iOS Pentesting Series Part 3- The Ceasefire

Kishor balan · Following

7 min read · Aug 19, 2023

Listen Share More

Hola mates,

What's crackin'? I've miraculously wrapped up the last installment of this series in record time. And oh, by the way, my trusty microphone decided to play hide and seek during the video recording. Not cool, right? :/

Time's ticking away, so let's dive right in. In this episode, we're gonna tackle the following points:

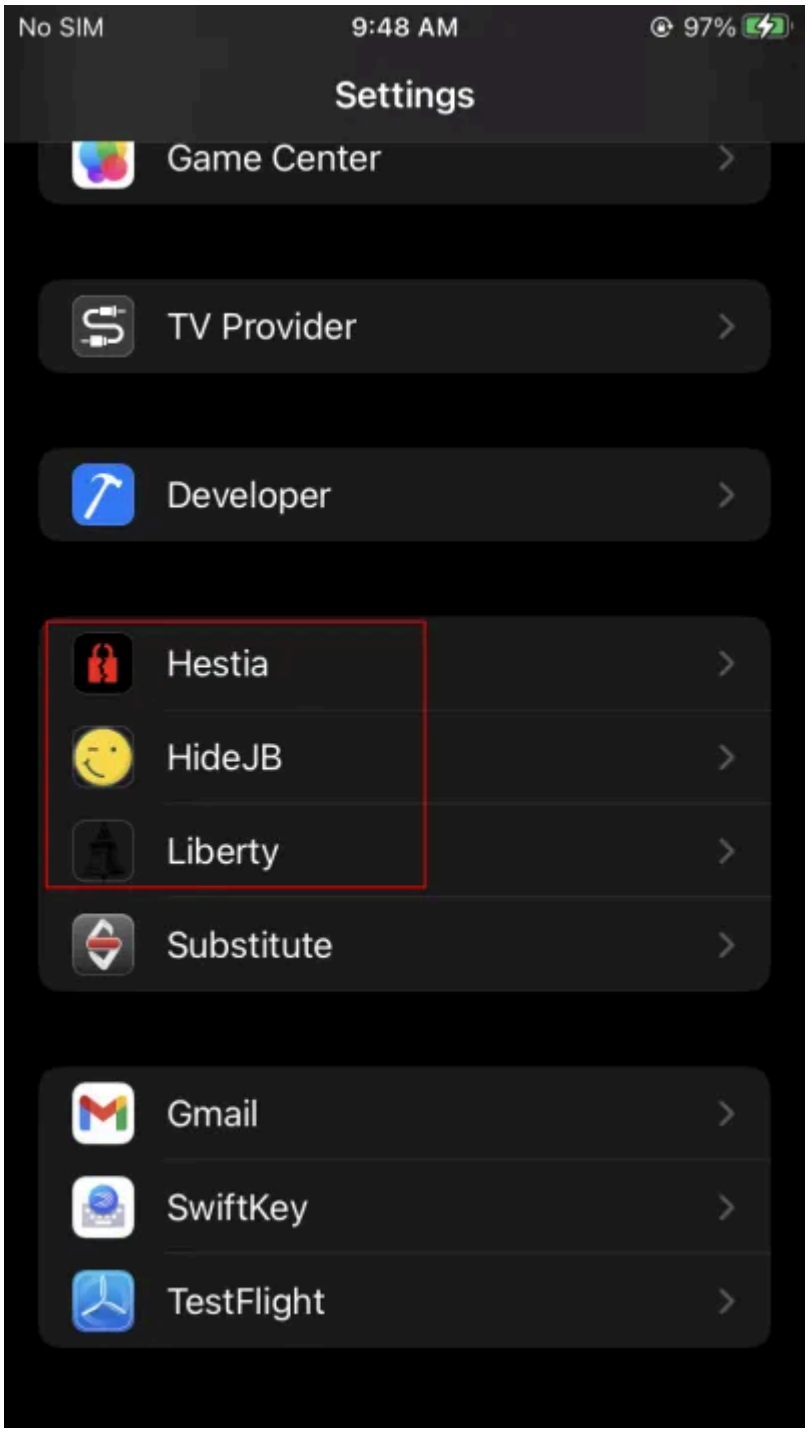
1. JailBreak detection Bypass
i. Using Tweaks
ii. Using Publicly available frida scripts
iii.Using Using the default Jailbreak Bypass module that is available in Objection framework.
iv.Making our own frida script for bypassing JailBreak detection.

2. Setting up the Proxy and Bypassing SSL Pinning
i. Configuring the Proxy
ii. SSL Pinning Bypass using Tweaks
iii. SSL Pinning Bypasss Using Objection
iv. SSL Pinning Bypasss using Frida scripts

3. Intercepting Flutter iOS applications.

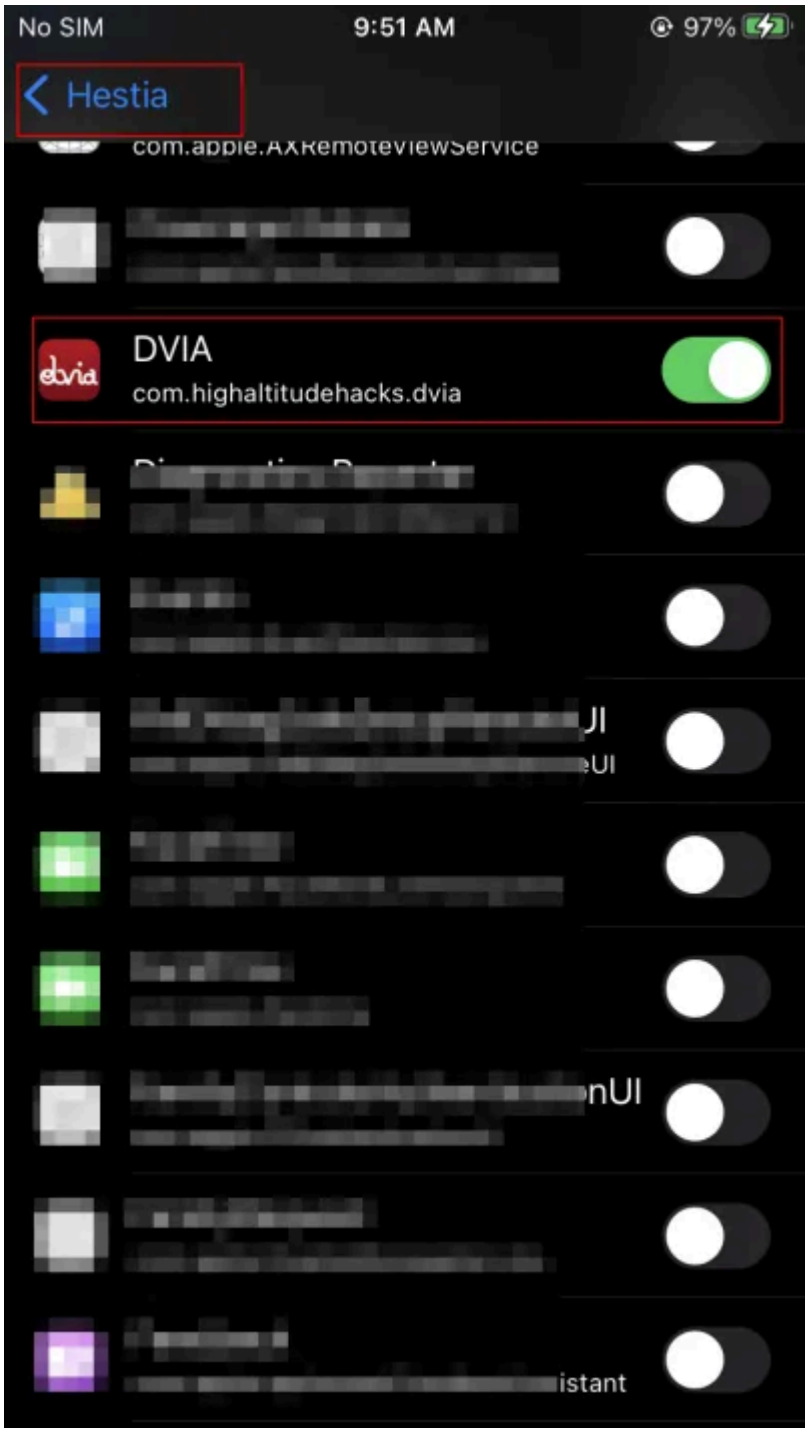
1. JailBreak detection Bypass.

Method — 1: Using Tweaks, Hope you have already installed the Jailbreak detection bypass tweaks that are mentioned in my Part-1 article.
Most of those tweaks can be seen in the device settings.



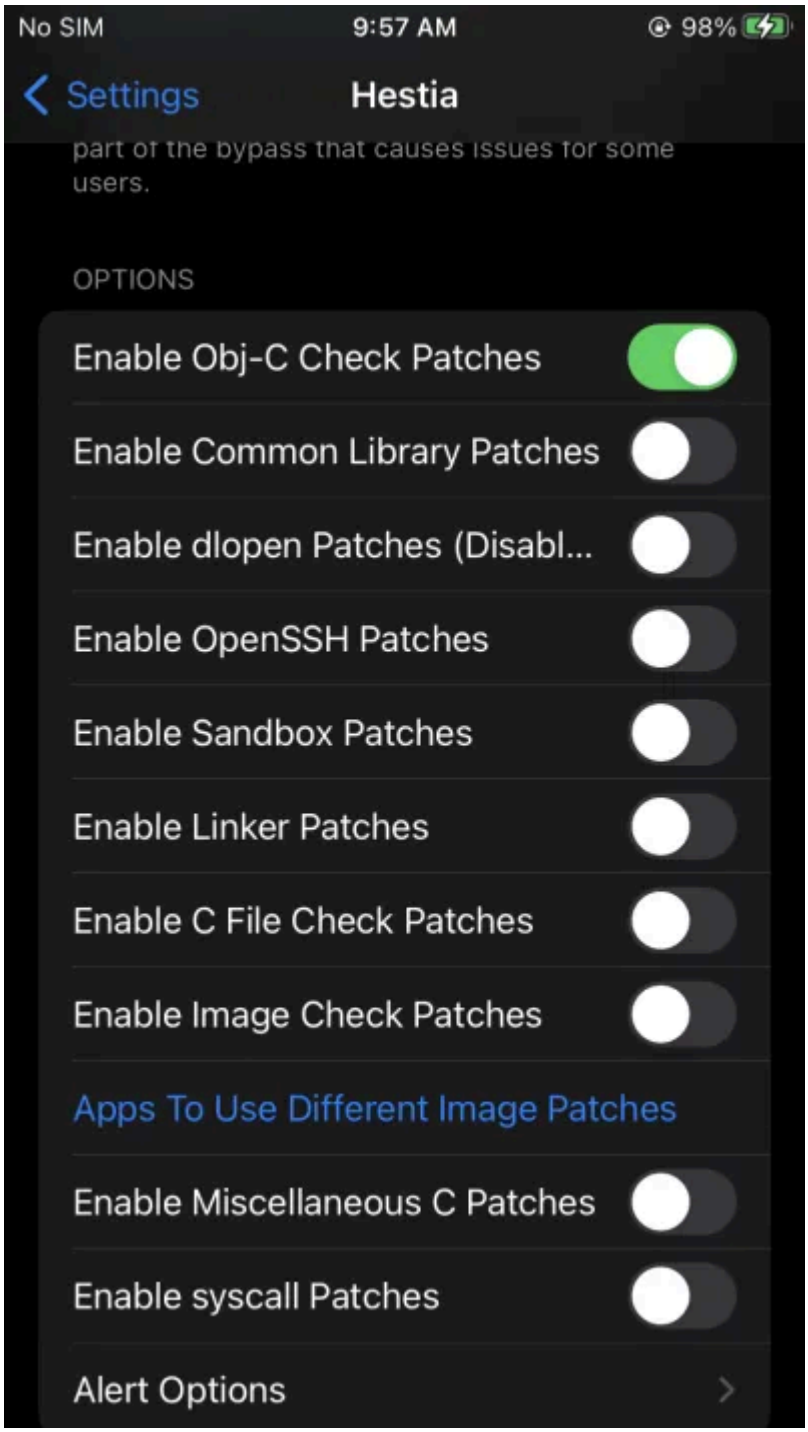
Let's try the Hestia Bypass

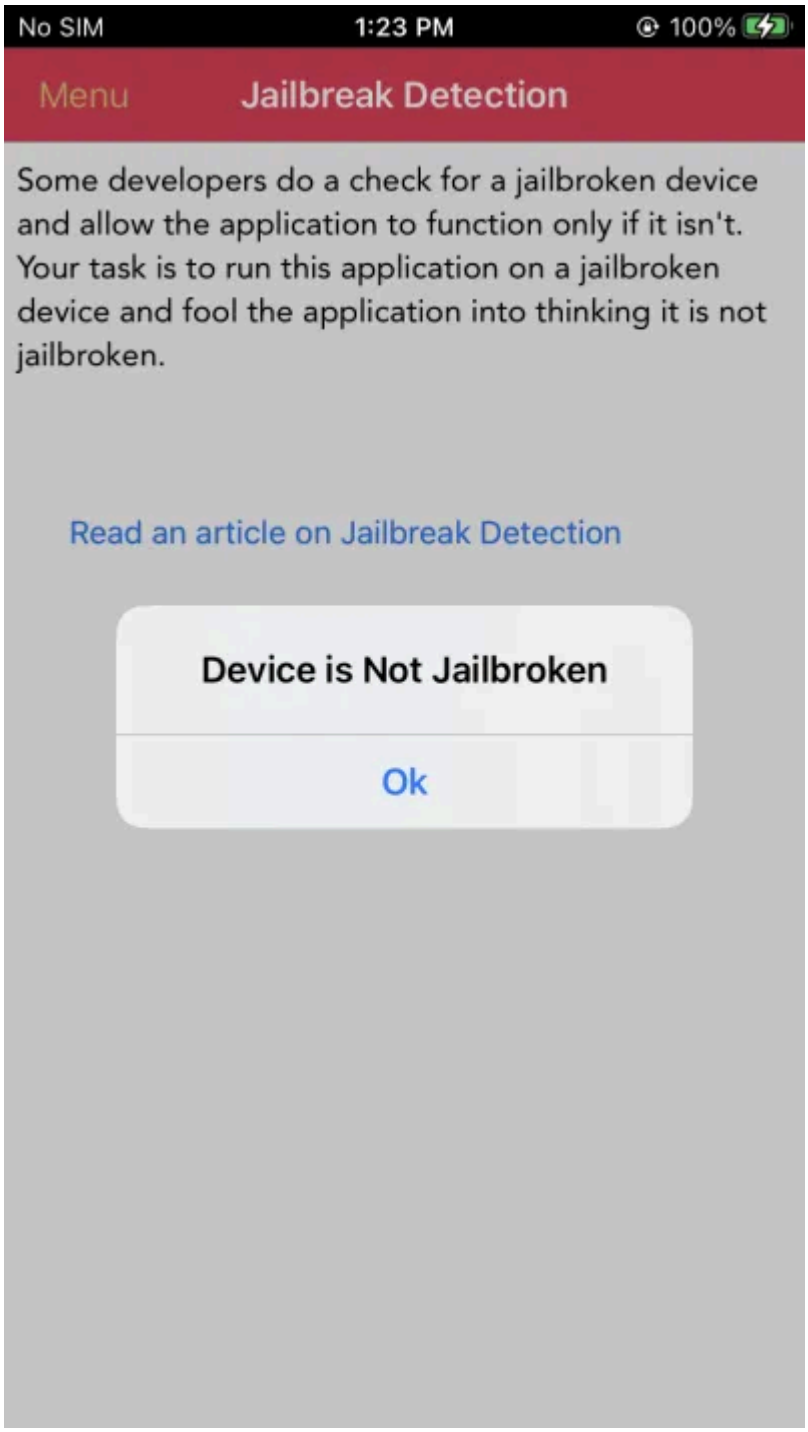
Enabling Hestia for the target application.



Few applications may misbehave or won't bypass even if we are enabling Hestia for them. ummm..Well in those cases, we have other options available in Hestia application.

> From the Hestia settings, Enable the Obj-C checks Patches only.


No SIM
9:57 AM
98%
Settings
Hestia
part of the bypass that causes issues for some users.
OPTIONS
Enable Obj-C Check Patches
Enable Common Library Patches
Enable dlopen Patches (Disabl...
Enable OpenSSH Patches
Enable Sandbox Patches
Enable Linker Patches
Enable C File Check Patches
Enable Image Check Patches
Apps To Use Different Image Patches
Enable Miscellaneous C Patches
Enable syscall Patches
Alert Options

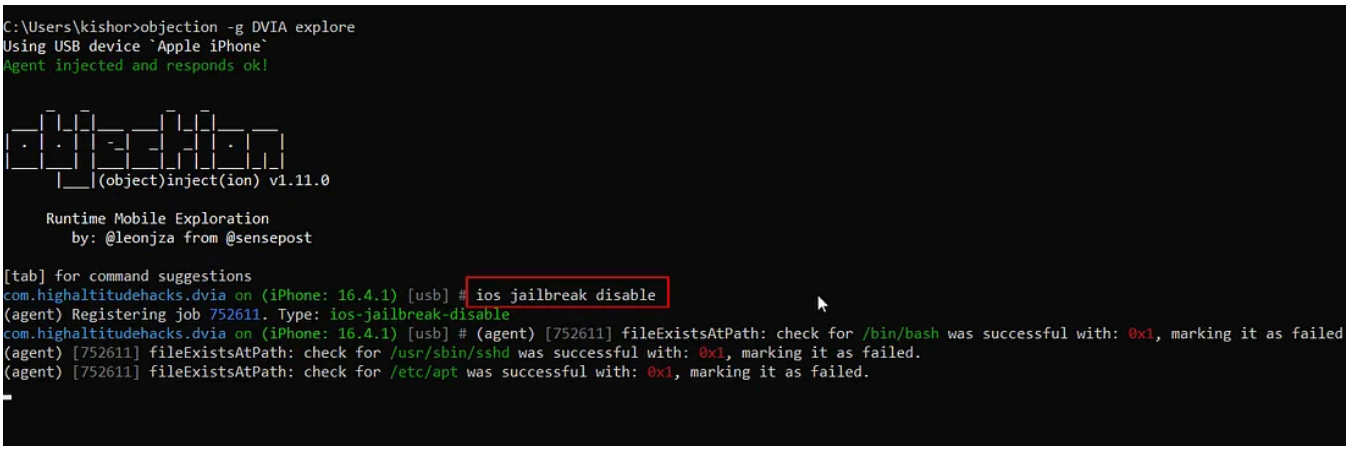


This will work in most cases, otherwise feel free to try other tweaks as well.

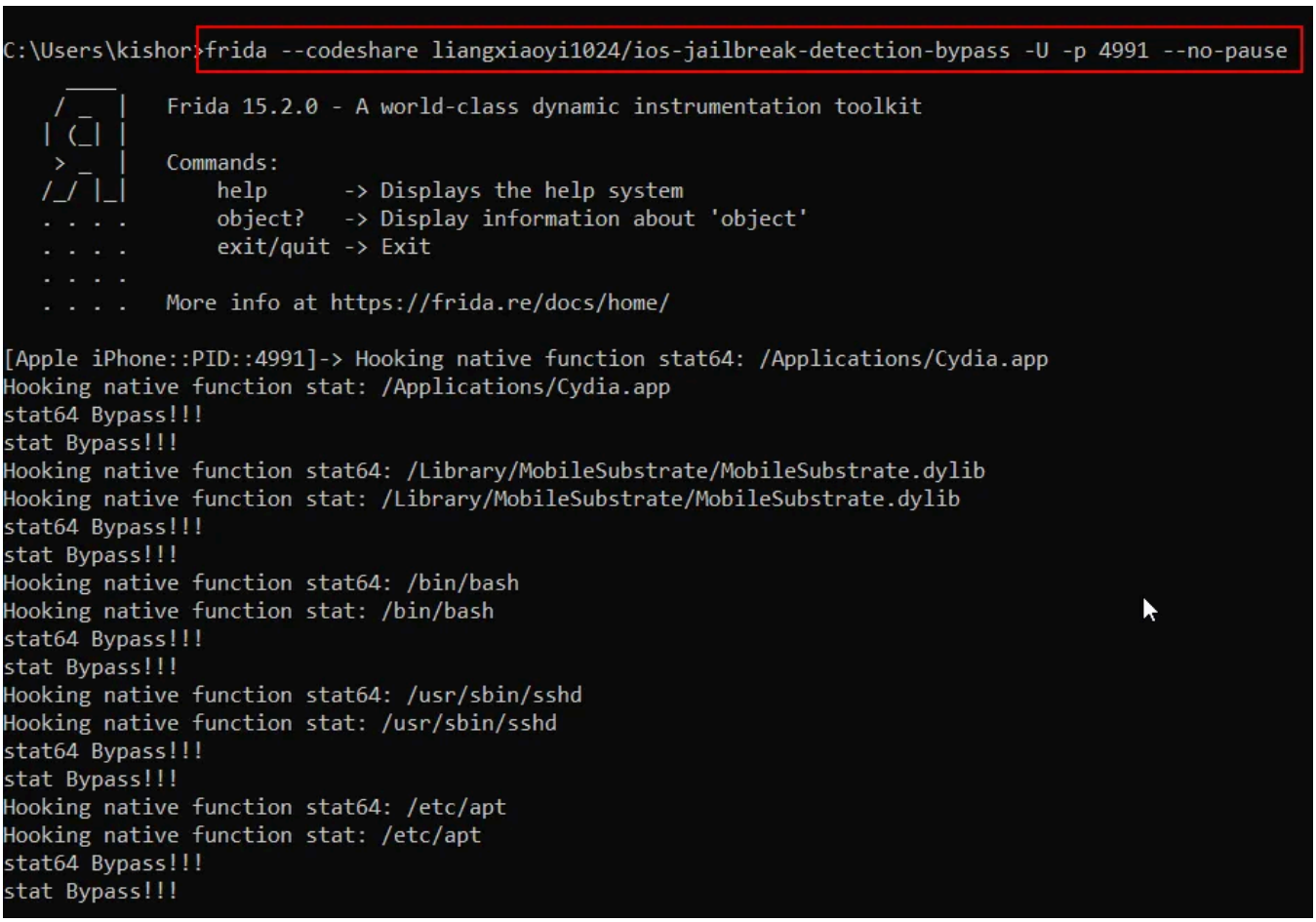
Method: 2

Using the default Jailbreak Bypass module that is available in Objection framework

i) Connect the target Application and execute the command shown below



Method -3 : Blindly trying out the publicly available frida scripts for bypassing Jailbreak detection



Here i used a publicly available script that was published in codeshare.
Frida CodeShare

Method-4 : Making our custom Frida Scripts for bypassing jailbreak Detection.

No coding exp? Don’t worry same goes here too ..lol

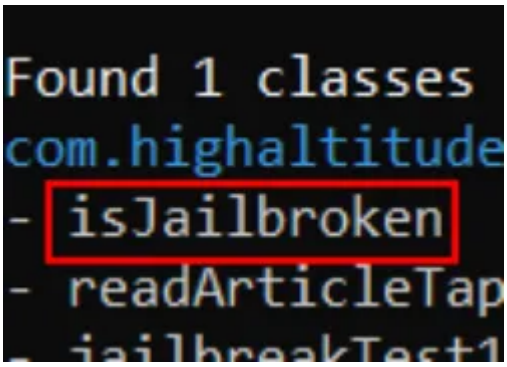
i) Connect the target application with objection and search for the Jailbreak detection classes

```
C:\Users\kishor>objection -g DVIA explore
Using USB device `Apple iPhone`
Agent injected and responds ok!

     _   _         _   _
 ___| |_|_|___ ___| |_|_|___ ___
| . | . | | -_|  _|  _| | . |   |
|___|___| |___|___|_| |_|___|_|_|
      |___|(object)inject(ion) v1.11.0

     Runtime Mobile Exploration
        by: @leonjza from @sensepost

[tab] for command suggestions
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # ios hooking search classes jail
JailbreakDetectionVC

Found 1 classes
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] #
```

ii) Now enumerate the available methods in our target class

```
Found 1 classes
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # ios hooking list class_methods JailbreakDetectionVC
- isJailbroken
- readArticleTapped:
- jailbreakTest1Tapped:
- jailbreakTest2Tapped:
- initWithNibName:bundle:
- didReceiveMemoryWarning
- viewDidLoad

Found 7 methods
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] #
```

iii) Look for the Blask sheep xD



Awww there he is..

iv) Now let's make a simple script for Intercepting our target class

```
[tab] for command suggestions
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # ios hooking generate simple JailbreakDetectionVC
var target = ObjC.classes.JailbreakDetectionVC;

Interceptor.attach(target['- isJailbroken'].implementation, {
  onEnter: function (args) {
    console.log('Entering - isJailbroken!');
  },
  onLeave: function (retval) {
    console.log('Leaving - isJailbroken');
  },
});


Interceptor.attach(target['- readArticleTapped:'].implementation, {
  onEnter: function (args) {
    console.log('Entering - readArticleTapped:!');
  },
  onLeave: function (retval) {
    console.log('Leaving - readArticleTapped:');
  },
});


Interceptor.attach(target['- jailbreakTest1Tapped:'].implementation, {
  onEnter: function (args) {
    console.log('Entering - jailbreakTest1Tapped:!');
  },
  onLeave: function (retval) {
    console.log('Leaving - jailbreakTest1Tapped:');
  },
});


Interceptor.attach(target['- jailbreakTest2Tapped:'].implementation, {
  onEnter: function (args) {
    console.log('Entering - jailbreakTest2Tapped:!');
  },
  onLeave: function (retval) {
    console.log('Leaving - jailbreakTest2Tapped:');
  },
});
```

vi) Aww dear Objection made that this easy ❤

vii) However, The entire script is not required for us except the following small part..

```
[tab] for command suggestions
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # ios hooking generate simple JailbreakDetectionVC
var target = ObjC.classes.JailbreakDetectionVC;

Interceptor.attach(target['- isJailbroken'].implementation, {
  onEnter: function (args) {
    console.log('Entering - isJailbroken!');
  },
  onLeave: function (retval) {
    console.log('Leaving - isJailbroken');
  },
});


Interceptor.attach(target['- readArticleTapped:'].implementation, {
  onEnter: function (args) {
    console.log('Entering - readArticleTapped:!');
  },
  onLeave: function (retval) {
    console.log('Leaving - readArticleTapped:');
  },
});


Interceptor.attach(target['- jailbreakTest1Tapped:'].implementation, {
```

vii) Lets modify the code to see whats really happening with those target class method and save that code that as a JS file.

```
var target = ObjC.classes.JailbreakDetectionVC;

Interceptor.attach(target['- isJailbroken'].implementation, {
  onEnter: function (args) {
    console.log('Entering - isJailbroken!');
  },
  onLeave: function (retval) {
    console.log('Return value '+ retval);
  },
});
```

Printing the return value

viii) Import the script in Objection and see what value returns when the jailbreak detection prompt appears

```
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # import ios.js
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # Entering - isJailbroken!
Return value0x1
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # import ios.js
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # Entering - isJailbroken!
Entering - isJailbroken!
Return value0x1
```

It looks like a boolean data type and its returns 1 if the device is jailbroken

lx) Now lets slightly modify the code for tampering the return value with 0

```
var target = ObjC.classes.JailbreakDetectionVC;

Interceptor.attach(target['- isJailbroken'].implementation, {
  onEnter: function (args) {
    console.log('Entering - isJailbroken!');
  },
  onLeave: function (retval) {
    retval.replace(0)
    console.log('Modified Return value '+ retval);
  },
});
```

Replacing the return value with 0

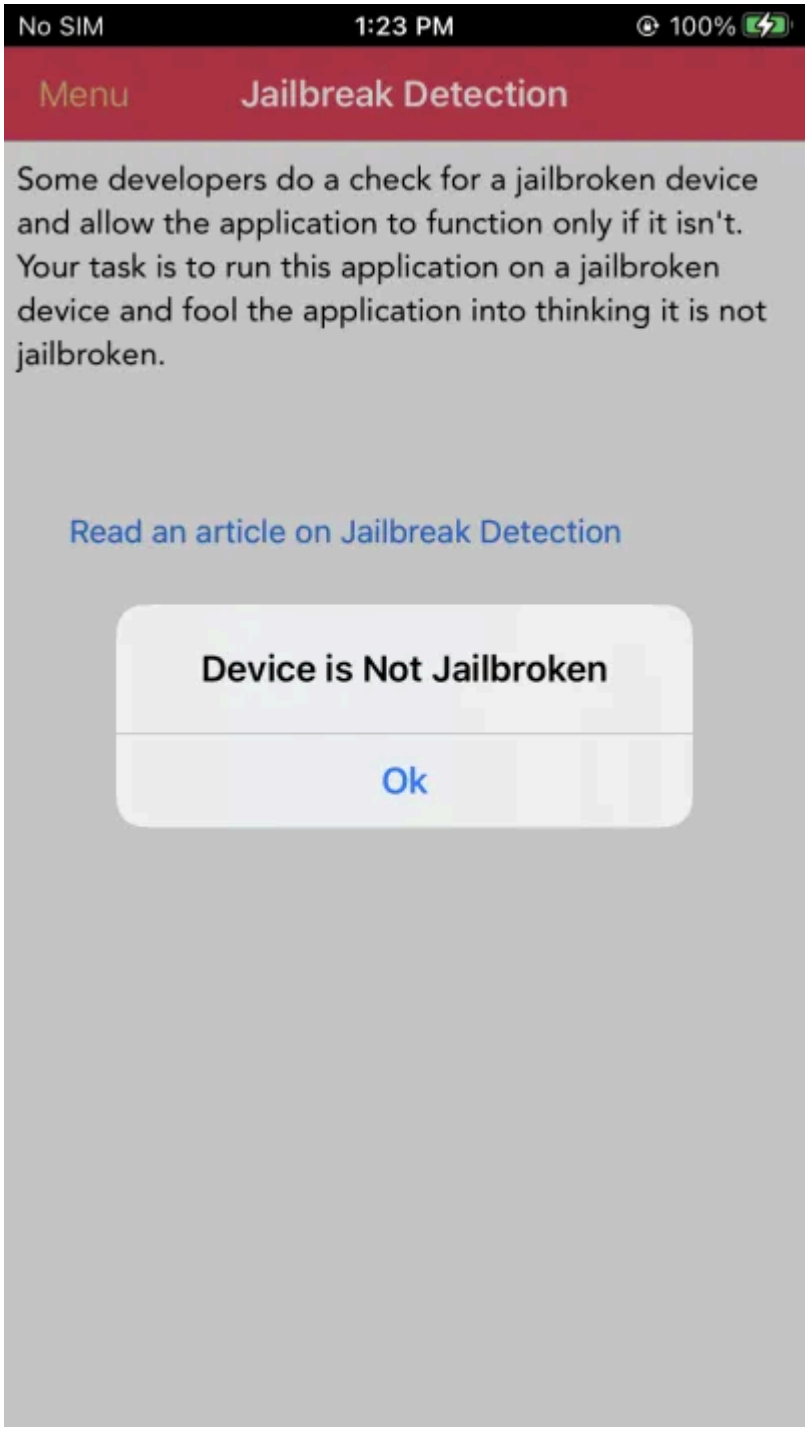
x) import the script again and see what happens when the jailbreak detection class gets loaded

```
     _    _         _    _
 ___| |_|_|___ ___| |_|_|___ ___
| . | . | | -_|  _|  _| | . |   |
|___|___| |___|___|_| |_|___|_|_|
      |___|(object)inject(ion) v1.11.0

     Runtime Mobile Exploration
        by: @leonjza from @sensepost

[tab] for command suggestions
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # import ios.js
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # Entering - isJailbroken!
Modified Return value 0x0
Entering - isJailbroken!
Modified Return value 0x0
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] #
```

Tampered the return value

As you can see in the picture, with our custom frida script, we were able to tamper the return value if isJailbroken class method during the runtime.,

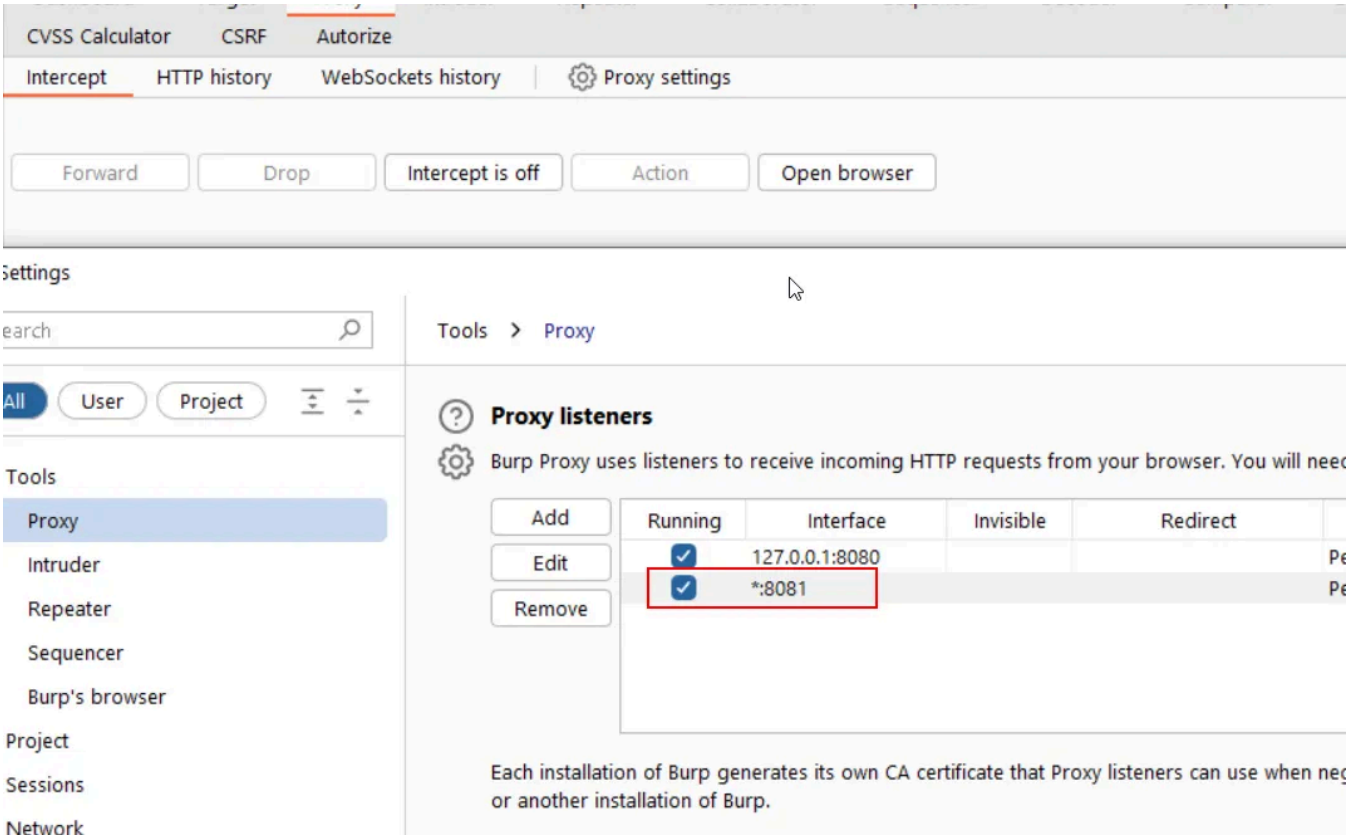
As a result, The Jailbreak detection will be bypassed.



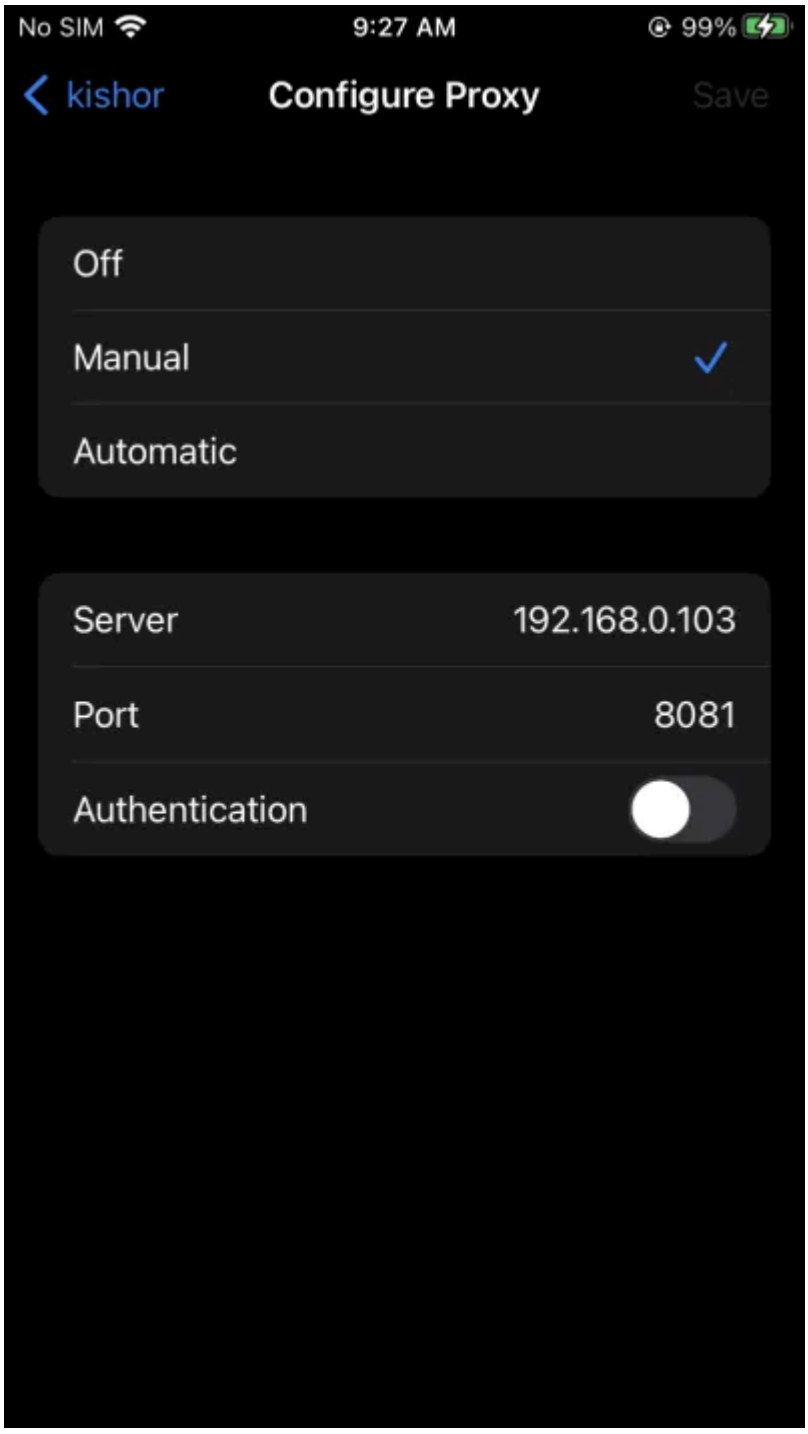
2. Setting up the Proxy and Bypassing SSL Pinning

i) Setting up proxy:

i. Fire up the BurpSuite and listen a port for all interface.

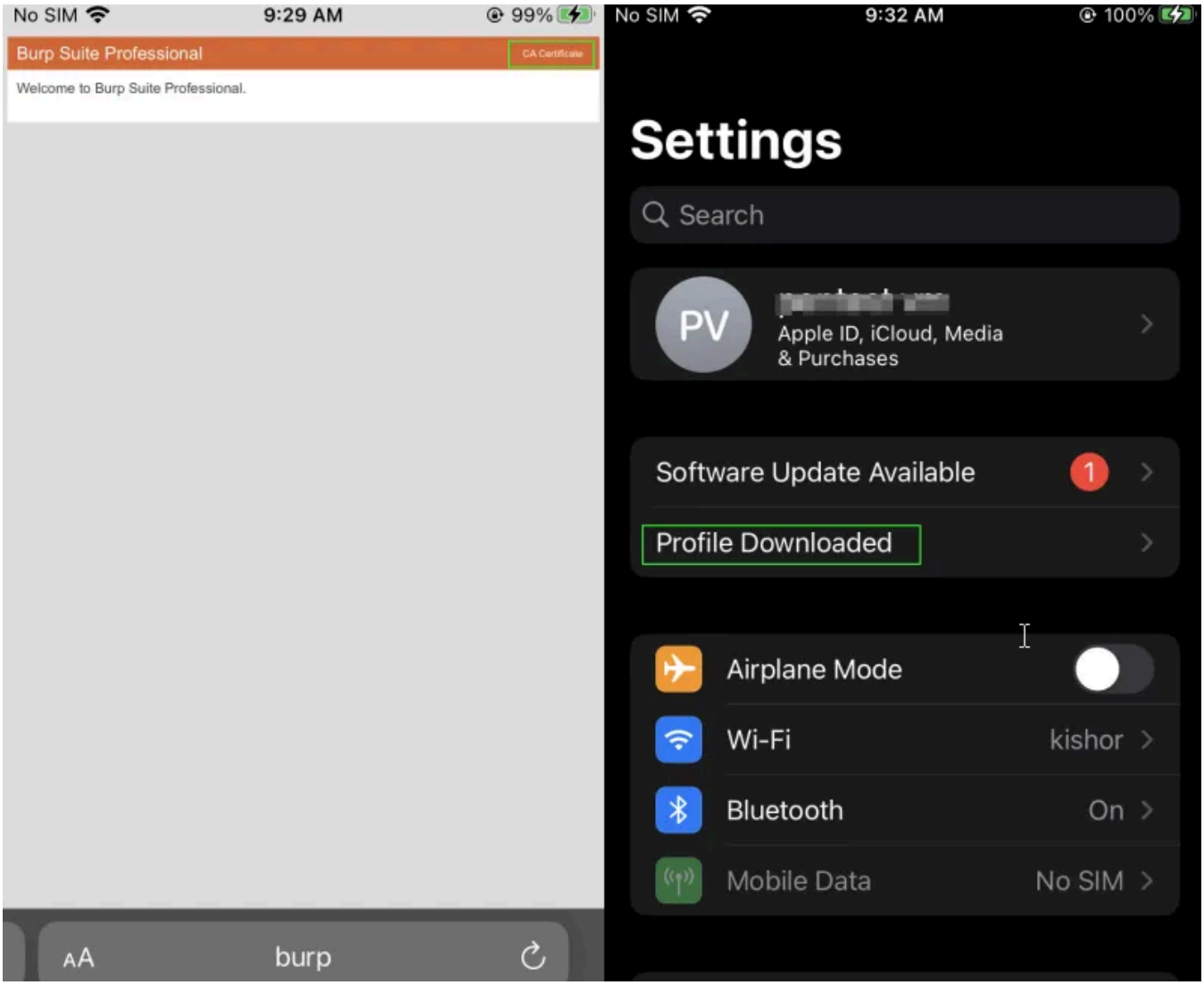


ii. Add the proxy details in the device wifi configuration

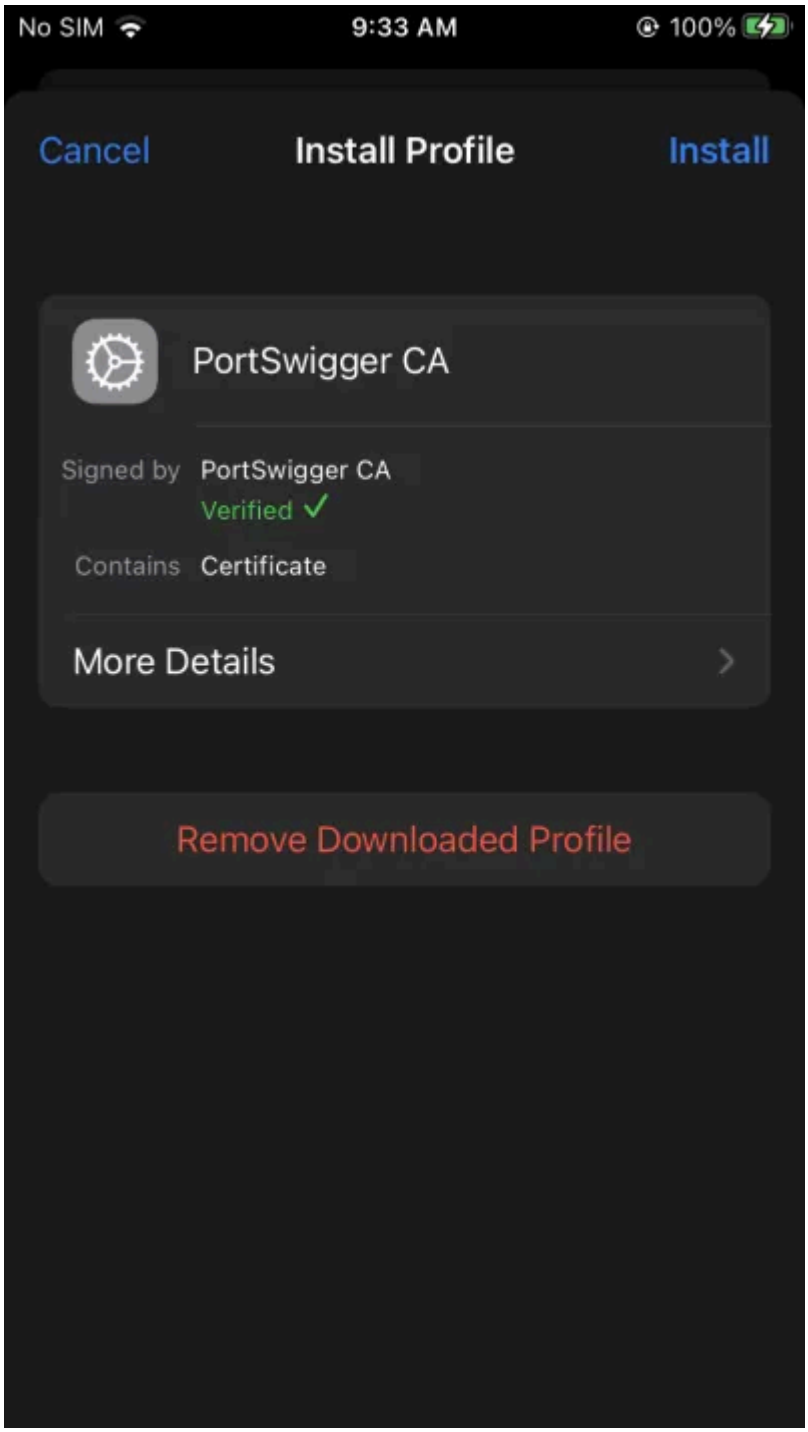


iii. Now go the device browser and hit “http://burp”
iv. Download the CA certificate

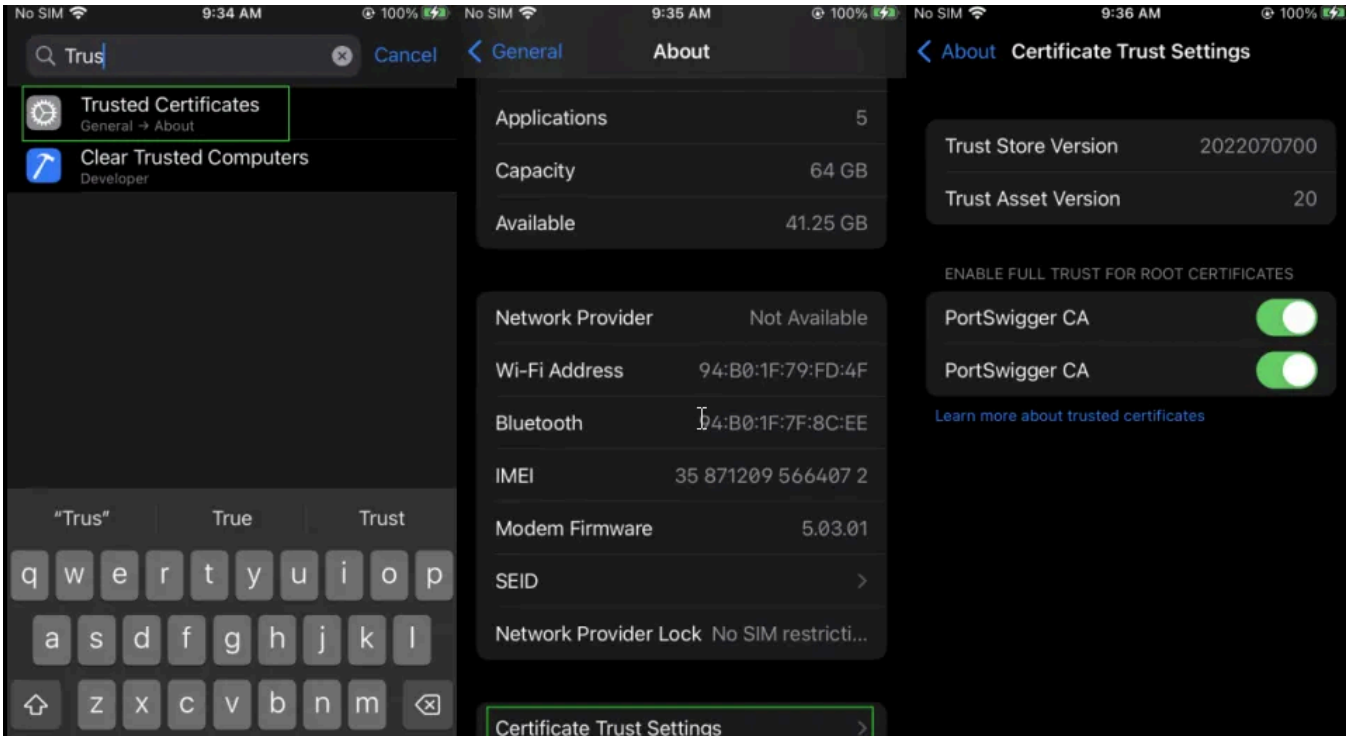
v. After downloading the certificate, Head to the device settings , there we can see “Profile Downloaded”



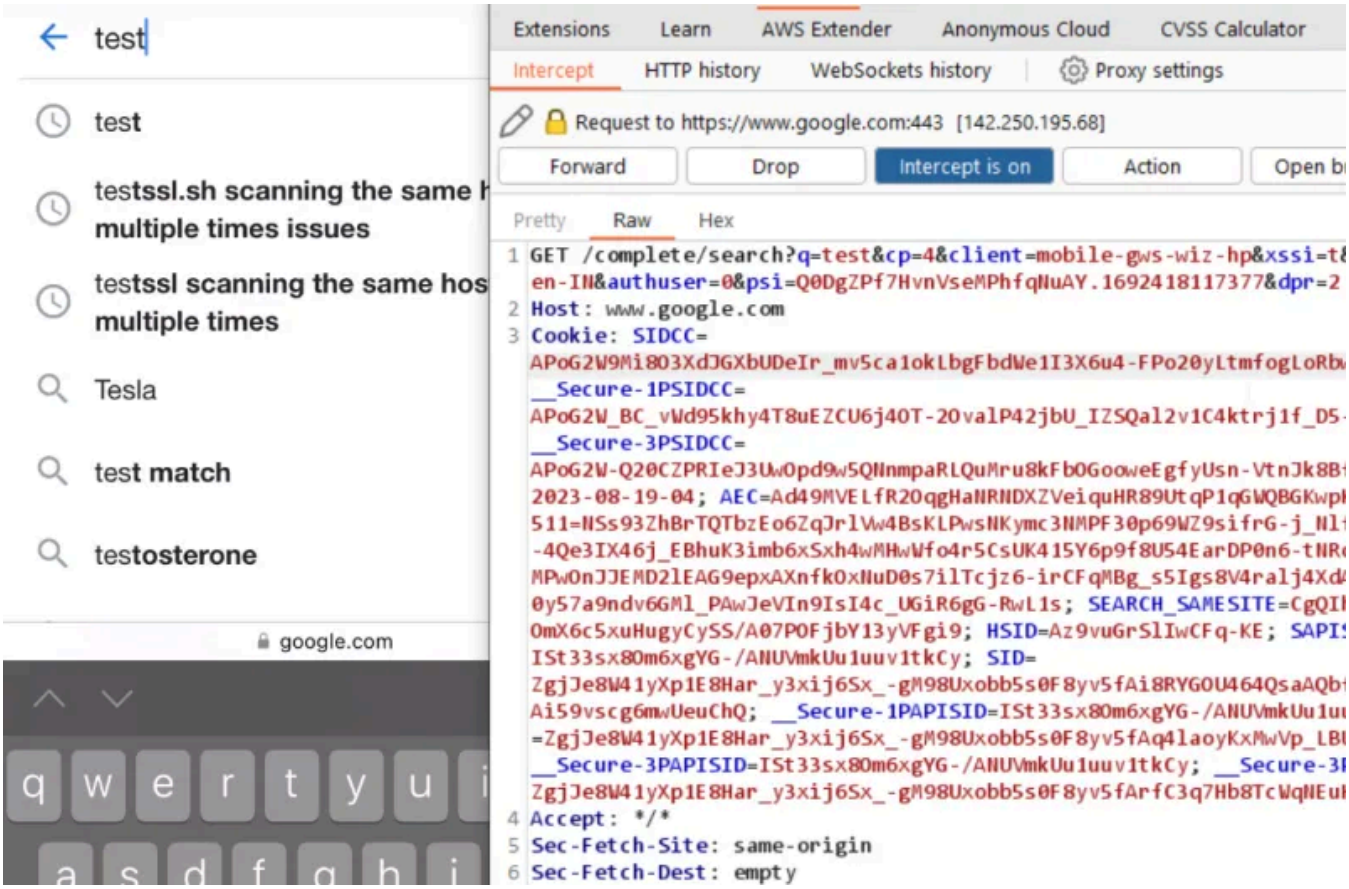
vi. Install the profile



vii. Now from the settings, Search for “Trusted cert”, Grant Full trust.



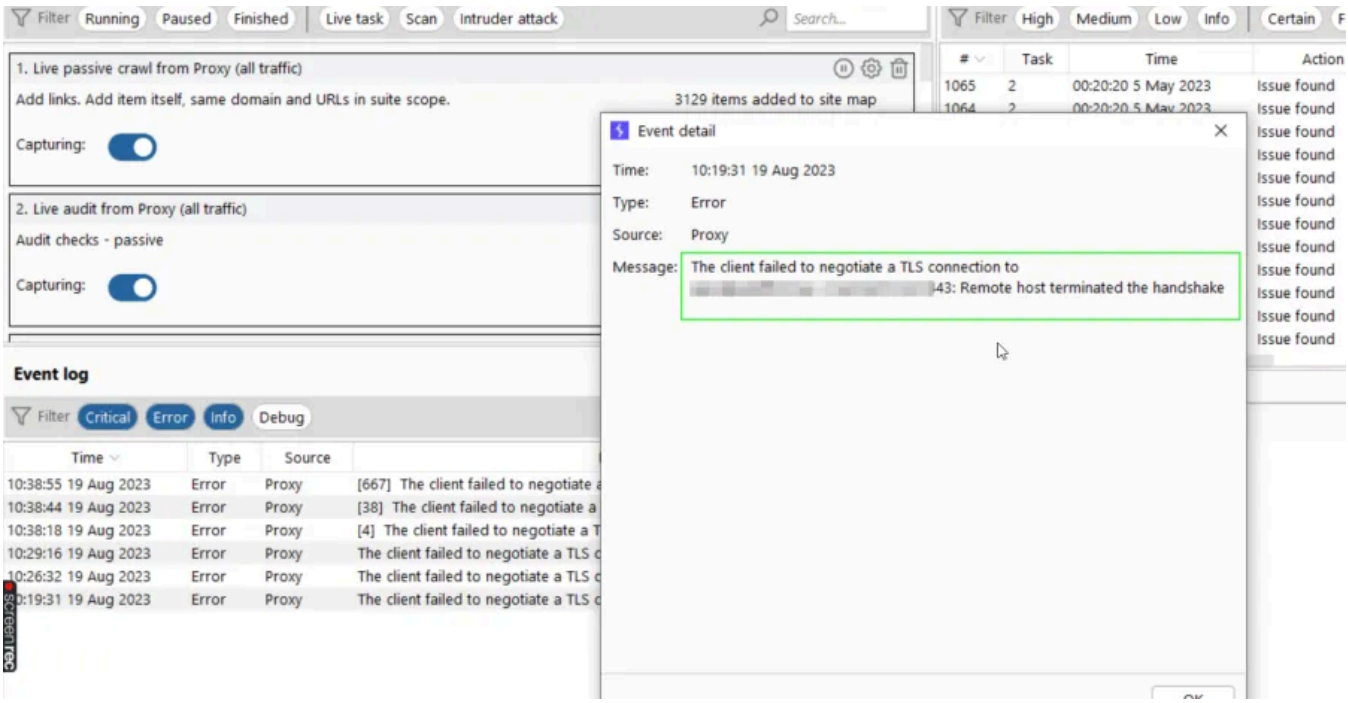
vii. Now we will be able to capture applications in which the SSL pinning is not enabled (Eg: Browser)



Ah, well, now it's time for the true antagonist to make its grand entrance — SSL certificate pinning.

1. How we can identify the chances of SSL certificate pinning?

i. After Configuring the Proxy, Fire up the target application and play with it .
ii. The application may misbehave as it is getting messed up with our fake CA certificate (Burp)
iii. Take a look at the Burp dashboard and confirm this Error



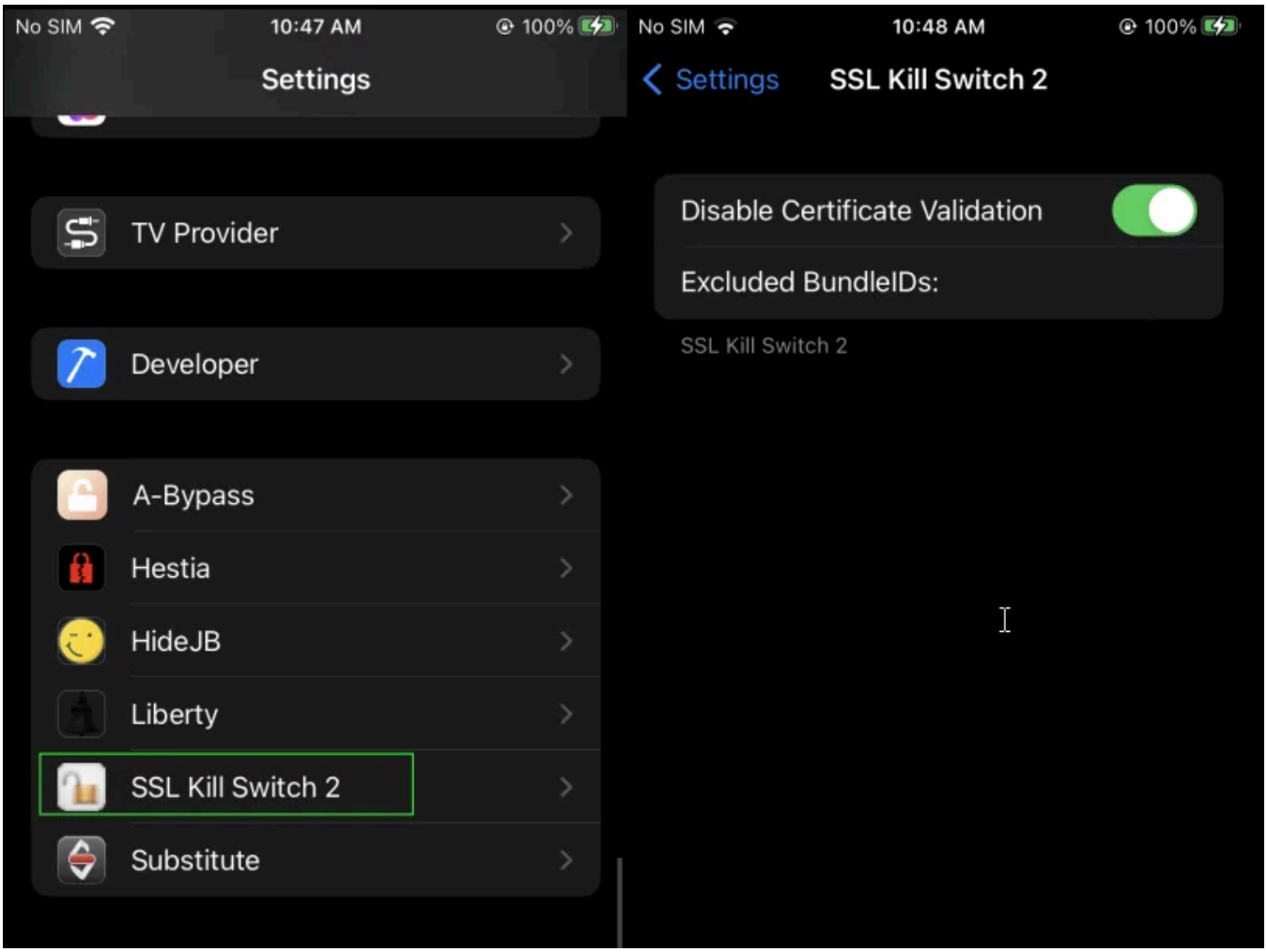
iv. This confirms the SSL pinning.

Now What? Let's try to get rid of them with some iOS tweaks:

• SSLBypass
• SSL kill Switch

i. SSL Kill Switch

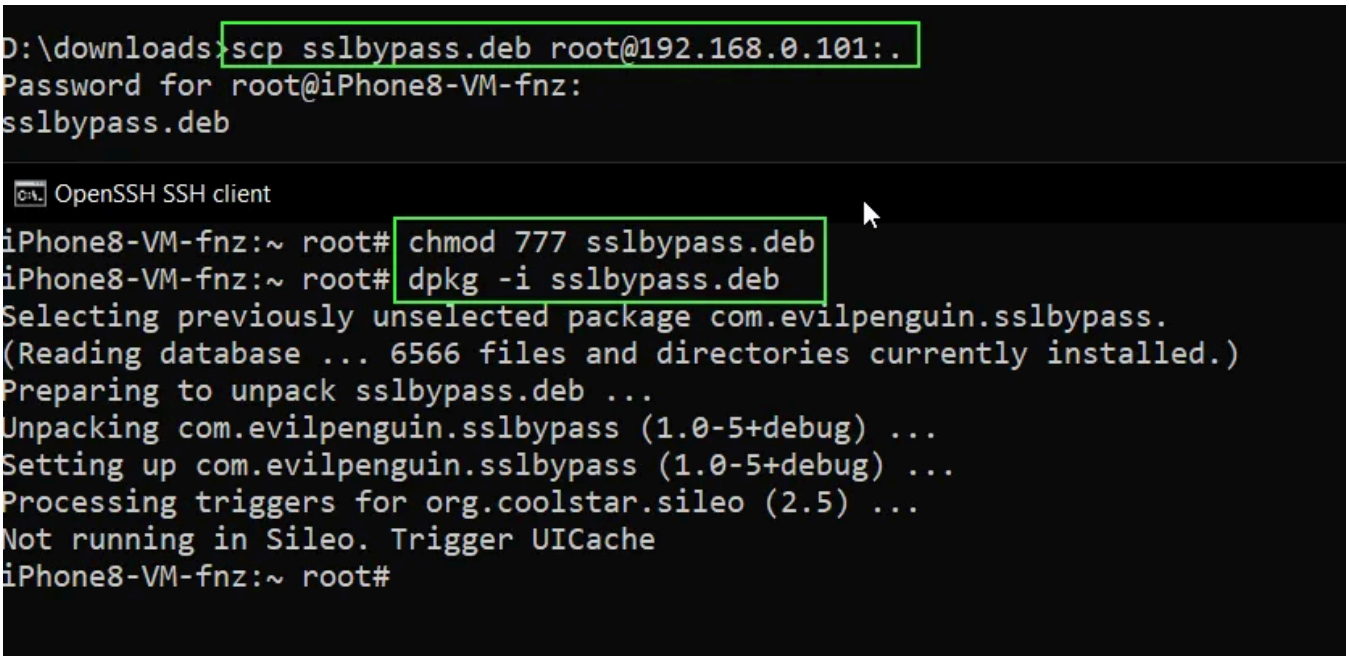
i. After enabling the SSL Kill switch, Restart the target application and make traffics.



ii. SSLBypass

You can download the binary file from : SSLBypass/packages/com.evilpenguin.sslbypass_1.0–5+debug_iphoneos-arm.deb at main · evilpenguin/SSLBypass (github.com)

And install it manually.



“Thats it! And mostly, like other ‘innocent’ tweaks, the application won’t bother showing up anywhere noticeable. Instead, it will be working its magic behind the scenes, all hush-hush. it’ll casually bypass the target application’s SSL pinning. ;)

• Similar to Android, We can also try the Frida scripts for bypassing SSL pinning:

1. https://codeshare.frida.re/@federicodotta/ios13-pinning-bypass/
2. https://codeshare.frida.re/@snooze6/ios-pinning-disable/
3. https://codeshare.frida.re/@machoreverser/ios12-ssl-bypass/

We can run the codeshare scripts as below:

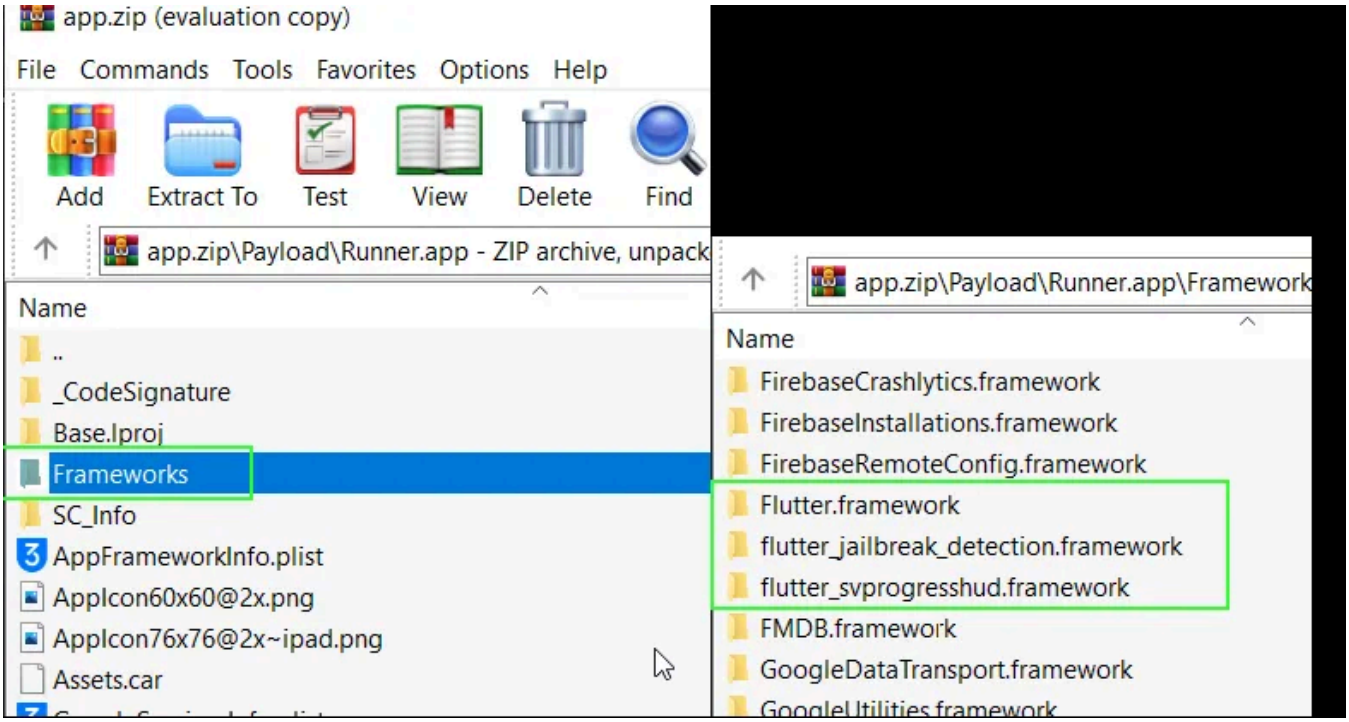
```
frida -U --codeshare <project_name> -f <target_app_package_name>
```

3. Intercepting Flutter iOS applications.

No Errors in the the Burp dashboard and can’t Intercept???

Well then, it seems I detect the distinct aroma of the Flutter framework within your application.

i. Rename the app.ipa to app.zip and look inside
ii. Find the ‘frameworks’ folder and get inside

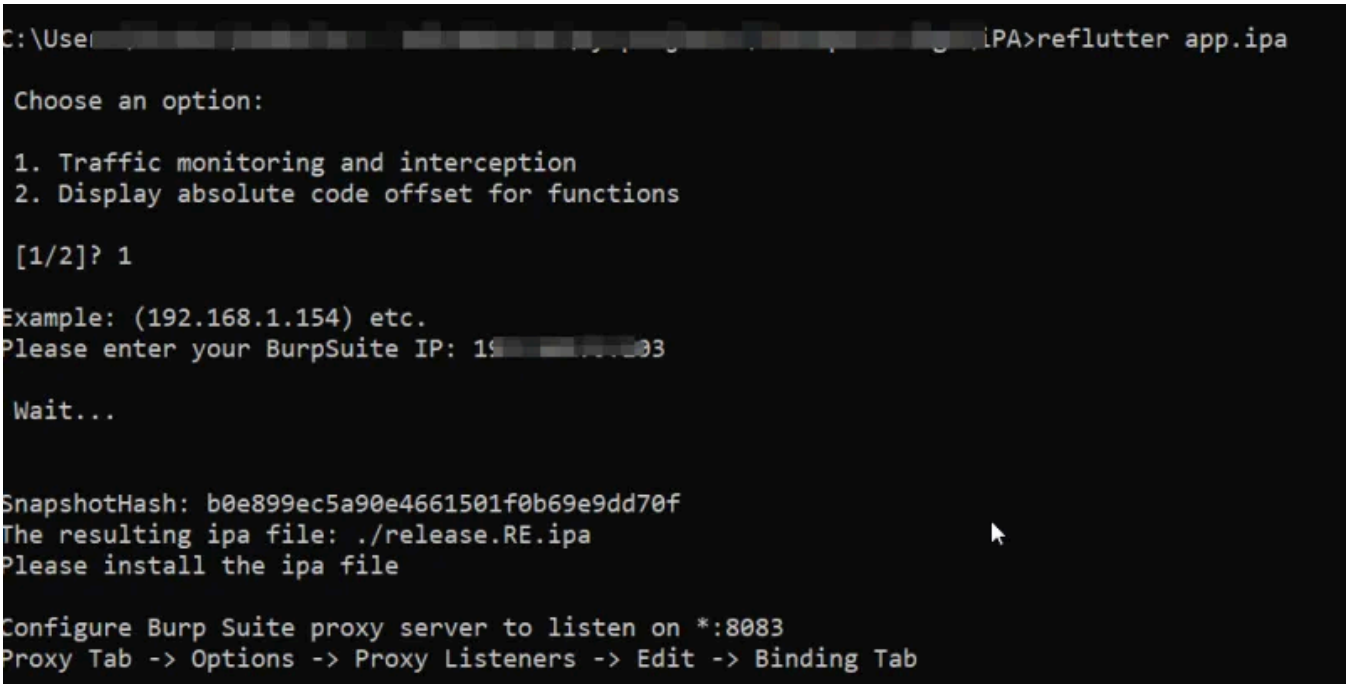


See, just as I oh-so-wisely predicted, this turned out to be one heck of a Flutter app...

Hope you remember what we just did in cases of an Android flutter application (https://medium.com/@kishorbalan/its-all-about-android-ssl-pinning-bypass-and-intercepting-proxy-unaware-applications-91689c0763d8)

“Reflutter!!, We remember you”

i. Waste no time, reflutter the IPA



It appears that I am concluding my iOS Pentesting series today. I hope it serves as a helping hand for those who have recently stepped into iOS pentesting and might be feeling a bit stuck. However, it's worth noting that this doesn't mean I'm finished with iOS pentesting altogether. There's still a wealth of iOS pentesting topics to explore. Stay tuned for more exciting content to come! 😄

Awww damn i wrote a lot today!!!

Cybersecurity IOS Pentesting Infosec Bug Bounty

Following

Written by Kishor balan

418 Followers

Security Analyst || eWPTXv2 || eJPT

More from Kishor balan

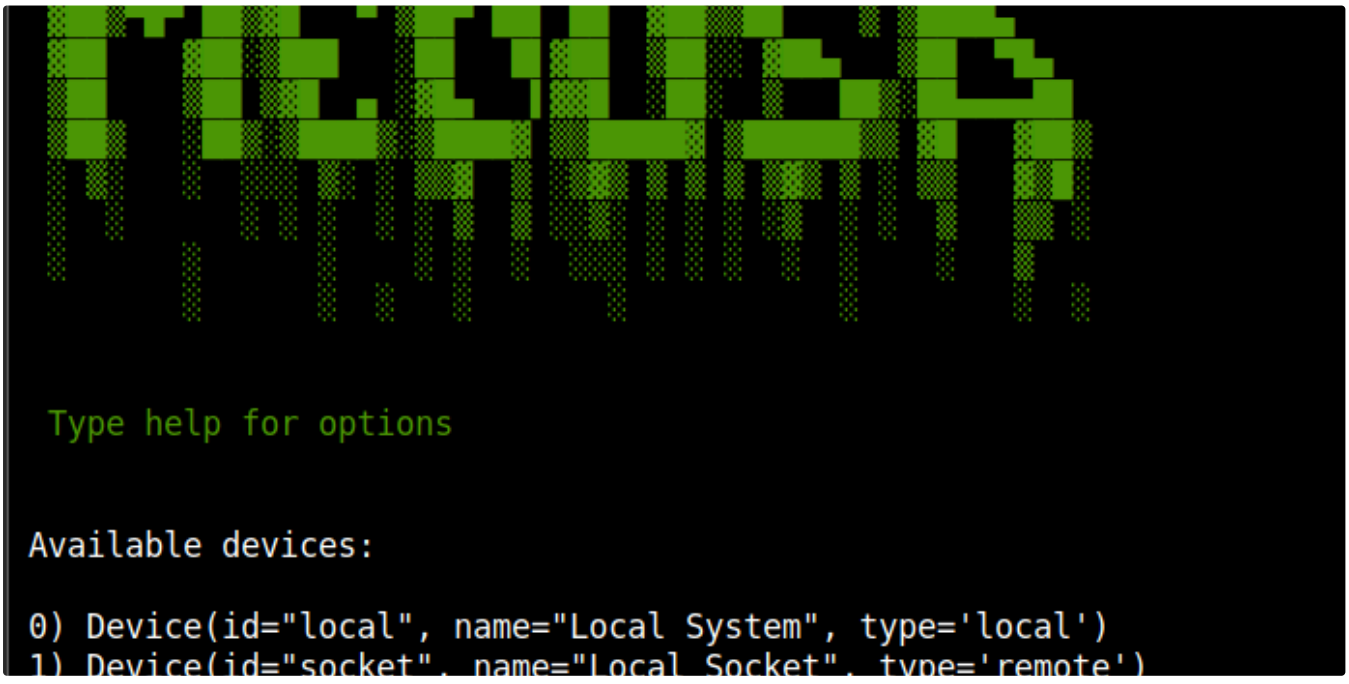
Kishor balan

It's all about Bypassing Android SSL Pinning and Intercepting Proxy Unaware applications.

Hola H3ckers,

6 min read · Nov 27, 2022

243 2



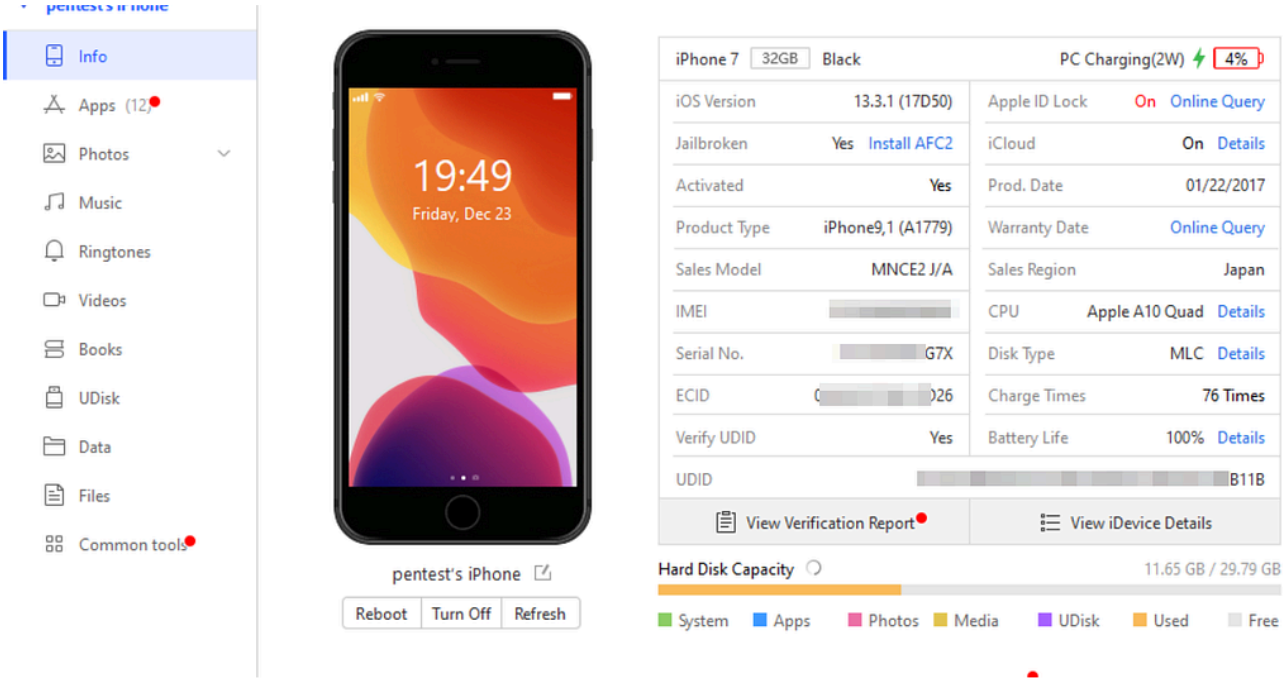
Kishor balan

My fav 7 methods for Bypassing Android Root detection

Hola H3ck3rs,

6 min read · Oct 16, 2022

185 2



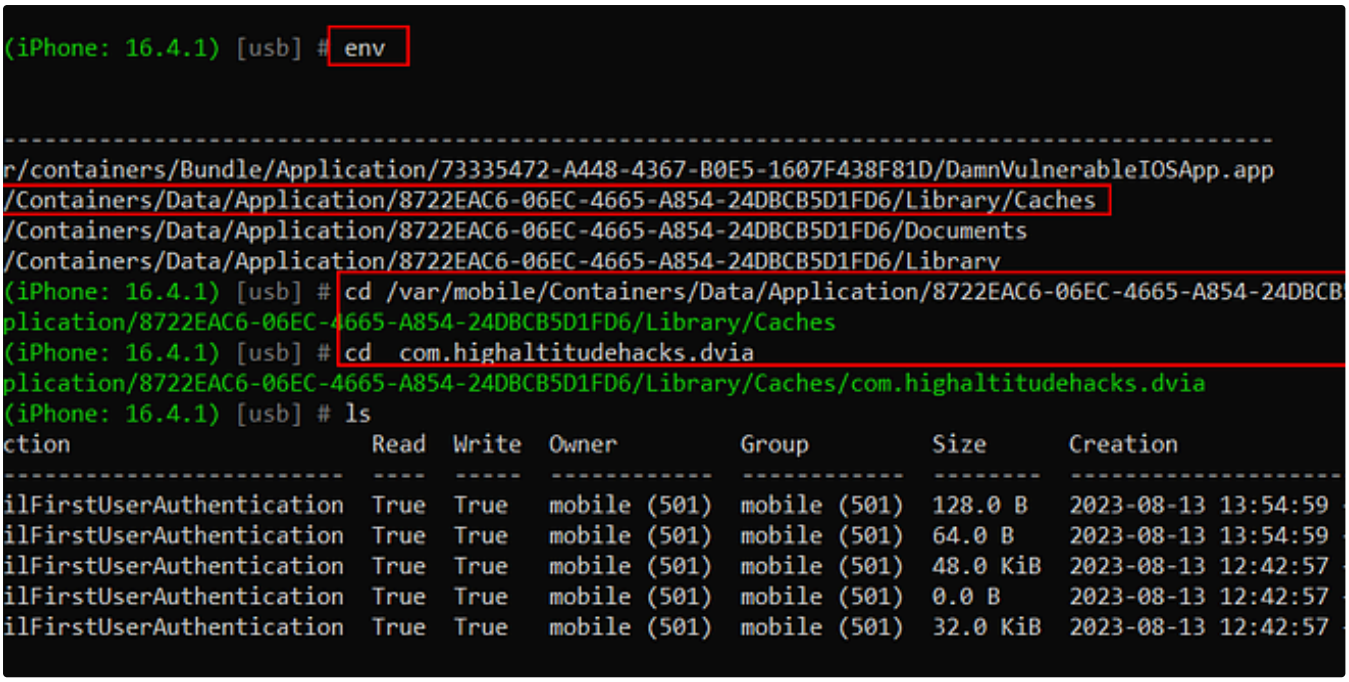
Kishor balan

Start your first iOS Application Pentest with me.. (Part- 1)

Hola Heckers,

6 min read · Jan 14, 2023

240 3



Kishor balan

iOS Pentesting Series Part 2- Into The Battlefield..

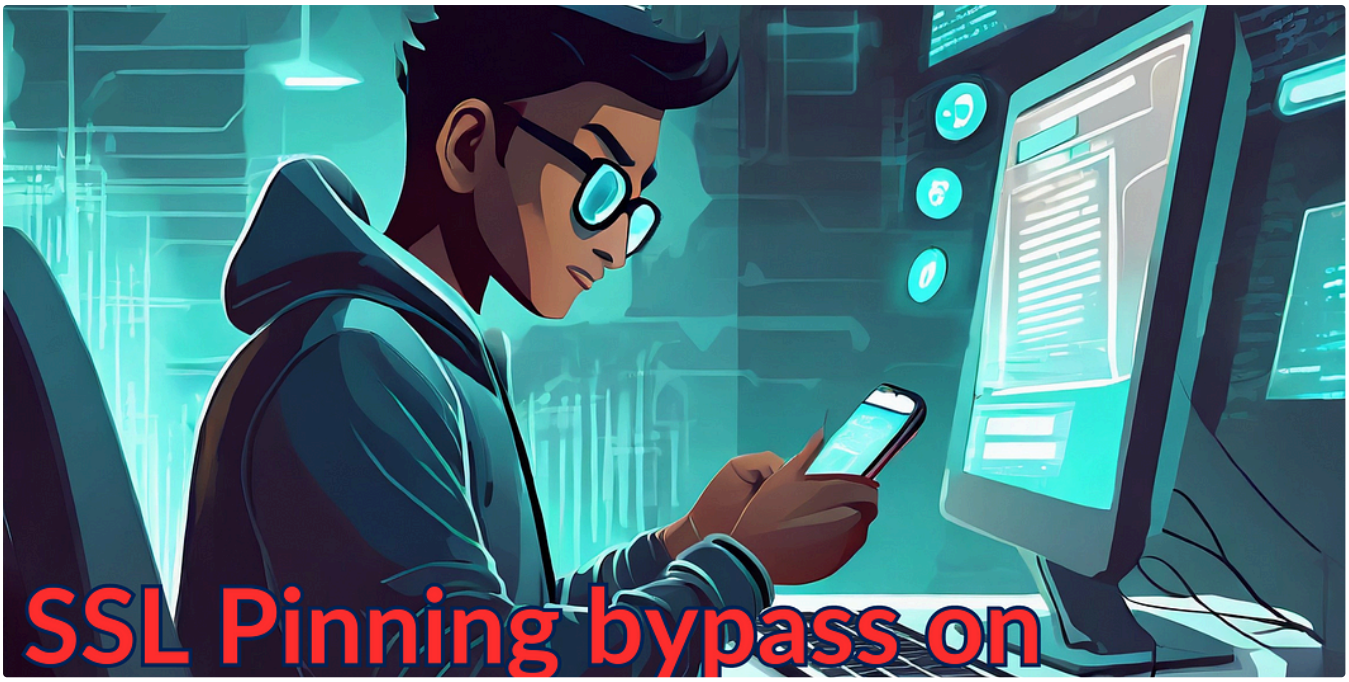
Hola Peeps,

6 min read · Aug 14, 2023

44 2

See all from Kishor balan

Recommended from Medium



Sahil Choudhary

SSL Pinning Bypass on Flutter-Based Android Apps

In the ever-evolving landscape of mobile application security, SSL pinning has become a common defense mechanism to protect sensitive data...

2 min read · Jan 3, 2024

13

Sarang Khilare

Securing Android Applications: A Comprehensive Guide to SSL Pinning Techniques

5 min read · Dec 13, 2023

Lists

Tech & Tools

16 stories · 212 saves

Apple's Vision Pro

7 stories · 65 saves

Icon Design

36 stories · 284 saves

Productivity

240 stories · 407 saves

DianaOpanga

A beginners guide to using Frida to bypass root detection.

This is a simple use case to bypass root detection using Frida. For this example we have created a sample APK that has implemented root...

3 min read · Nov 28, 2023

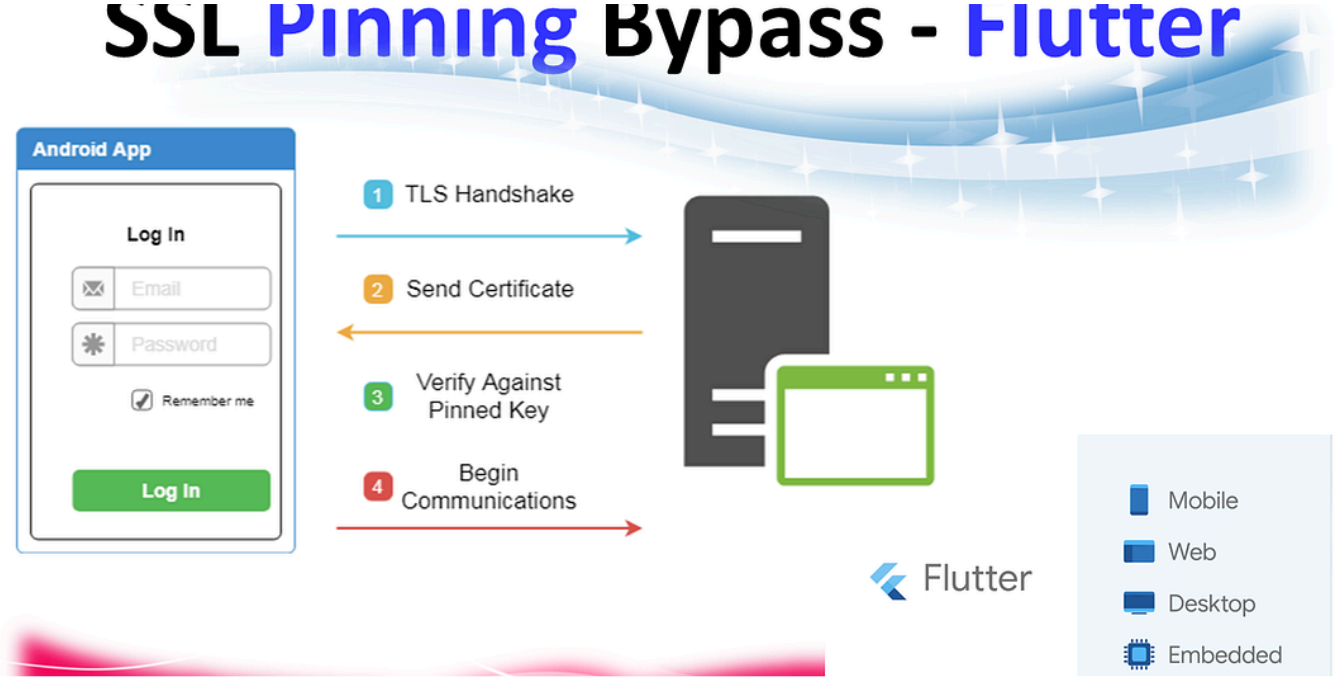
Pawan Jaiswal

Android Penetration Testing with Frida: A Comprehensive Guide with Examples

As mobile devices become increasingly integral to our daily lives, securing Android applications against potential vulnerabilities is of...

4 min read · Jan 27, 2024

19



PRASAD

Bypass SSL Pinning for Flutter

What is Flutter?

7 min read · Dec 15, 2023

138 2

Jordi Benito

SET UP AN ANDROID REVERSE ENGINEERING ENVIRONMENT

The process of reverse engineering an Android application (whether for malware analysis, testing purposes or development) requires a...

9 min read · Jan 4, 2024

See more recommendations