iOS Pentesting Series Part 2- Into The Battlefield..

Kishor balan · Following
6 min read · Aug 14, 2023

Listen Share ••• More

Hola Peeps,

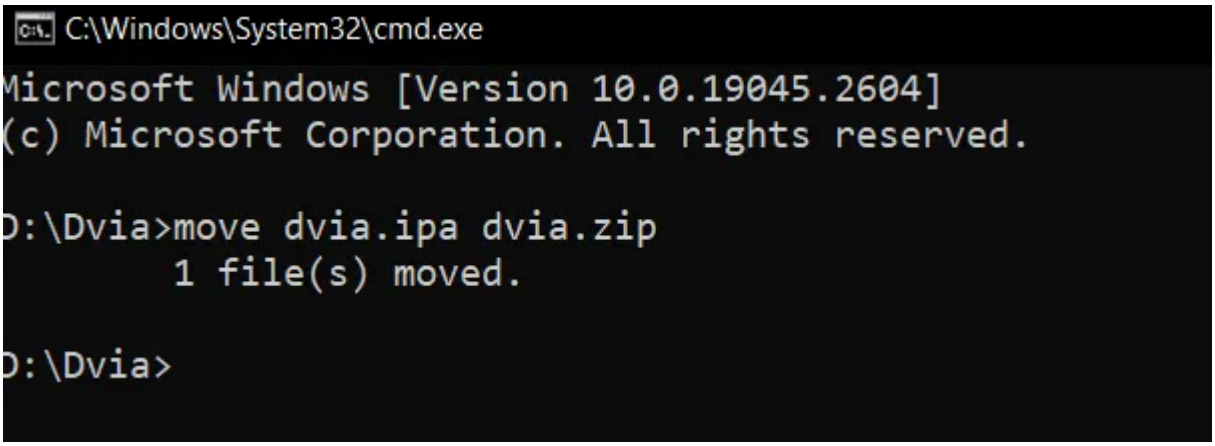
Recently, much to my surprise, I've managed to wrap up the Part-2 section of my iOS Pentesting series, all thanks to my incredibly efficient and intelligent self. Please note, just a heads up, I've kept the theory parts brief; if you're craving more in-depth knowledge, feel free to consult dear ChatGPT. xD

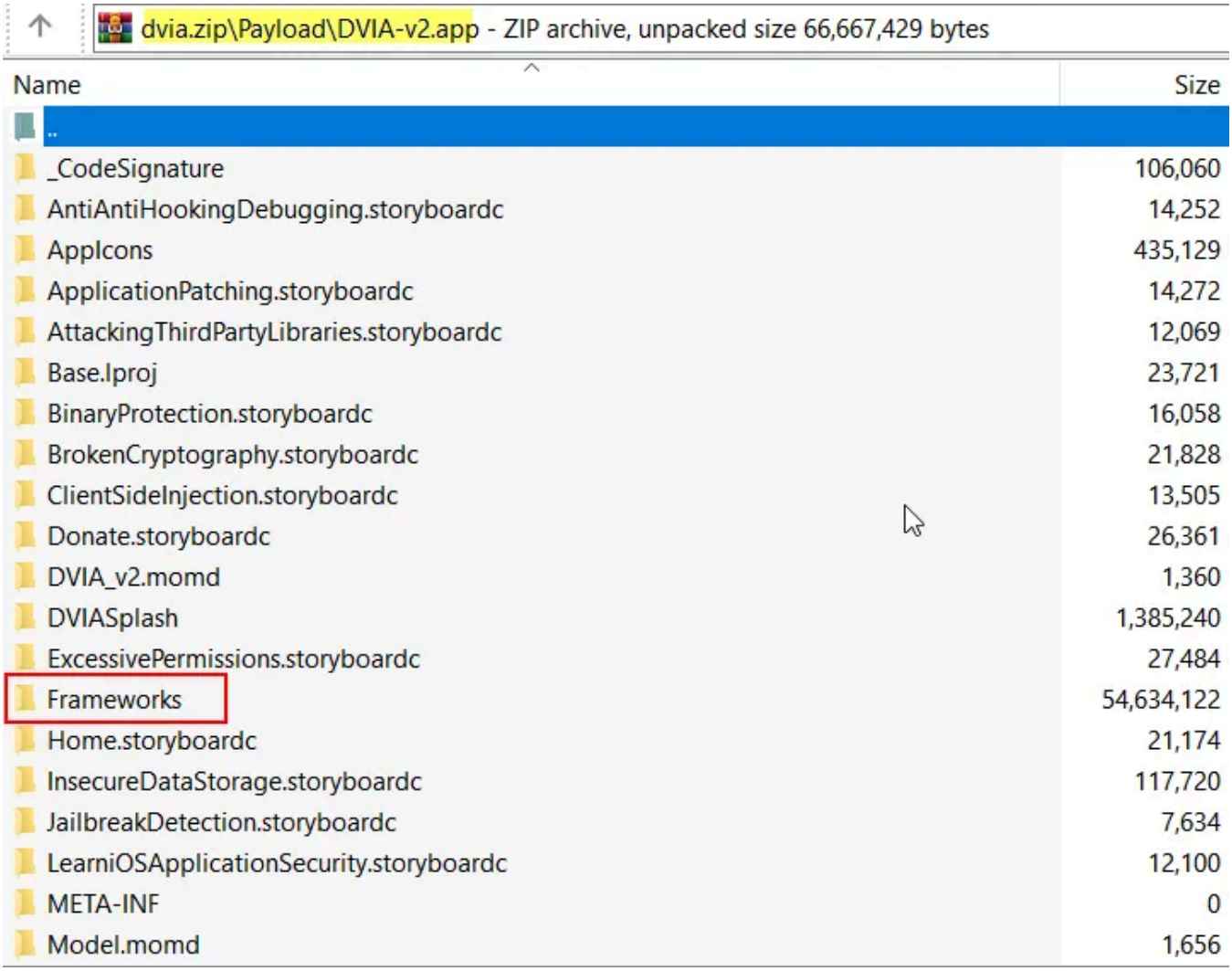
Btw, If you haven't gone through the Part-1 yet, here it is

Start your first iOS Application Pentest with me.. (Part- 1) | by Kishor balan | Medium

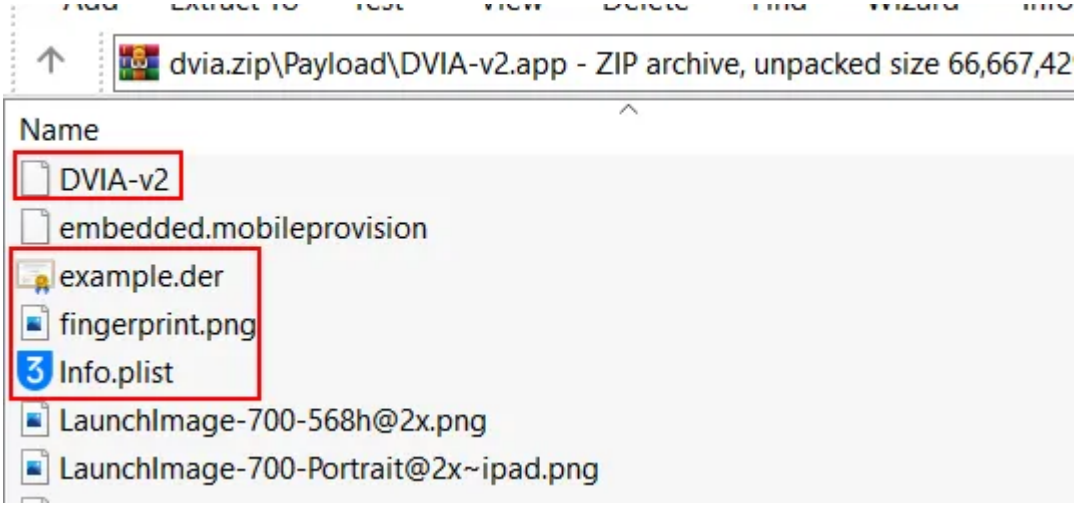
IPA file Anatomy

Rename the IPA to ZIP and extract it. Let's analyze some of the important the components.





Frameworks: This is one of the important directory that we can take a look. This folder may contains extrernal frameworks that are implemented within the application (Eg: Frameworks for Jailbreak detection, SSL pinning ,etc)

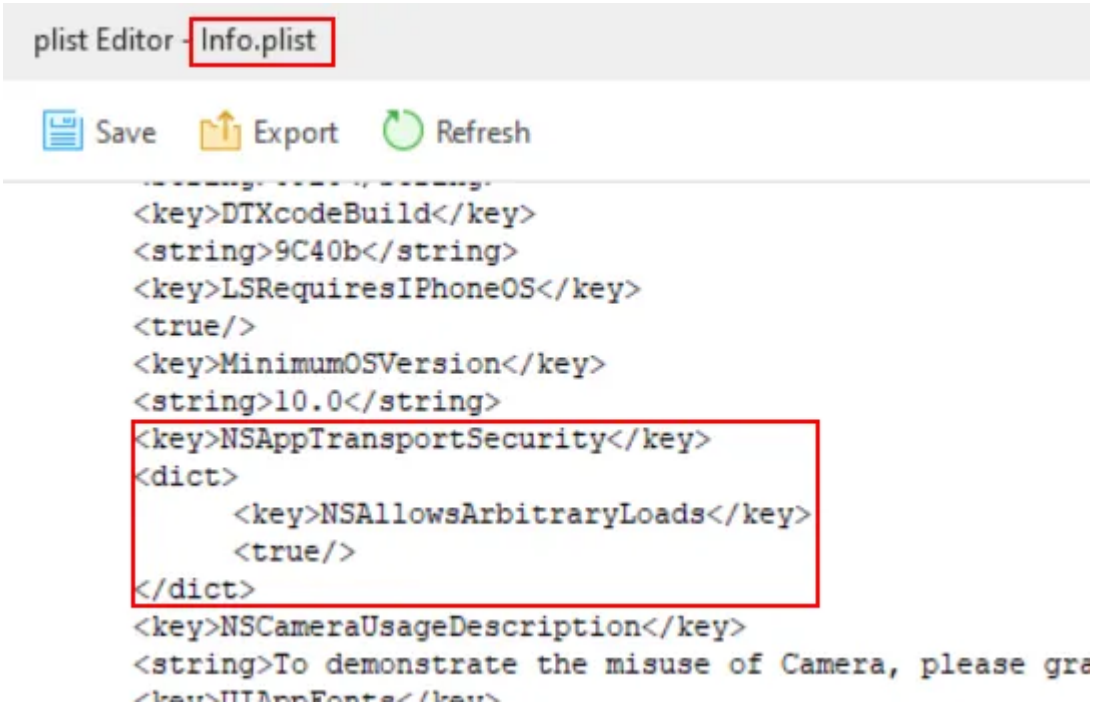


DVIA-2 — The application's executable file
example.der — CA Certificate that may use for the SSL Pinning process or any other purposes.
Info.plist -The Info. plist file contains critical information about the configuration of an iOS mobile app

Test Cases:

1: Check for App Transport Security misconfiguration.

i)Read the Info.plist file using 3uTools or any other plist readers.



The boolean value is set **TRUE** for the "**NSAllowArbitraryLoads**" property, indicates that the App Transport Policy (ATS) is globally disabled. That means the app allows insecure HTTP communication.

Note: If Exception domains are defined, in that case, The insecure HTTP loads will applicable only for that domains.

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSExceptionDomains</key>
    <dict>
        <key>example.com</key>
        <dict>
            <!-- Allow insecure HTTP connections -->
            <key>NSAllowsArbitraryLoads</key>
            <true/>
        </dict>
```

In this example, the app is allowed to communicate with "example.com" using insecure HTTP connections, despite the general ATS requirements.

2: Check for hardcoded secrets/URLs from the IPA file using Grep or ack commands.

i) Change the file extension from .ipa to .zip
ii) Extract the zip
iii) Get inside the app package

Find the hardcoded secrets.

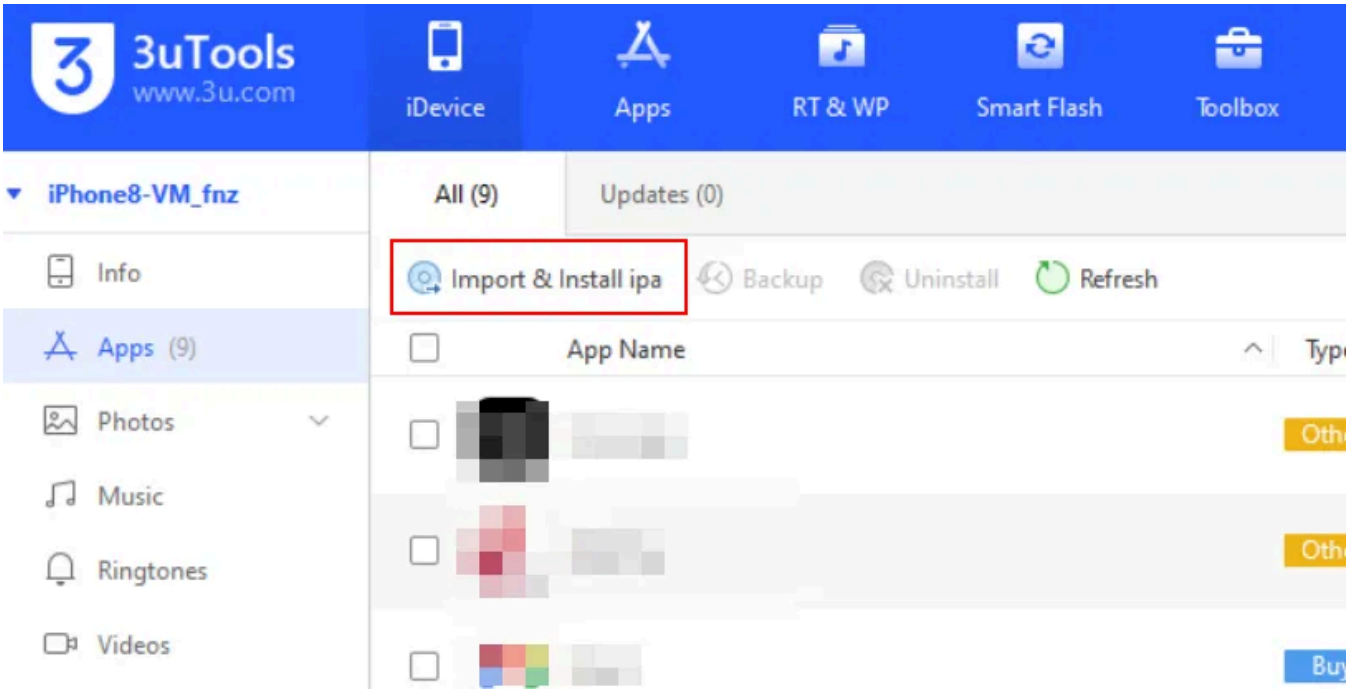
```
┌──(kali㉿kali)-[~/Payload]
└─$ ack "Secret"
[redacted]low.app/Info.plist
69:     <key>APISecretKey</key>

┌──(kali㉿kali)-[~/Payload]
└─$
```

Obatin the secret from that particular file:

```
</dict>
<key>CFBundlePackageType</key>
<key>APISecretKey</key>
<string>TTYHSJJJKI44NNABALSKSKSHTRRR</string>
<string>APPL</string>
<key>UISupportedDevices</key>
```


Unified must be installed on the iOS device for bypassing the signature checks.).



Note: If the testing IOS application is shared through Testflight, then we can directly install from it.

Insecure Storage

Checking the sensitive data in local storage

1: SharedPreference file

i) Connect to the target application using Objection and obtain its working directory using the below command.

ii) Navigate to the application's Library folder

```
[tab] for command suggestions
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # env

Name               Path
-----------------  ----------------------------------------------------------------------------------------------------
BundlePath         /private/var/containers/Bundle/Application/73335472-A448-4367-B0E5-1607F438F81D/DamnVulnerableIOSApp.app
CachesDirectory    /var/mobile/Containers/Data/Application/8722EAC6-06EC-4665-A854-24DBCB5D1FD6/Library/Caches
DocumentDirectory  /var/mobile/Containers/Data/Application/8722EAC6-06EC-4665-A854-24DBCB5D1FD6/Documents
LibraryDirectory   /var/mobile/Containers/Data/Application/8722EAC6-06EC-4665-A854-24DBCB5D1FD6/Library
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # cd /var/mobile/Containers/Data/Application/8722EAC6-06EC-4665-A854-24DBCB5D1F
/var/mobile/Containers/Data/Application/8722EAC6-06EC-4665-A854-24DBCB5D1FD6/Library/Caches
```

iii) Get into the SharedPreference folder

```
/var/mobile/Containers/Data/Application/8722EAC6-06EC-4665-A854-24DBCB5D1FD6/Library
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # ls
NSFileType  Perms  NSFileProtection  Read  Write  Owner         Group         Size    Creation                  Name
----------  -----  ----------------  ----  -----  ------------  ------------  ------  ------------------------  ----------------------
Directory     493  n/a               True  True   mobile (501)  mobile (501)  96.0 B  2023-08-11 07:08:47 +0000  Caches
Directory     493  n/a               True  True   mobile (501)  mobile (501)  96.0 B  2023-08-13 12:42:57 +0000  Saved Application State
Directory     493  n/a               True  True   mobile (501)  mobile (501)  96.0 B  2023-08-13 12:42:57 +0000  HTTPStorages
Directory     493  n/a               True  True   mobile (501)  mobile (501)  96.0 B  2023-08-13 12:44:51 +0000  SplashBoard
Directory     493  n/a               True  True   mobile (501)  mobile (501)  96.0 B  2023-08-13 12:42:57 +0000  Private Documents
Directory     493  n/a               True  True   mobile (501)  mobile (501)  96.0 B  2023-08-11 07:08:47 +0000  Preferences

Readable: True  Writable: True
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # cd Preferences
/var/mobile/Containers/Data/Application/8722EAC6-06EC-4665-A854-24DBCB5D1FD6/Library/Preferences
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # ls
NSFileType  Perms  NSFileProtection                    Read  Write  Owner         Group         Size     Creation                  Name
----------  -----  ---------------------------------  ----  -----  ------------  ------------  -------  ------------------------  ------------------------------
Regular       384  CompleteUntilFirstUserAuthentication  True  True   mobile (501)  mobile (501)  405.0 B  2023-08-13 13:55:02 +0000  com.highaltitudehacks.dvia.plist
```

iv) Read the Prefernce file using below command.

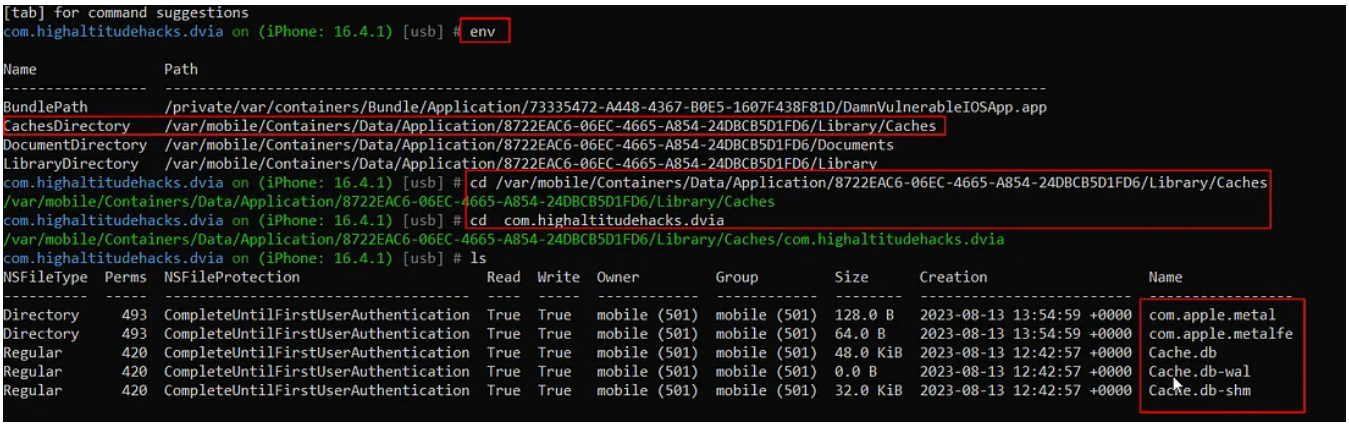
```
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # cd Preferences
/var/mobile/Containers/Data/Application/8722EAC6-06EC-4665-A854-24DBCB5D1FD6/Library/Preferences
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # ios plist cat com.highaltitudehacks.dvia.plist
{
    WebDatabaseDirectory = "/var/mobile/Containers/Data/Application/8722EAC6-06EC-4665-A854-24DBCB5D1FD6/Library/Caches";
    WebKitLocalStorageDatabasePathPreferenceKey = "/var/mobile/Containers/Data/Application/8722EAC6-06EC-4665-A854-24DBCB5D1FD6/Library/Caches";
    WebKitOfflineWebApplicationCacheEnabled = 1;
    WebKitShrinksStandaloneImagesToFit = 1;
}
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # exit
```

(Sorry, No sensitive data is found in my current target application)

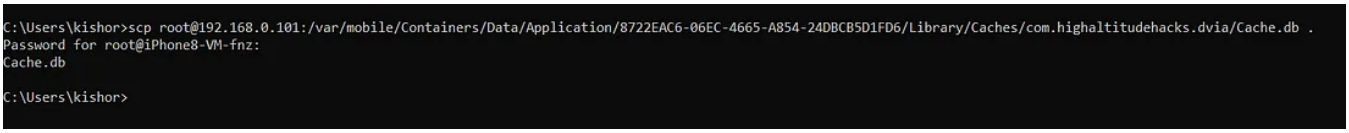
2: Local DB files

i) Connect the target application with objection

ii) Find the application path in which the cache files gets stored sing the below command and navigate to that path

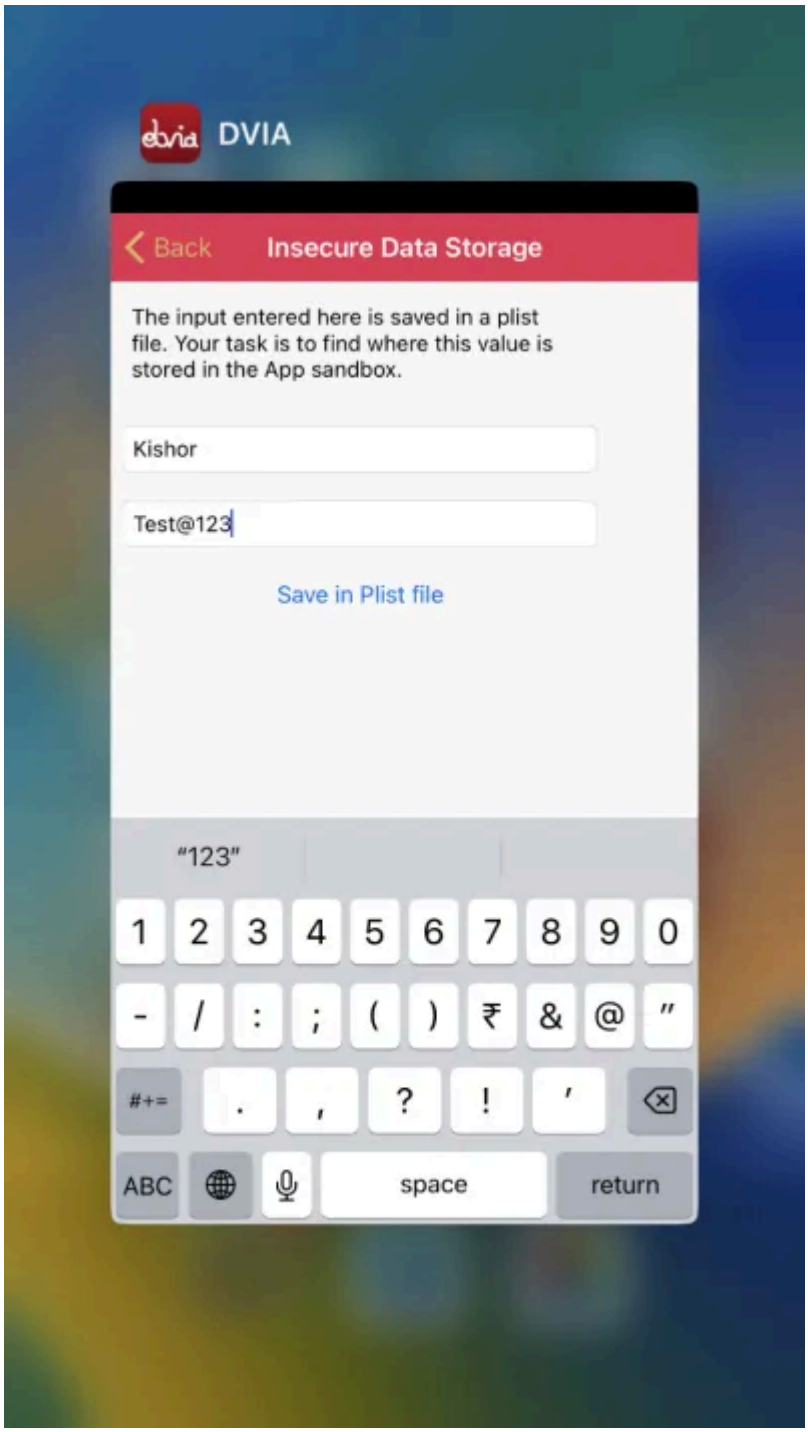


iii) Copy the “Cache.db” to your local machine with the scp command



iv) Explore the db file using any DB browsers and look for the sensitive data

3: Background screencaching: (Lol i guess i dont wanna spit out a deep explanation for this, Its just the thing when the app being sent to the background state. If the app screen is visible, then it is vulnerable)



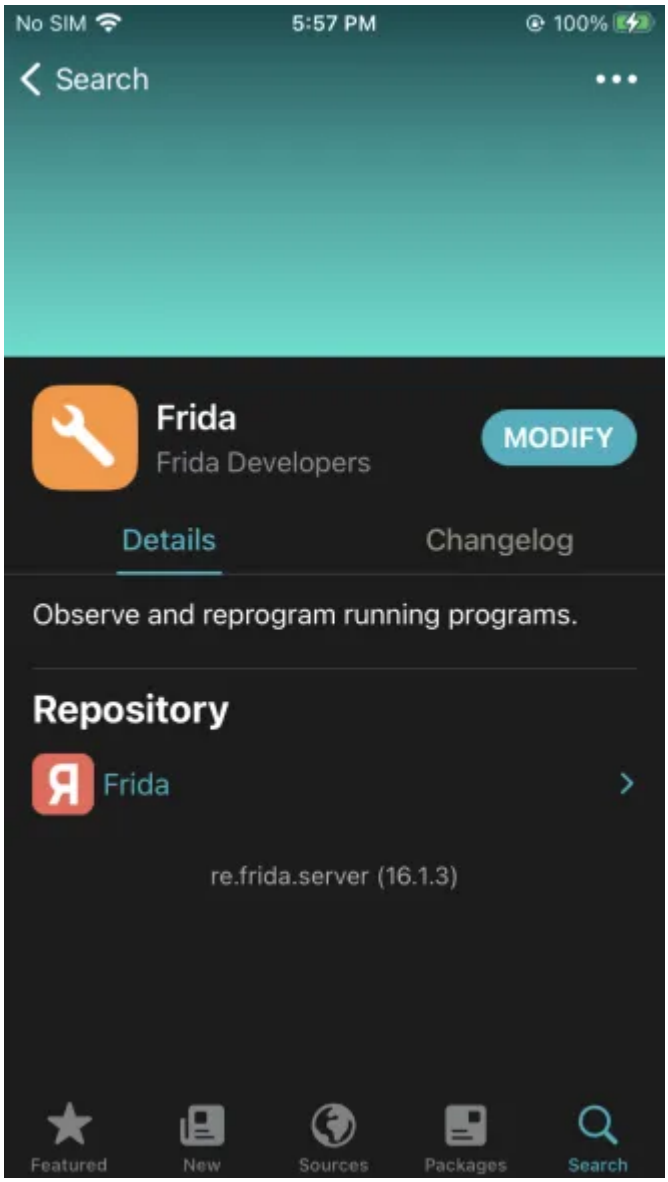
4: Checking sensitive data in Heap memory:

sensitive data such as login credentials / OTP/ security PIN codes/ Session Tokens / PII data

Tool: Fridummp

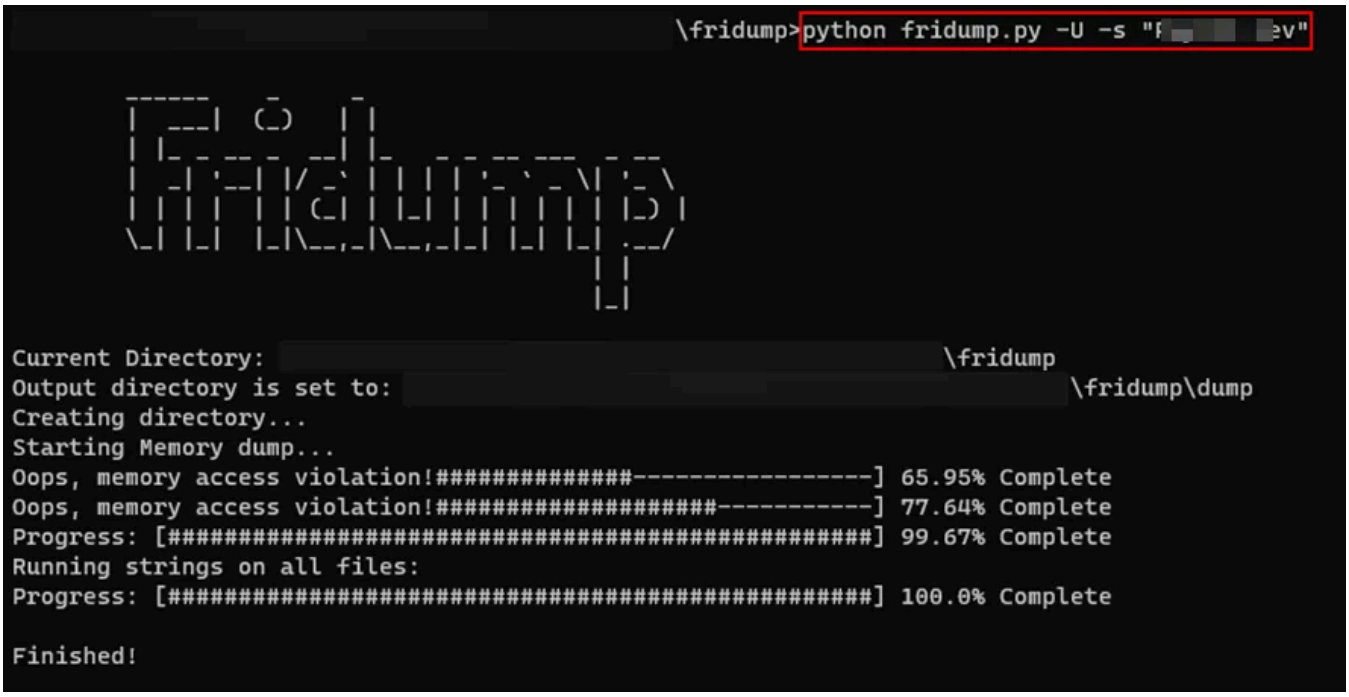
GitHub — Nightbringer21/fridump: A universal memory dumper using Frida

i) Make sure the frida tweak is installed and running in your Cydia/Sileo

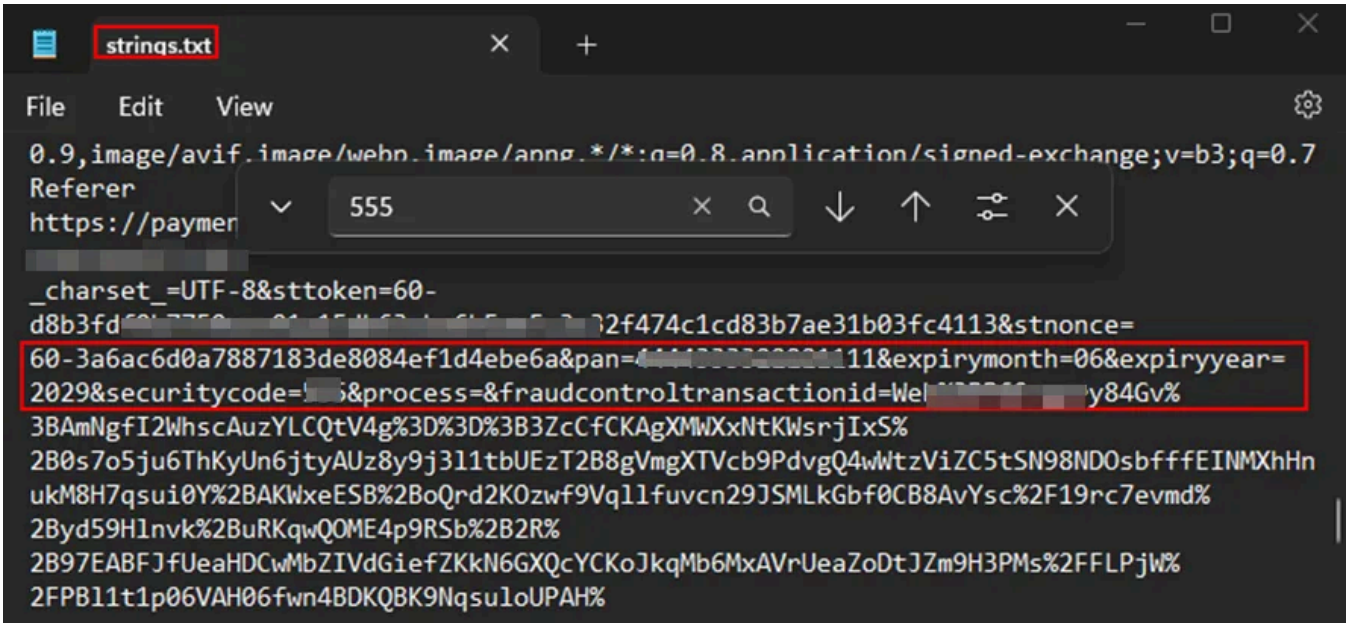


ii) Execute the fridump command as follows:

python3 fridump.py -U -s “App name”



iii) By executing the previous step, the command will create a folder in the current directory called dump and a file named ‘strings.txt’, which contains all the memory strings. Open the ‘strings.txt’ file and it is observed that user-sensitive data are not cleared or replaced from memory.



5: Sensitive Data in the Keychain:

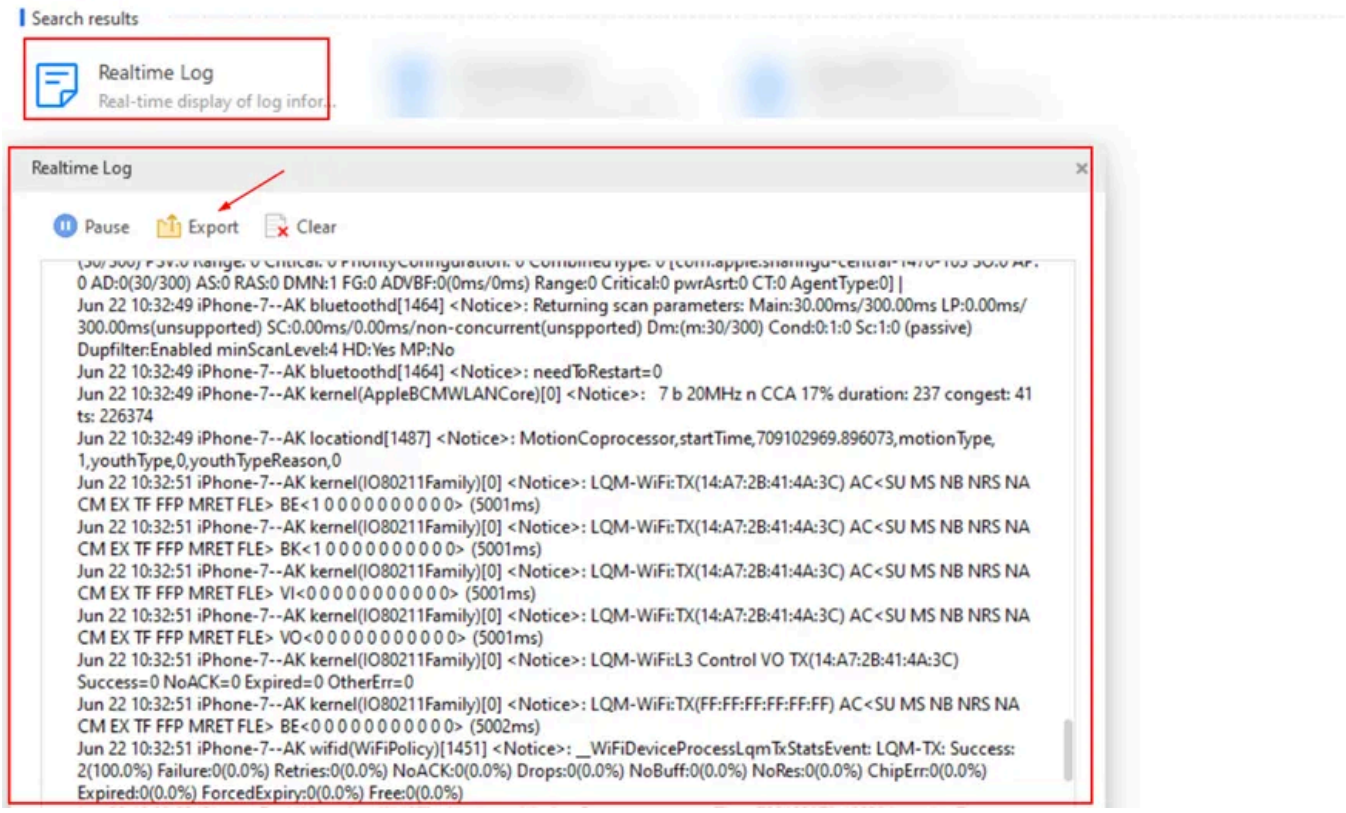
Keychain services provide APIs (Application Programming Interfaces) that allow developers to securely store, retrieve, and manage sensitive data in a way that’s isolated from the rest of the app’s data. And some applications may store this data in plaintext format.

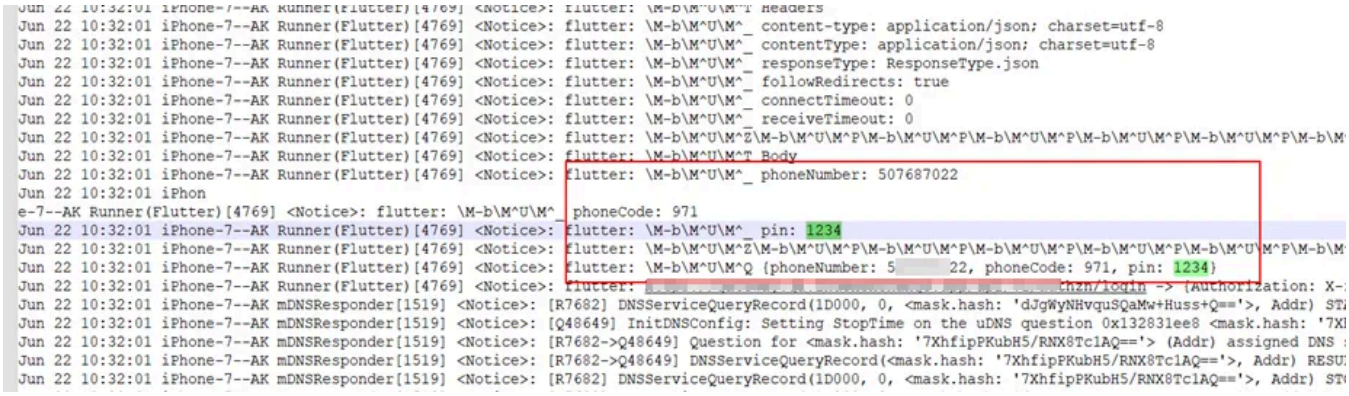
i) After Login process, Attach the target application with the objection and execute the following command:

```
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # ios keychain dump
Note: You may be asked to authenticate using the devices passcode or TouchID
Save the output by adding `--json keychain.json` to this command
Dumping the iOS keychain...
Created                  Accessible    ACL   Type      Account        Service                    Data
-----------------------  ------------  ----  --------  -------------  -------------------------  -------------
2023-08-13 13:32:56 +0000  WhenUnlocked  None  Password  keychainValue  com.highaltitudehacks.dvia  Passw0rd@1234
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] #
```

6) Insecure logging of Sensitive data: There are some dev dudes who logs everything including user data during the app developement process and some of them would forget to remove the log functions during the app releasing process. So in that cases, We gotta grab those snacks ;)

i) Identify the sensitive input fields in the target application.
ii) Submit the data via the input fields
iii) Collect the App logs and obatin the data (You can use “Real Time logger” utility provided by 3uTools

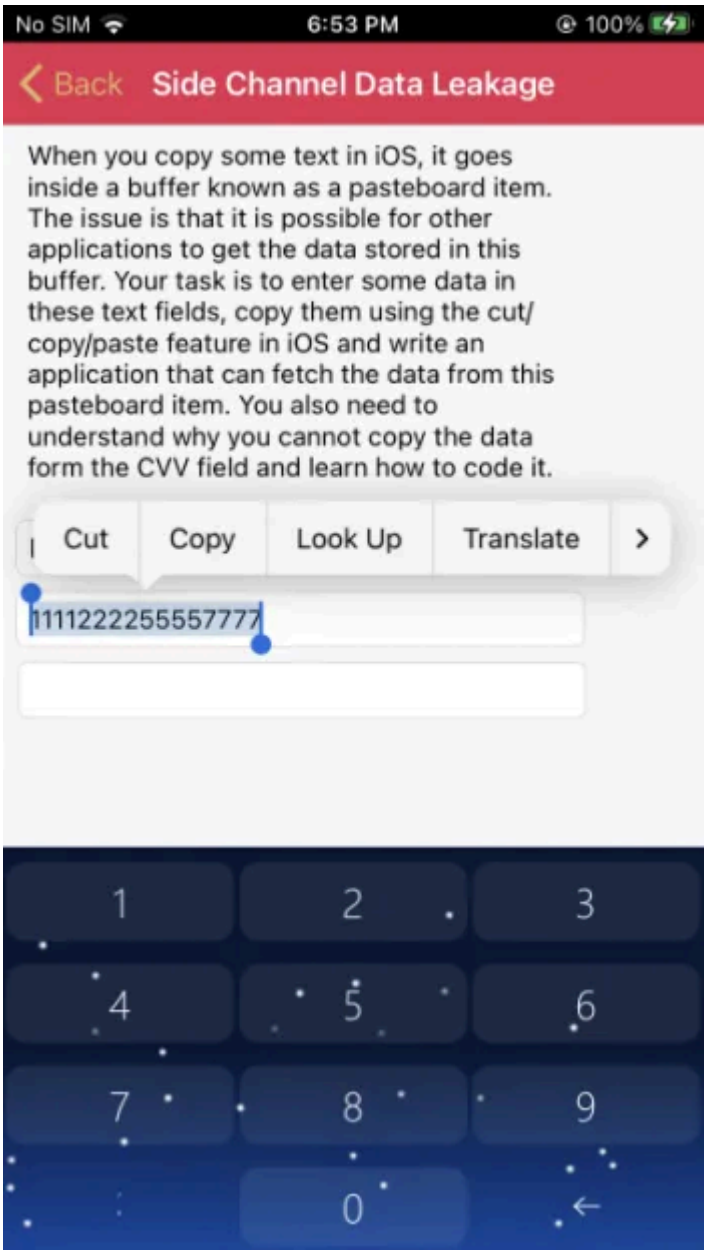




Side Channel Leakage:

1: Clipboard buffer cache

i) Find the sensitive input fields in the target application
ii) Try to copy the sensitive data



iii) Now attach the target application with Objection and execute the command "*ios pasteboard monitor*"

```
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 pasteboard monitor
com.highaltitudehacks.dvia on (iPhone: 16.4.1) [usb] # (agent) [pasteboard-monitor] Data: 1111222255557777
```

We can see the objection could retrieve the copied card data from clipboard cache.

And that's a wrap for today, folks! But hey, don't fret, there's still a bunch of exciting stuff in our toolkit waiting to be unleashed — you know, the whole shebang like setting up proxies, bypassing JailBreak detection and SSL pinning, Bypassing biometrics and more. Buckle up, because in my next installment, I'll dive into all of these juicy topics.

Oh, the exhaustion is real! I'm starting to think the next part might just have to come with its very own YouTube video... I mean, who needs all that typing, am I right?

Bella Ciao..

IOS Pentesting Cybersecurity Mobile Security Hacking

Following

Written by Kishor balan

418 Followers

Security Analyst || eWPTXv2 || eJPT

More from Kishor balan

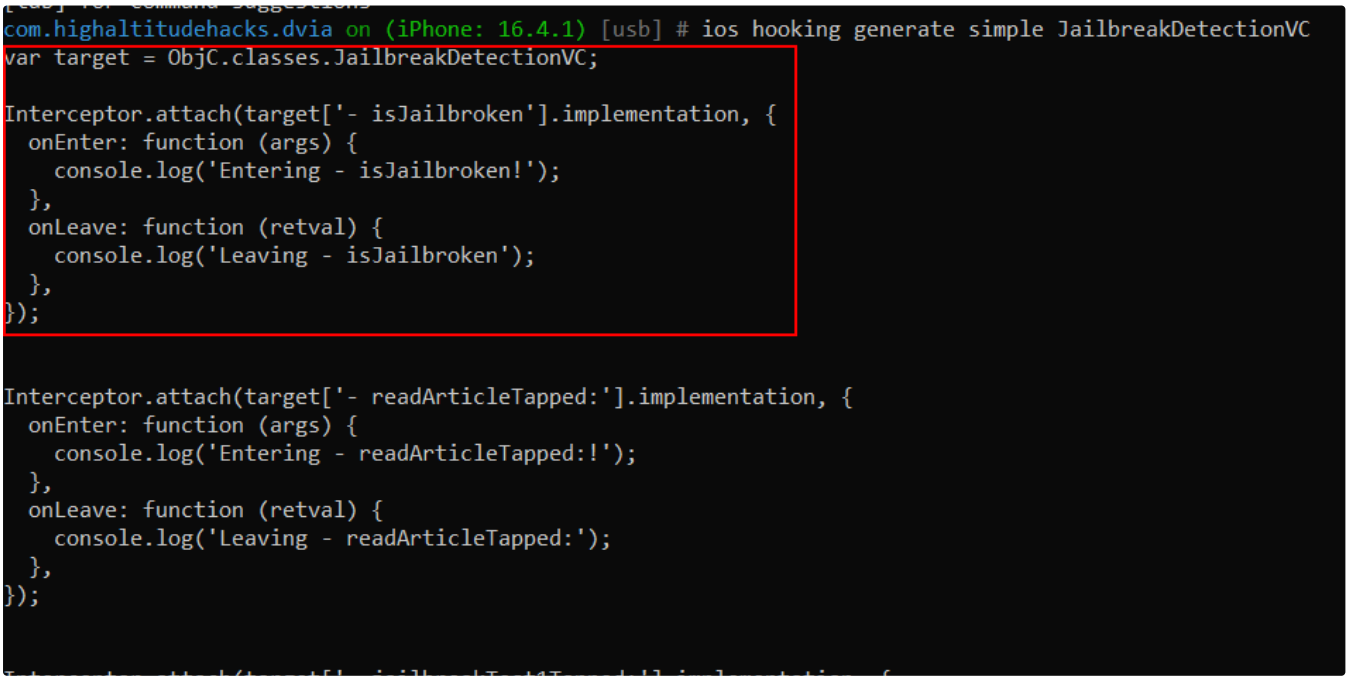
Kishor balan

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

Hola H3ckers,

6 min read · Nov 27, 2022

243 2



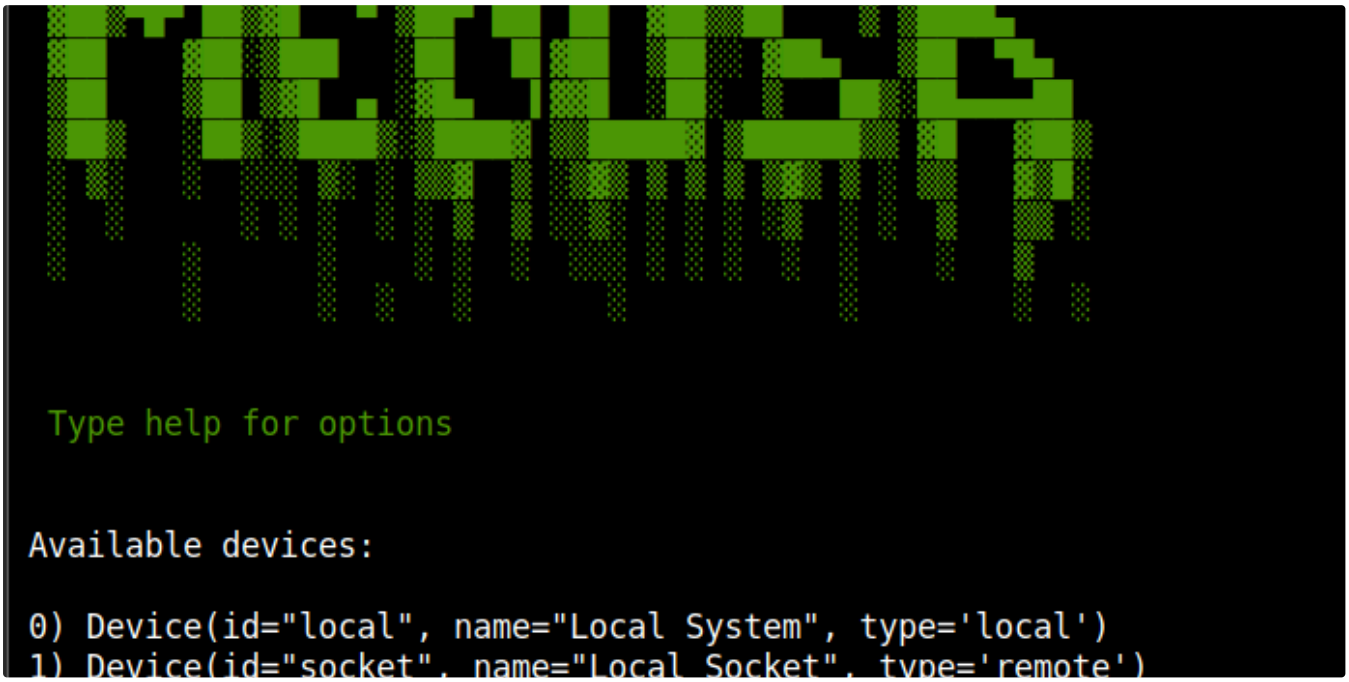
Kishor balan

iOS Pentesting Series Part 3- The Ceasefire

Hola mates,

7 min read · Aug 19, 2023

22



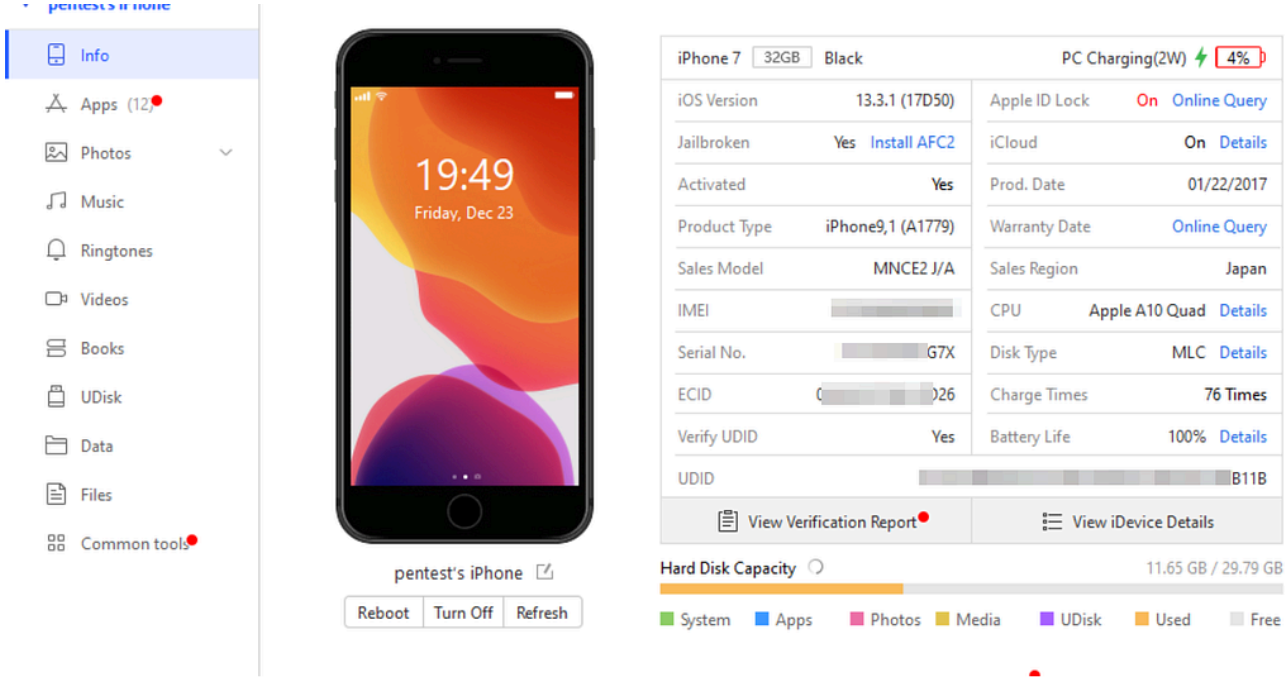
Kishor balan

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

See all from Kishor balan

Recommended from Medium

D 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

Chinmay Talad

Pass eJPT with Tryhackme Challenge Room

Below is the list of free Tryhackme rooms which will help you to pass the exam, The lists contains both walkthroughs and CTF challenges, I...

2 min read · Nov 22, 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

Sandeep Vishwakarma in InfoSec Write-ups

A step-by-step Android penetration testing guide for beginners

Greetings fellow hackers, my name is Sandy, Security Analyst and Bug bounty hunter.

18 min read · Nov 3, 2023

678 4

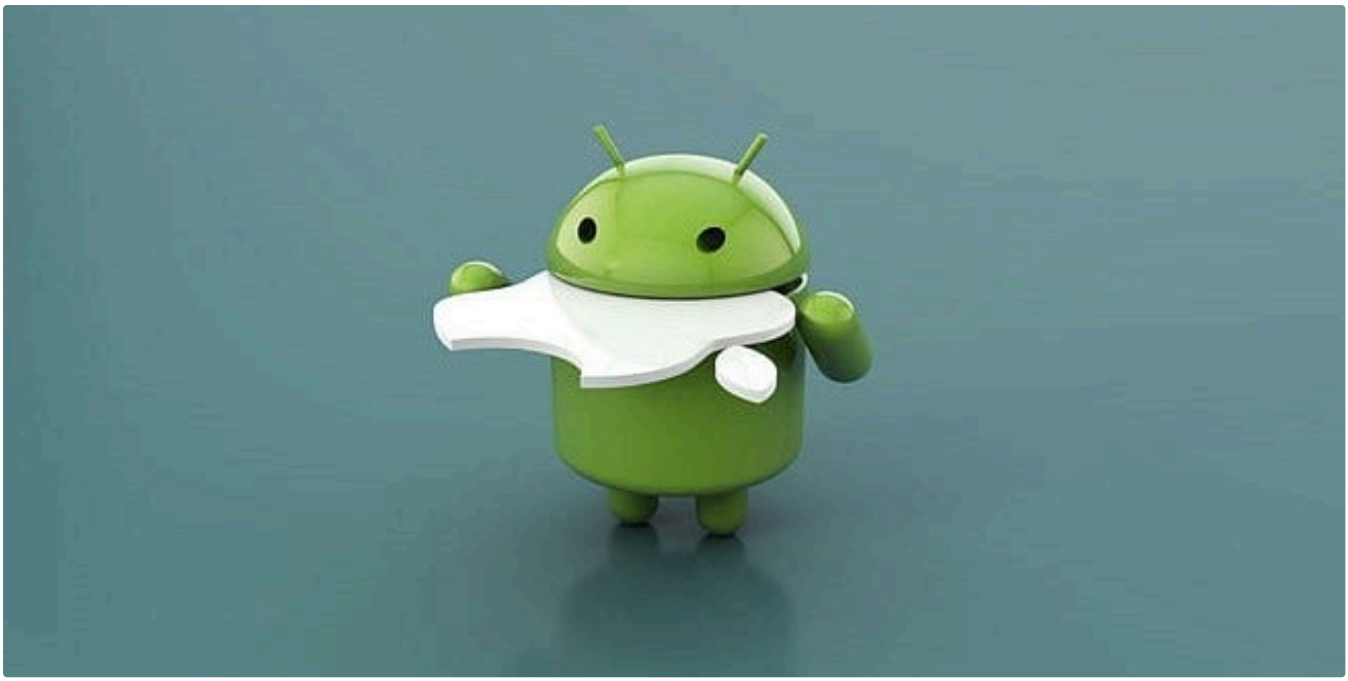
Pawan Jaiswal

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



Prince Varghese

Task Hijacking in Android

Hello friends, Welcome back to my blog. Today we are going to explore Task Hijacking vulnerability in Android.

7 min read · Nov 25, 2023

44

Hacker's Dump

Diving Deep: A Comprehensive Guide to Android Penetration Testing — Part 1

Introduction: Navigating the Android Abyss

7 min read · Nov 16, 2023

 28  

See more recommendations