to use the API.”



Types

Certificate pinning can be much harder to crack depending on how much the application cares:

- Pinning in Java - not that tricky, scripts exist that cover almost every case
 - <https://codeshare.frida.re/@pcipolloni/universal-android-ssl-pinning-bypass-with-frida/>
- Native Pinning - if using a standard pinning method (e.g. libboring.so + libssl.so), articles will exist to assist
- Flutter/Dart Bypass Pinning - tricky to setup

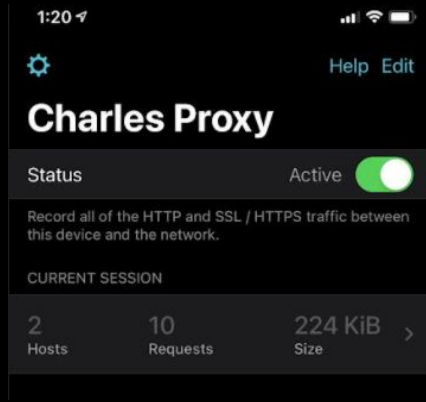
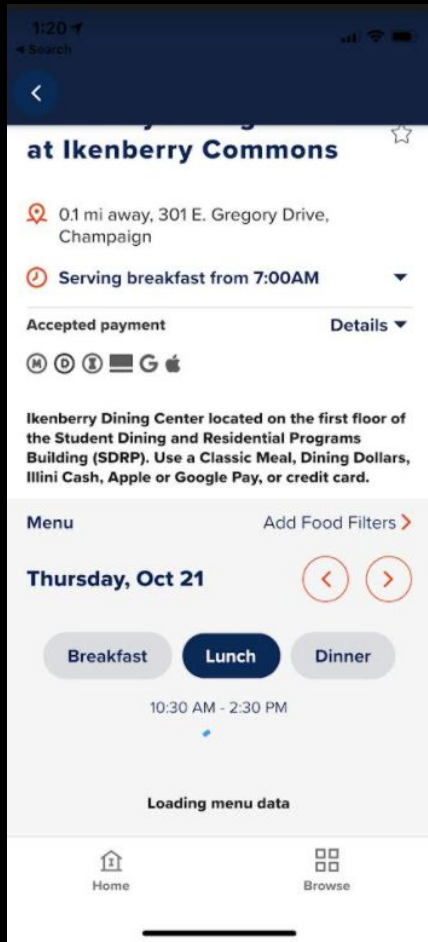
Super in-depth guide →

<https://http toolkit.tech/blog/android-reverse-engineering/>



Example: Unpinning the Dining Hall App

Step 1: Identify Pinning is in place



Step 2: Identify what language the application was made in

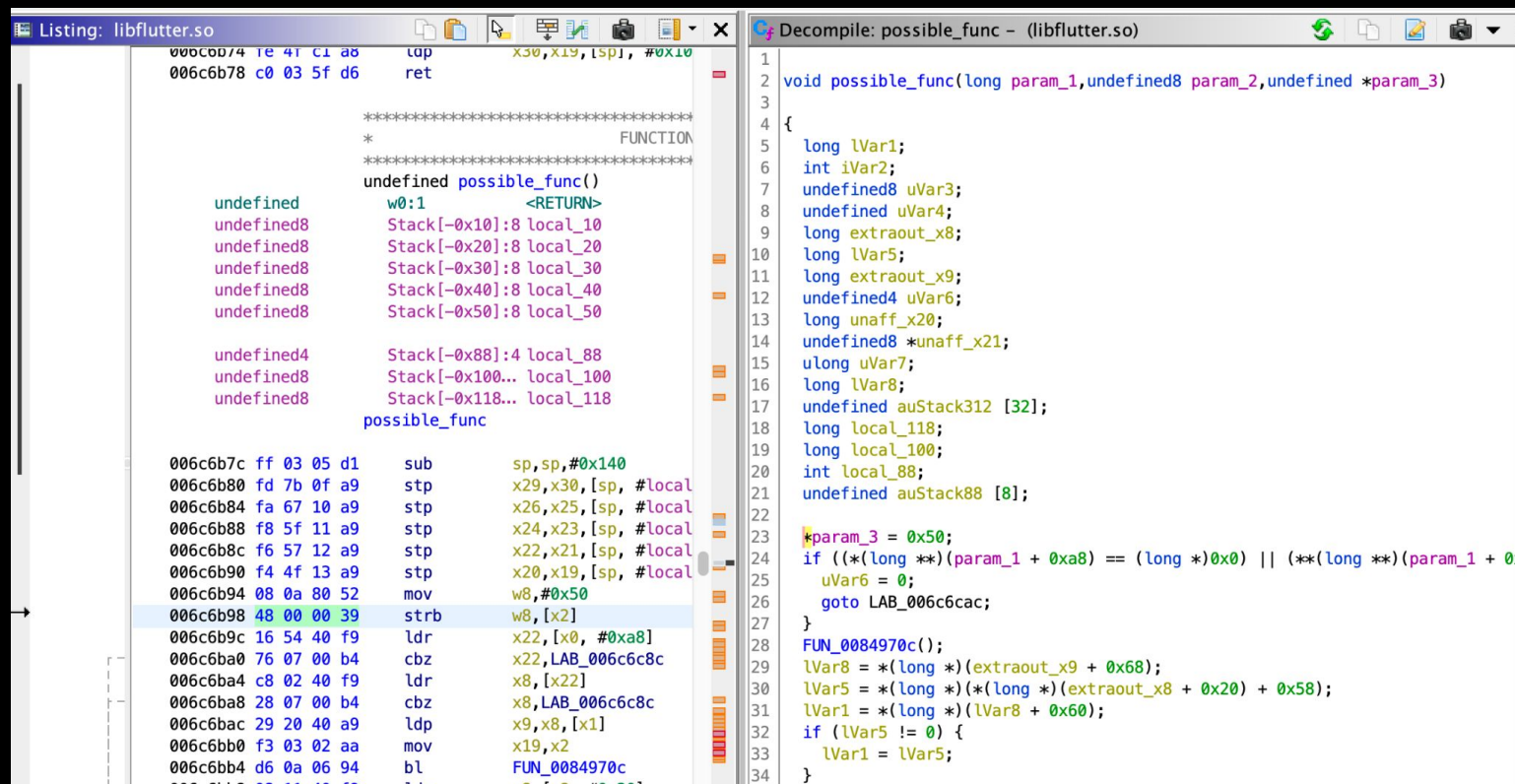
In my case, the libapp.so hints at this being a flutter app



Example: Unpinning the Dining Hall App

Step 3: Identify function to overwrite in Ghidra

- research! <https://id.horangi.com/blog/bypass-ssl-pinning-di-flutter-library/>



The screenshot displays the Ghidra interface with two windows. The left window, titled 'Listing: libflutter.so', shows assembly code for a function. The right window, titled 'Decompile: possible_func - (libflutter.so)', shows the corresponding decompiled C code.

```
Listing: libflutter.so
006c6b74 1e 41 c1 a8 ldp    x30,x19,[sp],#0x10
006c6b78 c0 03 5f d6 ret

*****
*                               FUNCTION
*****
undefined possible_func()
w0:1      <RETURN>
undefined Stack[-0x10]:8 local_10
undefined Stack[-0x20]:8 local_20
undefined Stack[-0x30]:8 local_30
undefined Stack[-0x40]:8 local_40
undefined Stack[-0x50]:8 local_50

undefined Stack[-0x88]:4 local_88
undefined Stack[-0x100... local_100
undefined Stack[-0x118... local_118
possible_func
006c6b7c ff 03 05 d1 sub    sp,sp,#0x140
006c6b80 fd 7b 0f a9 stp    x29,x30,[sp,#local
006c6b84 fa 67 10 a9 stp    x26,x25,[sp,#local
006c6b88 f8 5f 11 a9 stp    x24,x23,[sp,#local
006c6b8c f6 57 12 a9 stp    x22,x21,[sp,#local
006c6b90 f4 4f 13 a9 stp    x20,x19,[sp,#local
006c6b94 08 0a 80 52 mov    w8,#0x50
006c6b98 48 00 00 39 strb   w8,[x2]
006c6b9c 16 54 40 f9 ldr    x22,[x0,#0xa8]
006c6ba0 76 07 00 b4 cbz    x22,LAB_006c6c8c
006c6ba4 c8 02 40 f9 ldr    x8,[x22]
006c6ba8 28 07 00 b4 cbz    x8,LAB_006c6c8c
006c6bac 29 20 40 a9 ldp    x9,x8,[x1]
006c6bb0 f3 03 02 aa mov    x19,x2
006c6bb4 d6 0a 06 94 bl     FUN_0084970c
006c6bb8 00 11 40 f9 ldr    w0,[sp,#0x20]

Decompile: possible_func - (libflutter.so)
1 void possible_func(long param_1,undefined8 param_2,undefined *param_3)
2
3
4 {
5     long lVar1;
6     int iVar2;
7     undefined8 uVar3;
8     undefined uVar4;
9     long extraout_x8;
10    long lVar5;
11    long extraout_x9;
12    undefined4 uVar6;
13    long unaff_x20;
14    undefined8 *unaff_x21;
15    ulong uVar7;
16    long lVar8;
17    undefined auStack312 [32];
18    long local_118;
19    long local_100;
20    int local_88;
21    undefined auStack88 [8];
22
23    *param_3 = 0x50;
24    if ((*long **)(param_1 + 0xa8) == (long *)0x0 || (**(long **)(param_1 + 0
25        uVar6 = 0;
26        goto LAB_006c6cac;
27    }
28    FUN_0084970c();
29    lVar8 = *(long *)(extraout_x9 + 0x68);
30    lVar5 = *(long *)((*long *) (extraout_x8 + 0x20) + 0x58);
31    lVar1 = *(long *) (lVar8 + 0x60);
32    if (lVar5 != 0) {
33        lVar1 = lVar5;
34    }
```



Example: Frida Script

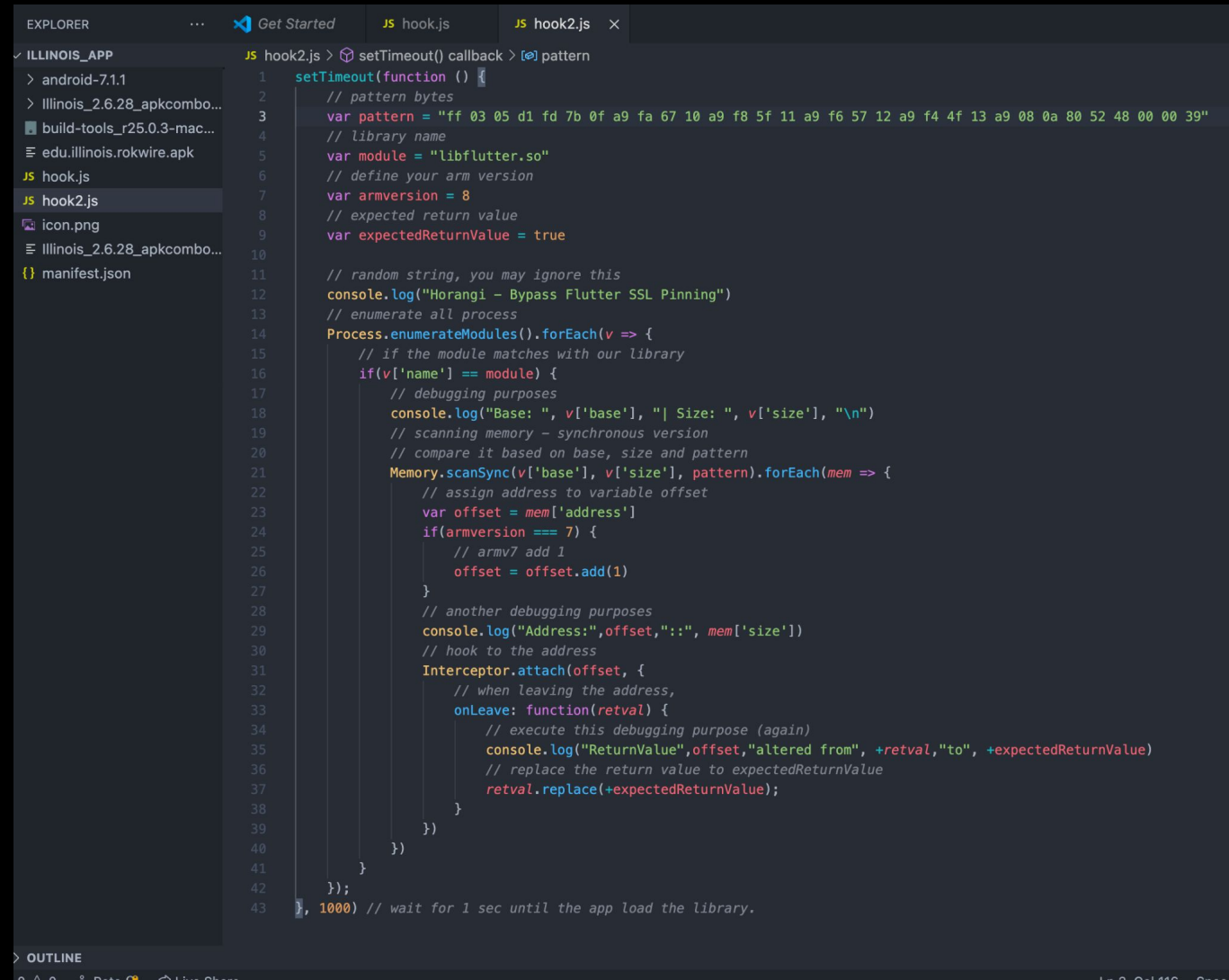
Step 4: Final script

A typical frida script will be able to overwrite function implementation

- Can add custom logic and logging

Native library functions injection is not as powerful

- Can only overwrite input and output values



```
EXPLORER
...
JS hook2.js
JS hook2.js x

ILLINOIS_APP
  > android-7.1.1
  > Illinois_2.6.28_apkcombo...
  build-tools_r25.0.3-mac...
  edu.illinois.rokwire.apk
  JS hook.js
  JS hook2.js
  icon.png
  Illinois_2.6.28_apkcombo...
  {} manifest.json

JS hook2.js > setTimeout() callback > pattern
1  setTimeout(function () {
2    // pattern bytes
3    var pattern = "ff 03 05 d1 fd 7b 0f a9 fa 67 10 a9 f8 5f 11 a9 f6 57 12 a9 f4 4f 13 a9 08 0a 80 52 48 00 00 39"
4    // library name
5    var module = "libflutter.so"
6    // define your arm version
7    var armversion = 8
8    // expected return value
9    var expectedReturnValue = true
10
11   // random string, you may ignore this
12   console.log("Horangi - Bypass Flutter SSL Pinning")
13   // enumerate all process
14   Process.enumerateModules().forEach(v => {
15     // if the module matches with our library
16     if(v['name'] == module) {
17       // debugging purposes
18       console.log("Base: ", v['base'], "| Size: ", v['size'], "\n")
19       // scanning memory - synchronous version
20       // compare it based on base, size and pattern
21       Memory.scanSync(v['base'], v['size'], pattern).forEach(mem => {
22         // assign address to variable offset
23         var offset = mem['address']
24         if(armversion == 7) {
25           // armv7 add 1
26           offset = offset.add(1)
27         }
28         // another debugging purposes
29         console.log("Address:", offset, ":", mem['size'])
30         // hook to the address
31         Interceptor.attach(offset, {
32           // when leaving the address,
33           onLeave: function(retval) {
34             // execute this debugging purpose (again)
35             console.log("ReturnValue", offset, "altered from", +retval, "to", +expectedReturnValue)
36             // replace the return value to expectedReturnValue
37             retval.replace(+expectedReturnValue);
38           }
39         })
40       })
41     }
42   });
43   //, 1000) // wait for 1 sec until the app load the library.
```


Any questions?

