

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 the default Jailbreak Bypass module that is available in Objection framework.
- iv. Making our own frida script for bypassing JailBreak detection.

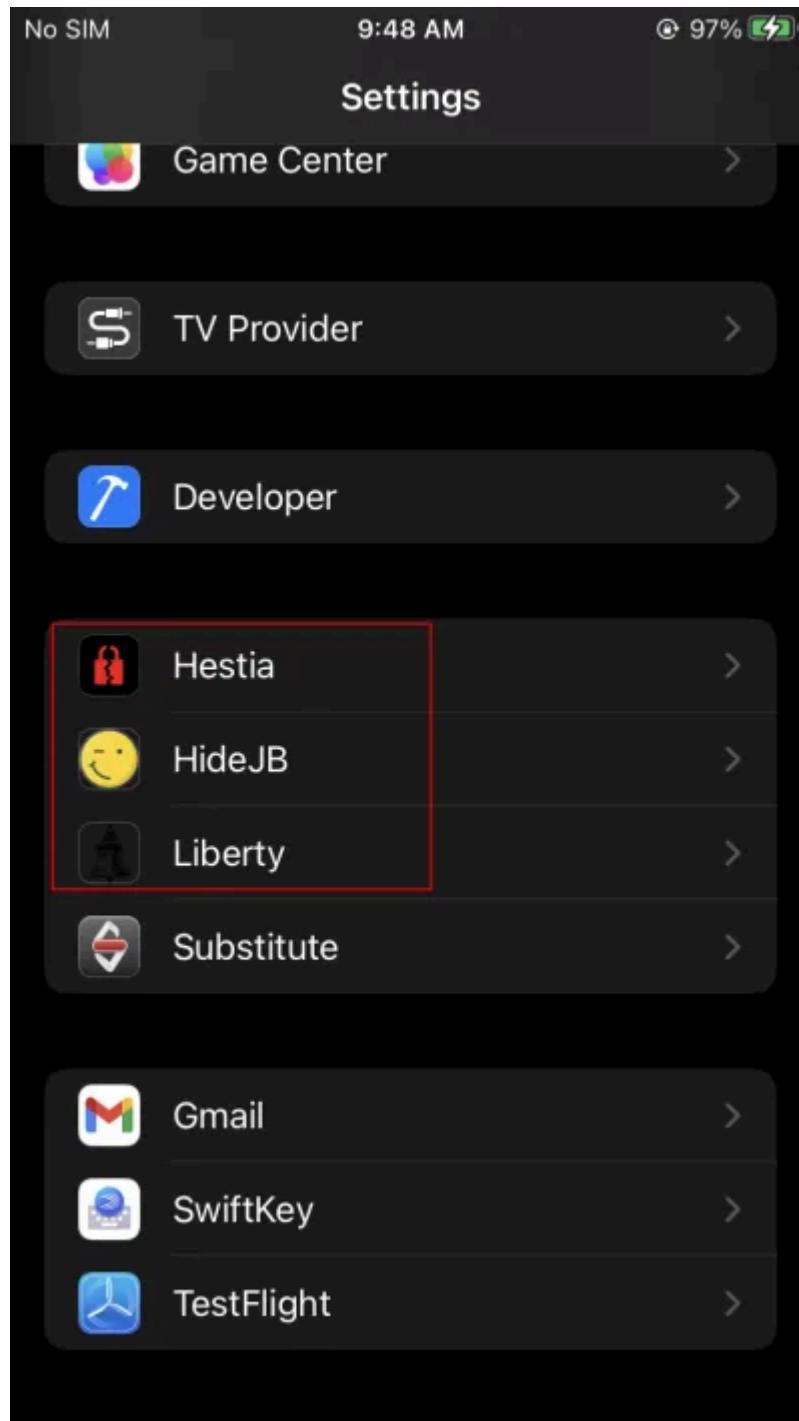
2. Setting up the Proxy and Bypassing SSL Pinning

- i. Configuring the Proxy
- ii. SSL Pinning Bypass using Tweaks
- iii. SSL Pinning Bypass Using Objection
- iv. SSL Pinning Bypass 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.



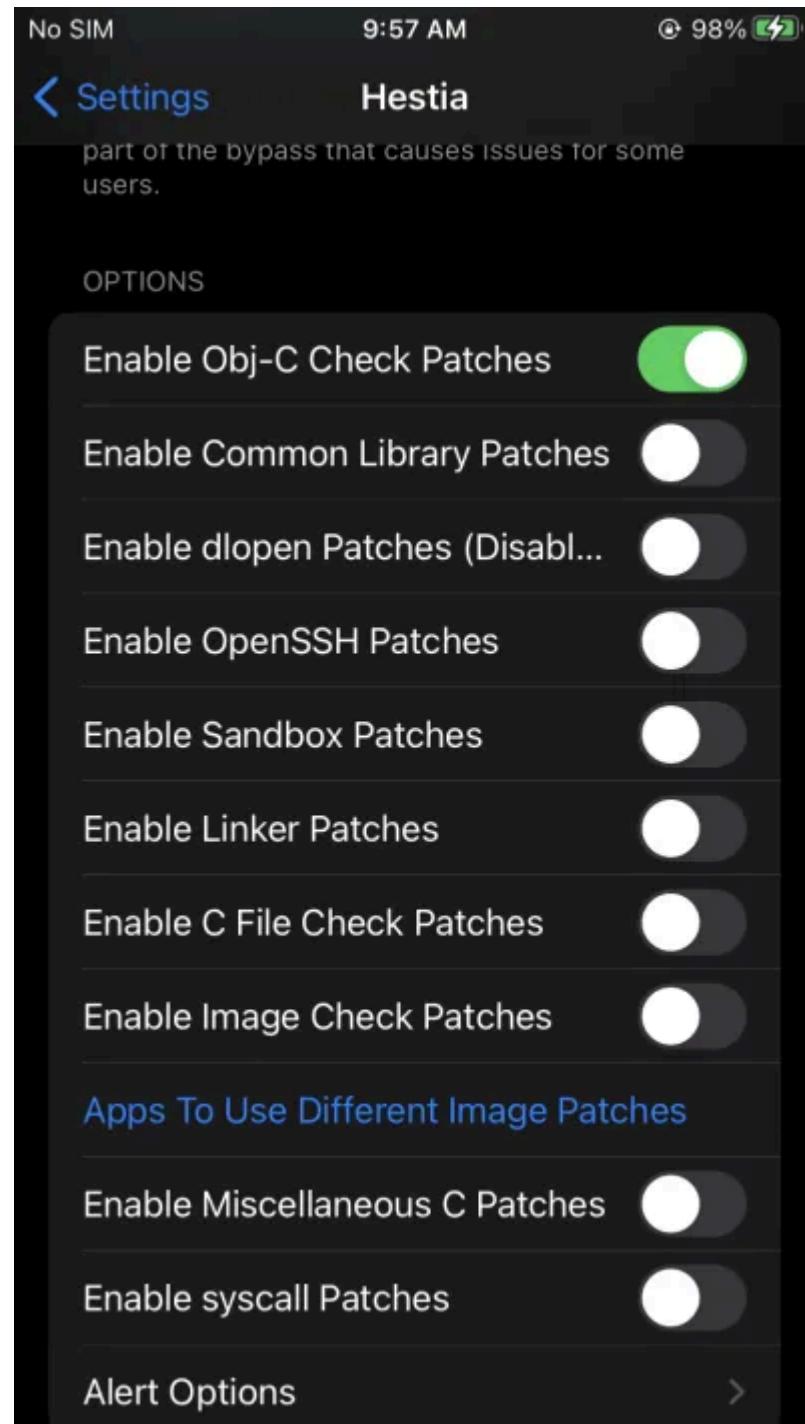
Let's try the Hestia Bypass

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.





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

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

[object] inject(v1.11.0)

Runtime Mobile Exploration
by: @leonjza from @sensepost

[tab] for command suggestions
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # ios jailbreak disable
(agent) Registering job 752611. Type: ios-jailbreak-disable
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # (agent) [752611] fileExistsAtPath: check for /bin/bash was successful with: 0x1, marking it as failed
(agent) [752611] fileExistsAtPath: check for /usr/sbin/sshd was successful with: 0x1, marking it as failed.
(agent) [752611] fileExistsAtPath: check for /etc/apt was successful with: 0x1, marking it as failed.
```

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

```
C:\Users\kishor>frida --codeshare liangxiaoyi1024/ios-jailbreak-detection-bypass -U -p 4991 --no-pause

Frida 15.2.0 - 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/

[Apple iPhone::PID::4991]-> Hooking native function stat64: /Applications/Cydia.app
Hooking native function stat: /Applications/Cydia.app
stat64 Bypass!!!
stat Bypass!!!
Hooking native function stat64: /Library/MobileSubstrate/MobileSubstrate.dylib
Hooking native function stat: /Library/MobileSubstrate/MobileSubstrate.dylib
stat64 Bypass!!!
stat Bypass!!!
Hooking native function stat64: /bin/bash
Hooking native function stat: /bin/bash
stat64 Bypass!!!
stat Bypass!!!
Hooking native function stat64: /usr/sbin/sshd
Hooking native function stat: /usr/sbin/sshd
stat64 Bypass!!!
stat Bypass!!!
Hooking native function stat64: /etc/apt
Hooking native function stat: /etc/apt
stat64 Bypass!!!
stat Bypass!!!
```

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

```
Found 1 classes
com.highaltitude
- isJailbroken
- readArticleTap
- jailbreakTest1
```

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, {
  onEnter: function (args) {
    console.log('Entering - jailbreakTest1Tapped:');
  },
  onLeave: function (retval) {
    console.log('Leaving - jailbreakTest1Tapped:');
  },
});
```

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
Modified Return value 0x0
```

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) // Replacing the return value with 0
        console.log('Modified Return value ' + retval);
    },
});
```

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

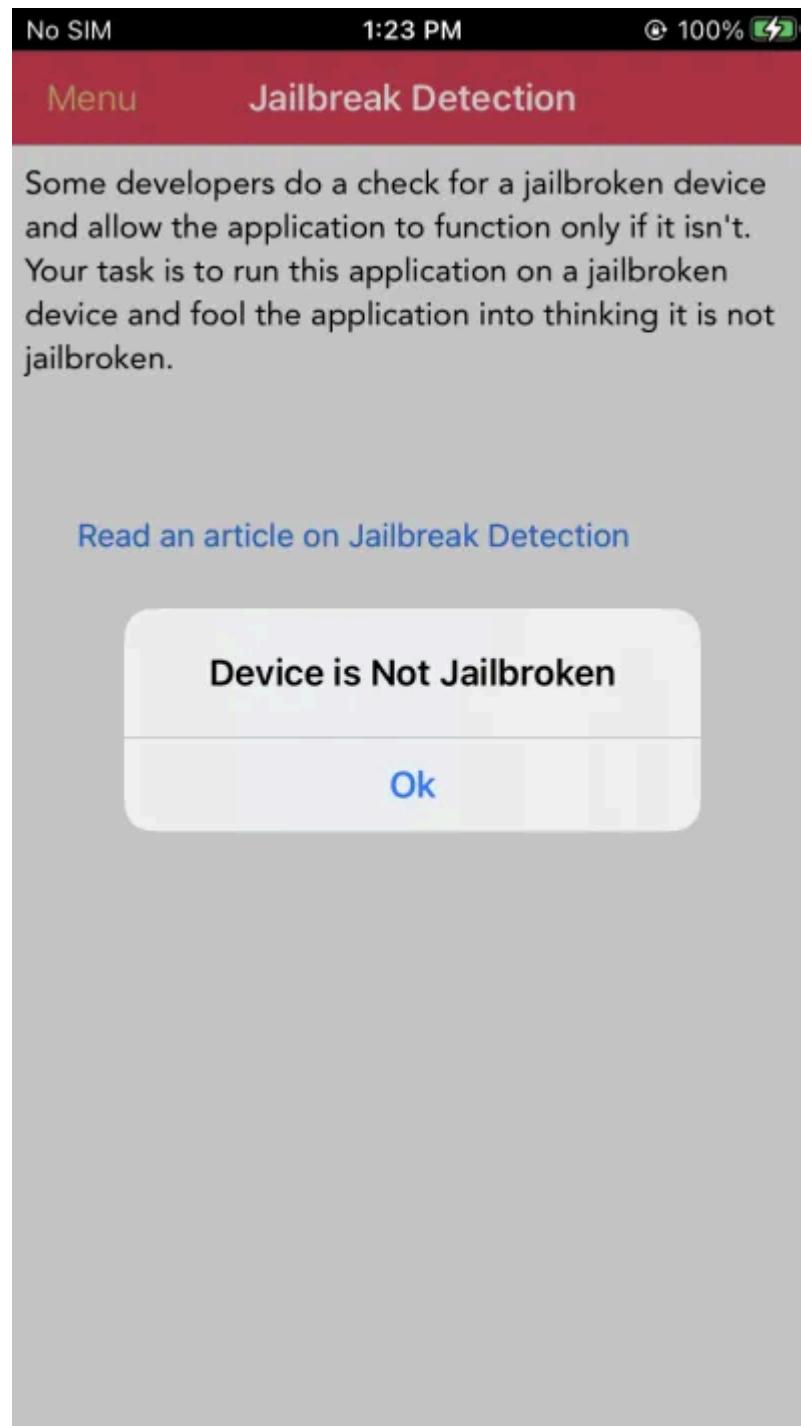
```
[object] inject/ion v1.11.0

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 // Tampered the return value
Entering - isJailbroken!
Modified Return value 0x0
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] #
```

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.



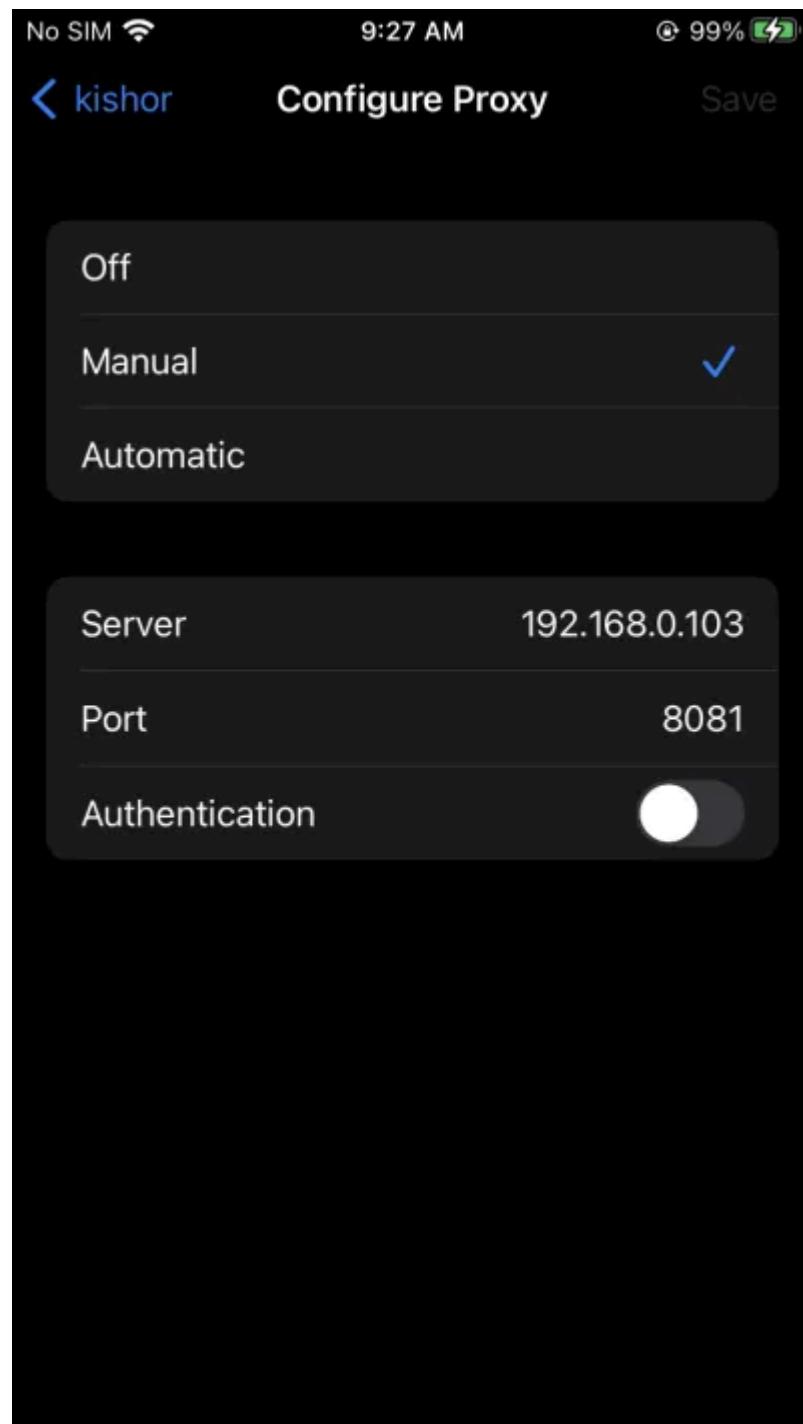
2. Setting up the Proxy and Bypassing SSL Pinning

i) Setting up proxy:

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

The screenshot shows the Burp Suite interface. At the top, there are tabs for CVSS Calculator, CSRF, Autorize, Intercept (which is underlined in red), HTTP history, WebSockets history, and Proxy settings. Below these are buttons for Forward, Drop, Intercept is off (which is greyed out), Action, and Open browser. A search bar and a magnifying glass icon are also present. On the left, a sidebar titled 'Settings' lists tools: All (selected), User, Project, Proxy (highlighted in blue), Intruder, Repeater, Sequencer, and Burp's browser. Under Project, Sessions and Network are listed. The main panel is titled 'Proxy' and shows 'Proxy listeners'. It explains that Burp Proxy uses listeners to receive incoming HTTP requests from your browser. It lists two listeners: '127.0.0.1:8080' and '*:8081'. The row for '*:8081' is highlighted with a red box. Buttons for Add, Edit, and Remove are available for managing listeners.

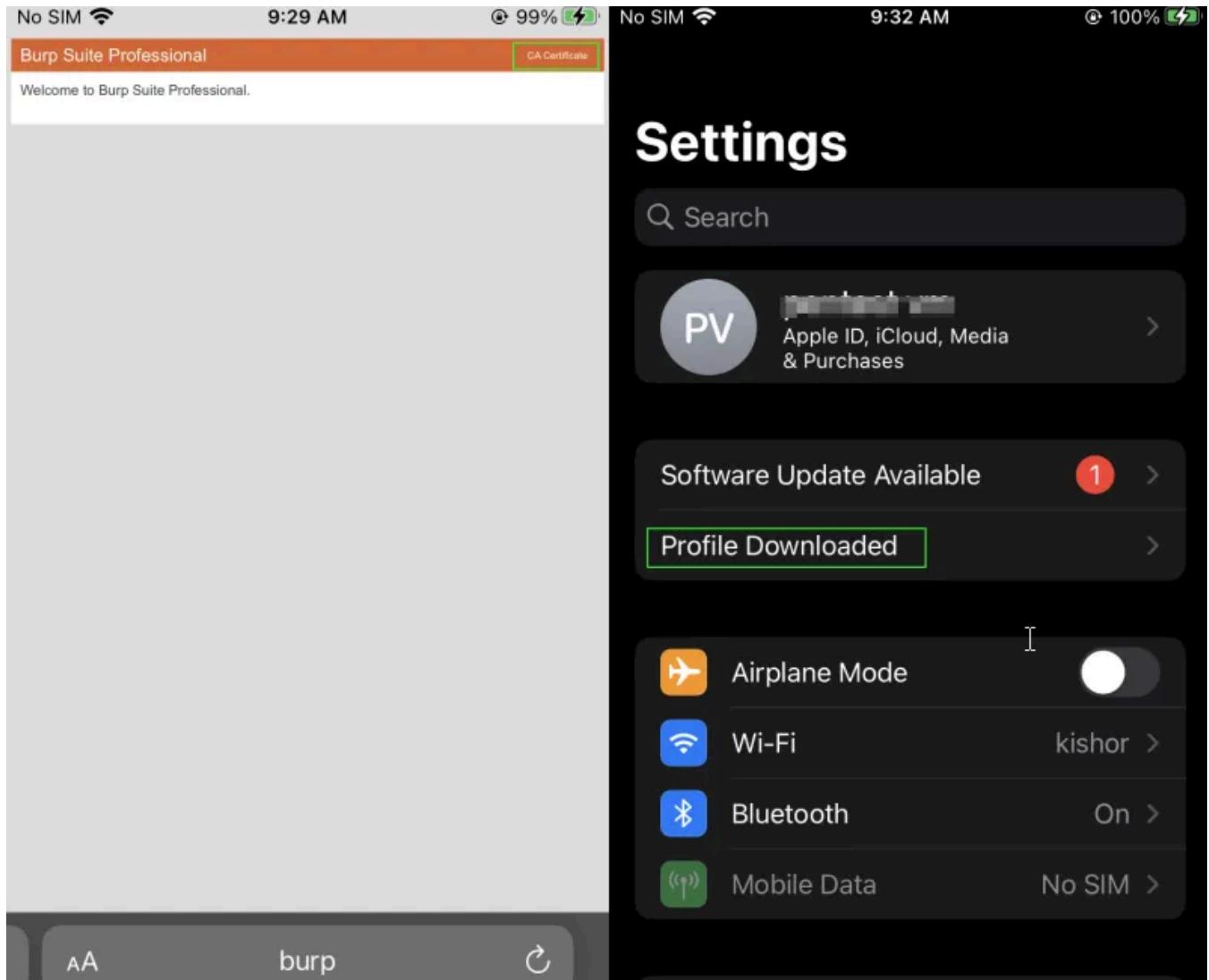
ii. Add the proxy details in the device wifi configuration



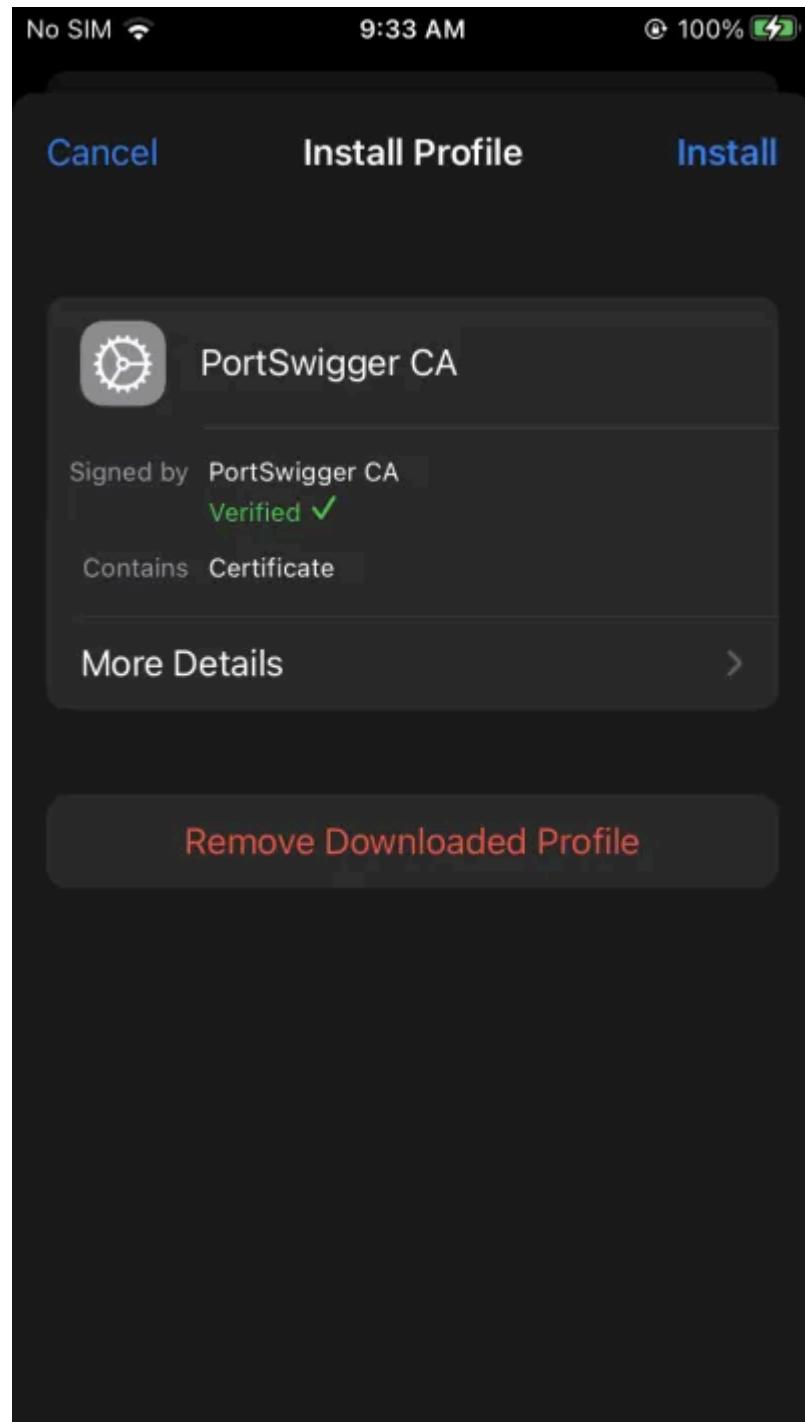
iii. Now go the device browser and hit “<http://burp>”

iv. Download the CA certificate

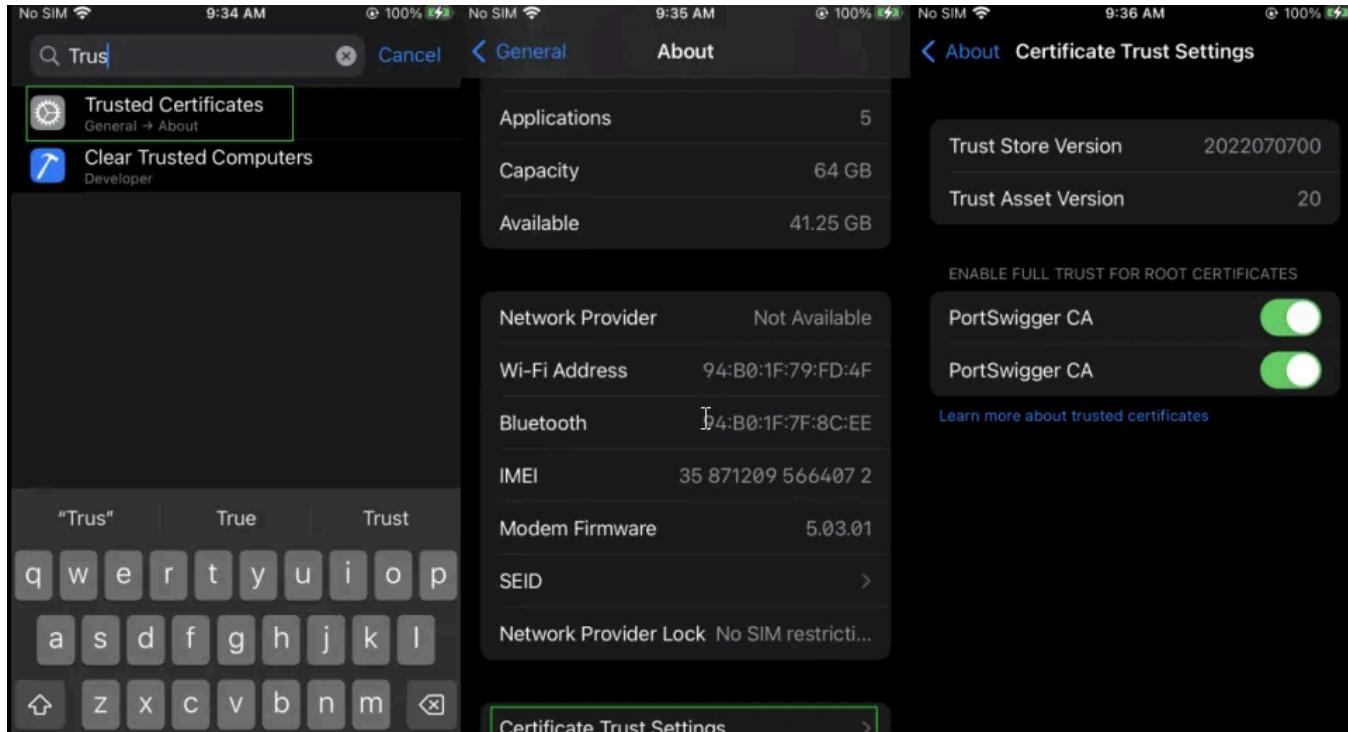
v. After downloading the certificate, Head to the device settings , there we can see “Profile Downloaded”



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)

```

GET /complete/search?q=test&cp=4&client=mobile-gws-wiz-hp&xssi=t&en-IL&authuser=0&psi=Q0DgZPf7HvnVseMPHfqNuAY.1692418117377&dpr=2
Host: www.google.com
Cookie: SIDCC=APoG2W9Mi803XjdJGXbUDeIr_mv5ca1okLbgFbdWe1I3X6u4-FPo20yLtmfogLoRb
__Secure-1PSIDCC=APoG2W_BC_vWd95khy4T8uEZCU6j40T-20valP42jbU_IZSQal2v1C4ktrj1f_D5
__Secure-3PSIDCC=APoG2W-Q20CZPRIeJ3Uw0pd9w5QInmpaRLQuMrU8kFb0GooweEgfyUsn-VtnJk8B
2023-08-19-04; AEC=Ad49MVELfR20qgHaNRNDXZVeiquHR89UtqP1qGWQBGKwpI
511=IISs93ZhBrTQTbzEo6ZqJrlVw4BsKLPwsIKymc3NMPF30p69WZ9sifrG-j_Nl
-4Qe3IX46j_EBhuK3imb6xSxh4wMhwf04r5CsUK415Y6p9f8U54EarDP0n6-tIR
MPwOnJJEMD2lEAG9epxAxnfk0xIluD0s7i1Tcjz6-irCFqMBg_s5Igs8V4ralj4Xd
0y57a9ndv6GM1_PAwJeViN9IsI4c_UGiR6gG-RwL1s; SEARCH_SAMESITE=CgQII
OmX6c5xuHugyCySS/A07POFjbjY13yVFgi9; HSID=Az9vuGrS1IwCFq-KE; SAPI
ISt33sx80m6xgYG-/ANUVmkUu1uuvitkCy; SID=
ZgjJe8W41yXp1E8Har_y3xij6Sx_-gM98Uxobb5s0F8yv5fAi8RYGOU464QsaAQBt
Ai59vscg6mwUeuChQ; __Secure-1PAPISID=ISt33sx80m6xgYG-/ANUVmkUu1uu
=ZgjJe8W41yXp1E8Har_y3xij6Sx_-gM98Uxobb5s0F8yv5fAq4laoyKxMwVp_LBI
__Secure-3PAPISID=ISt33sx80m6xgYG-/ANUVmkUu1uuvitkCy; __Secure-3I
ZgjJe8W41yXp1E8Har_y3xij6Sx_-gM98Uxobb5s0F8yv5fArFC3q7Hb8TcWqNEul
Accept: */
Sec-Fetch-Site: same-origin
Sec-Fetch-Dest: empty

```

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

The screenshot shows the Burp Suite interface with several panels:

- Proxy Panel:** Shows a live crawl from the proxy. It has sections for "1. Live passive crawl from Proxy (all traffic)" and "2. Live audit from Proxy (all traffic)". Both sections have "Capturing" toggles turned on.
- Site Map Panel:** Shows 3129 items added to the site map.
- Event Detail Panel:** Displays an event detail for an "Error" type message. The message content is highlighted with a green box: "The client failed to negotiate a TLS connection to [REDACTED] J43: Remote host terminated the handshake".
- Event Log Panel:** Shows a log of errors. One entry is visible: "10:38:55 19 Aug 2023 Error Proxy [667] The client failed to negotiate a TLS connection to [REDACTED] J43: Remote host terminated the handshake".
- Bottom Left:** A small "screen rec" icon is visible.

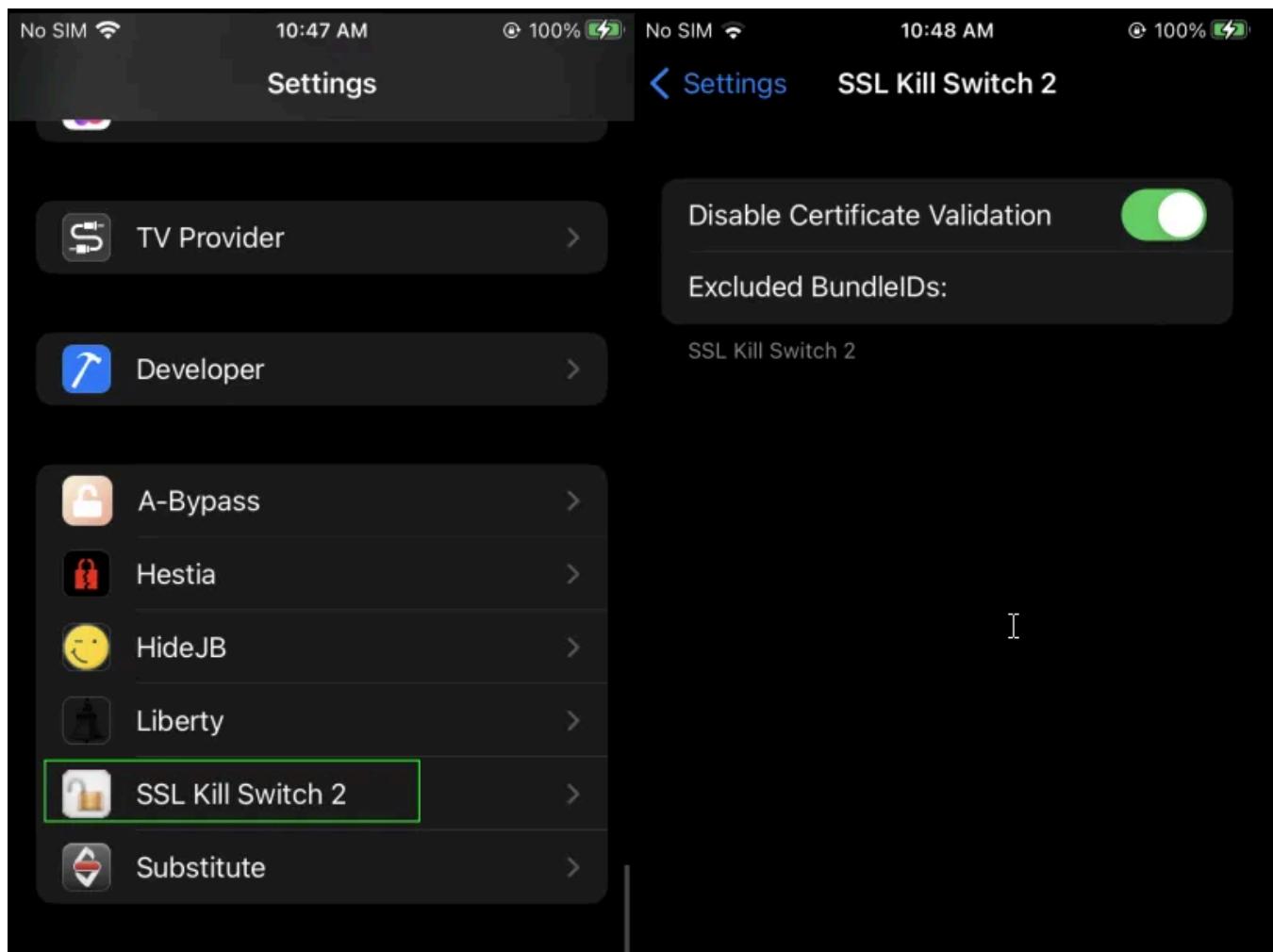
- iv. This confirms the SSL pinning.

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 \(github.com\)](https://github.com/evilpenguin/SSLBypass)

And install it manually.

```
D:\downloads>scp sslbypass.deb root@192.168.0.101:.
Password for root@iPhone8-VM-fnz:
sslbypass.deb

[!] OpenSSH SSH client
iPhone8-VM-fnz:~ root# chmod 777 sslbypass.deb
iPhone8-VM-fnz:~ root# dpkg -i sslbypass.deb
Selecting previously unselected package com.evilpenguin.sslbypass.
(Reading database ... 6566 files and directories currently installed.)
Preparing to unpack sslbypass.deb ...
Unpacking com.evilpenguin.sslbypass (1.0-5+debug) ...
Setting up com.evilpenguin.sslbypass (1.0-5+debug) ...
Processing triggers for org.coolstar.sileo (2.5) ...
Not running in Sileo. Trigger UICache
iPhone8-VM-fnz:~ root#
```

“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:

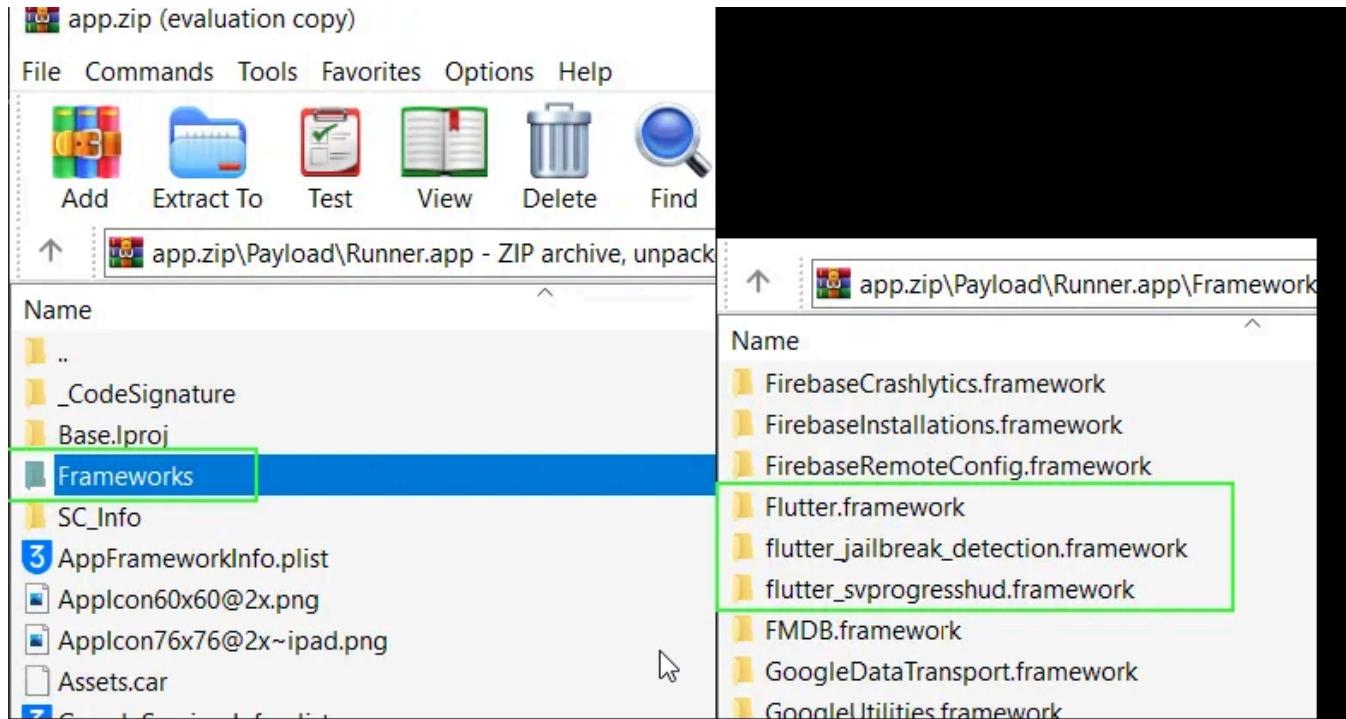
```
frida -U --codeshare <project_name> -f <target_app_package_name>
```

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

```
C:\Users\Kishor\Downloads>IPA>reflutter app.ipa
Choose an option:
1. Traffic monitoring and interception
2. Display absolute code offset for functions
[1/2]? 1
Example: (192.168.1.154) etc.
Please enter your BurpSuite IP: 192.168.1.154
Wait...
SnapshotHash: b0e899ec5a90e4661501f0b69e9dd70f
The resulting ipa file: ./release.RE.ipa
Please install the ipa file
Configure Burp Suite proxy server to listen on *:8083
Proxy Tab -> Options -> Proxy Listeners -> Edit -> Binding Tab
```

Open in app ↗



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

```
on="1.0" encoding="utf-8"?>
<security-config>
<config>
<main includeSubdomains="true">example.com</domain>
<pin-set expiration="2018-01-01">
  <pin digest="SHA-256">7HIpactkIAq2Y49orF00QKurWxmmSFZhBCoQYcRhJ3
  <!-- backup pin -->
  <pin digest="SHA-256">fwza0LRMXouZHRC8Ei+4PyuldPDcf3UKg0/04cDM1c
</pin-set>
</config>
<security-config>
```

 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



...



```
Type help for options

Available devices:

0) Device(id="local", name="Local System", type='local')
1) Device(id="socket", name="Local Socket", type='remote')
```

 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



...

```
(iPhone: 16.4.1) [usb] # env
```

```
r/containers/Bundle/Application/73335472-A448-4367-B0E5-1607F438F81D/DamnVulnerableIOSApp.app  
/Containers/Data/Application/8722EAC6-06EC-4665-A854-24DBCB5D1FD6/Library/Caches  
/Containers/Data/Application/8722EAC6-06EC-4665-A854-24DBCB5D1FD6/Documents  
/Containers/Data/Application/8722EAC6-06EC-4665-A854-24DBCB5D1FD6/Library  
(iPhone: 16.4.1) [usb] # cd /var/mobile/Containers/Data/Application/8722EAC6-06EC-4665-A854-24DBCB5D1FD6/Library/Caches  
(iPhone: 16.4.1) [usb] # cd com.highaltitudehacks.dvia  
application/8722EAC6-06EC-4665-A854-24DBCB5D1FD6/Library/Caches/com.highaltitudehacks.dvia  
(iPhone: 16.4.1) [usb] # ls
```

action	Read	Write	Owner	Group	Size	Creation
ilFirstUserAuthentication	True	True	mobile (501)	mobile (501)	128.0 B	2023-08-13 13:54:59
ilFirstUserAuthentication	True	True	mobile (501)	mobile (501)	64.0 B	2023-08-13 13:54:59
ilFirstUserAuthentication	True	True	mobile (501)	mobile (501)	48.0 KiB	2023-08-13 12:42:57
ilFirstUserAuthentication	True	True	mobile (501)	mobile (501)	0.0 B	2023-08-13 12:42:57
ilFirstUserAuthentication	True	True	mobile (501)	mobile (501)	32.0 KiB	2023-08-13 12:42:57

 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



SSL Pinning bypass on

 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

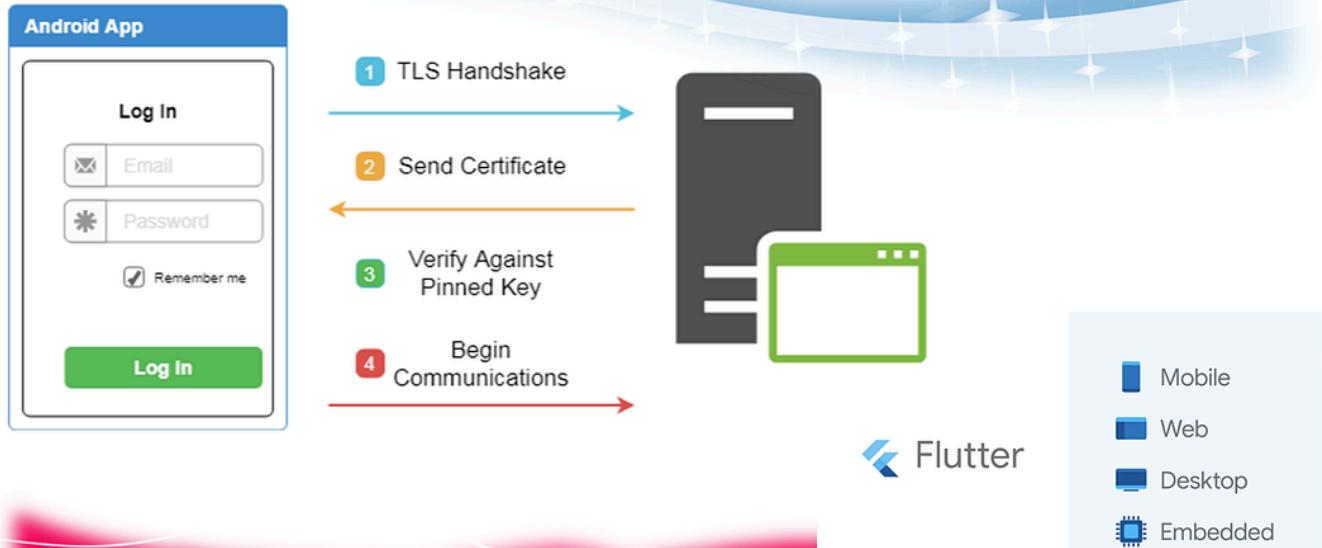
As mobile devices become increasingly integral to our daily lives, securing Android applications against potential vulnerabilities is of...

4 min read · Jan 27, 2024



...

SSL Pinning Bypass - Flutter



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