My fav 7 methods for Bypassing Android Root detection

Kishor balan · Following

6 min read · Oct 16, 2022

Hola H3ck3rs,

You peeps already know, when we are into an android pentest/BB, the root detection protections are basically a creepy thing for all of us.

I am not going to explain a new methodology or any of my own discovery rather than listing out a few of the existing traditional and modern methods.

Prerequisites:

Familiar with Basic Android Pentesting and tools such as adb, frida, Objection, Magisk application, Decompiling/Recompiling APK, APK Signing and Dex to Jar Conversion.

What we are gonna cover today:

1. MagiskHide module in the Magisk application (= <v23.0)
2. Zygisk DenyList in the Magisk Application (>v23.0)
3. Tampering the Smali code

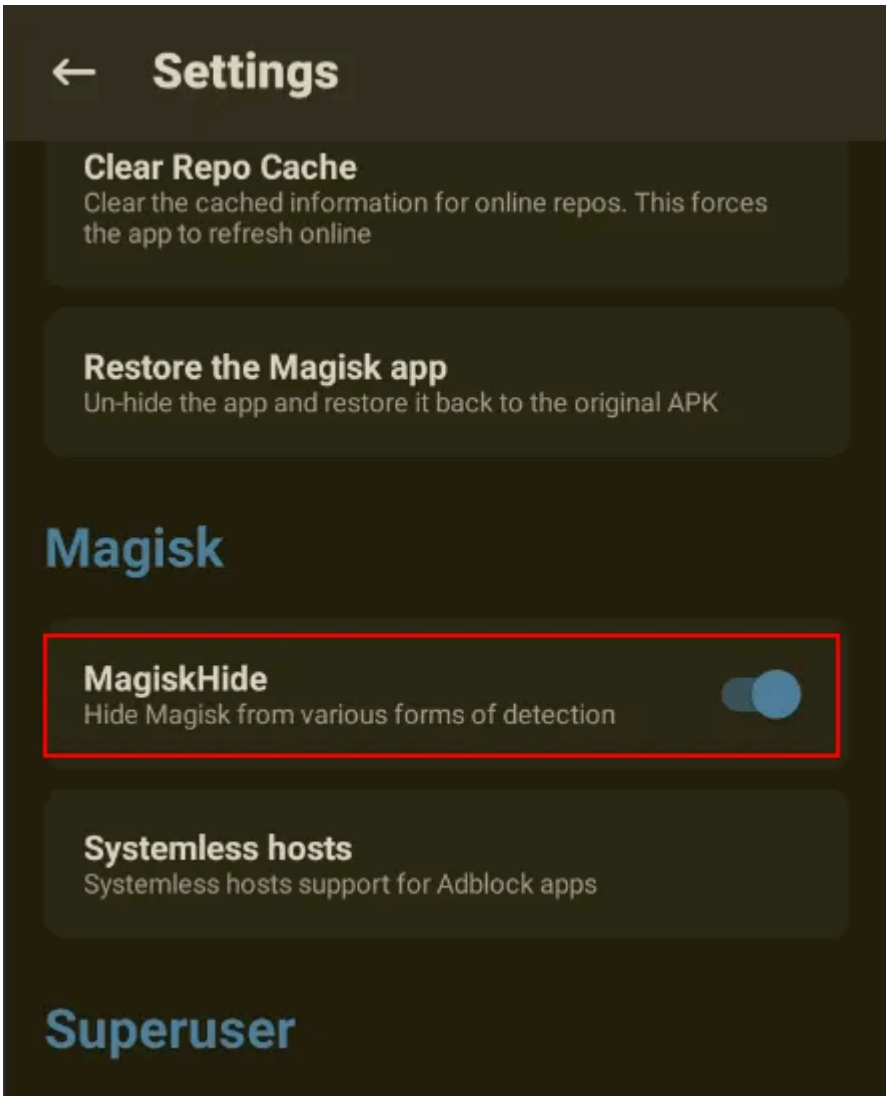
4. Objection's common method
5. Bit manual with Objection
6. Using Frida scripts
7. Medusa framework

!!! I' haven't covered the basic stuff like setting up the frida server, decompiling/Re-compiling the android application, etc. Hope u guys are already familiar with them!!!

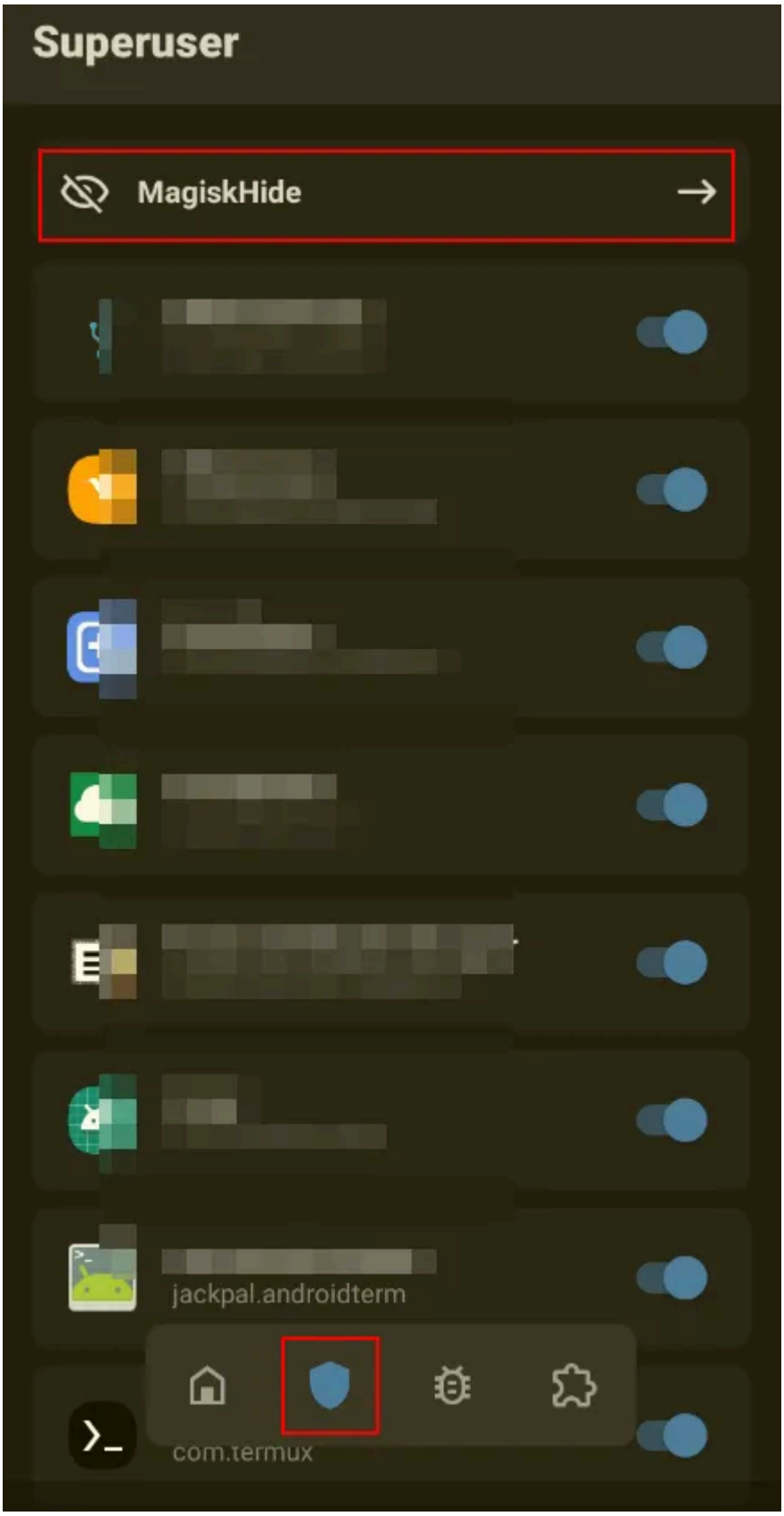
1) MagiskHide module in the Magisk application (v23.x< =)

If your device is rooted with Magisk, then you can try the MagiskHide available in the Magisk Application. **And please note that this feature is only available up to the Magisk version 23.x**

Step:1 *Enable the MagiskHide module from the app settings*



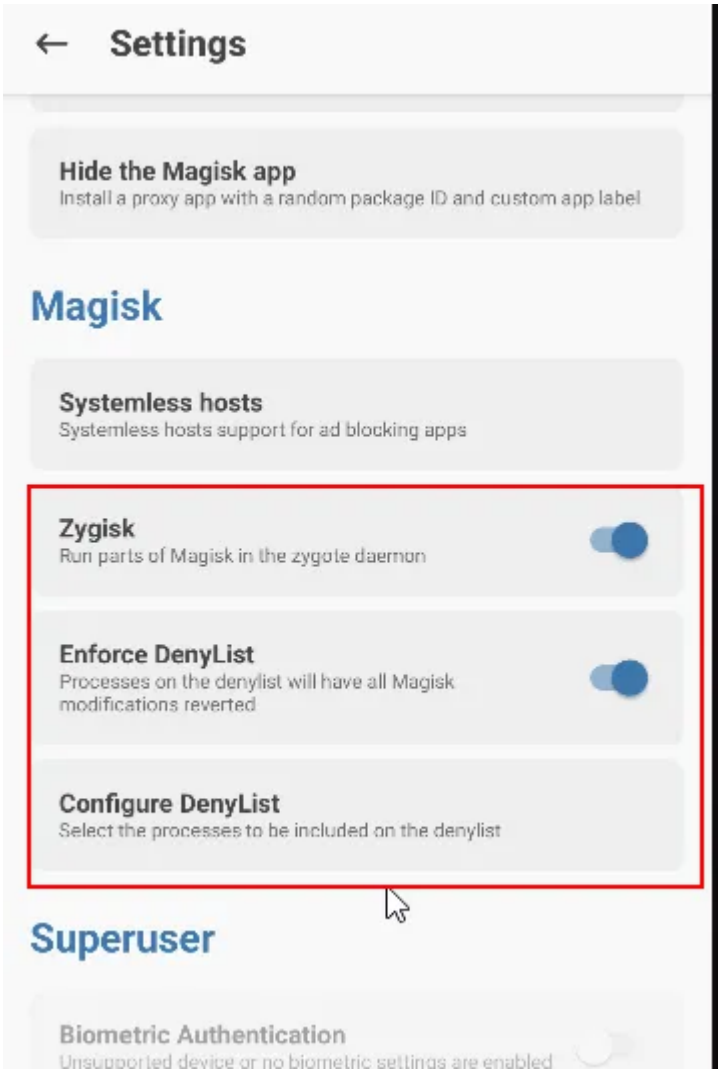
Step 2: *Choose the application in which we have to hide the root*


Superuser
MagiskHide
jackpal.androidterm
com.termux

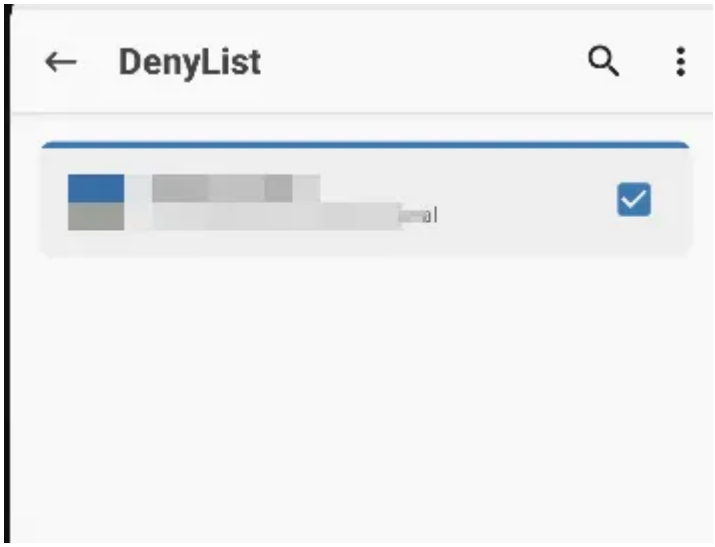
2. Zygisk Denylist (Magisk App > v23.x)

Magisk App > v23.0 provides an alternative feature -Zygisk Denylist instead of the MagiskHide. By enabling this feature, we can bypass the root detection of most applications.

Step 1*: Enable Zygisk, DenyList from the app settings*



Step 2: Choose the application that we have to hide the root detection

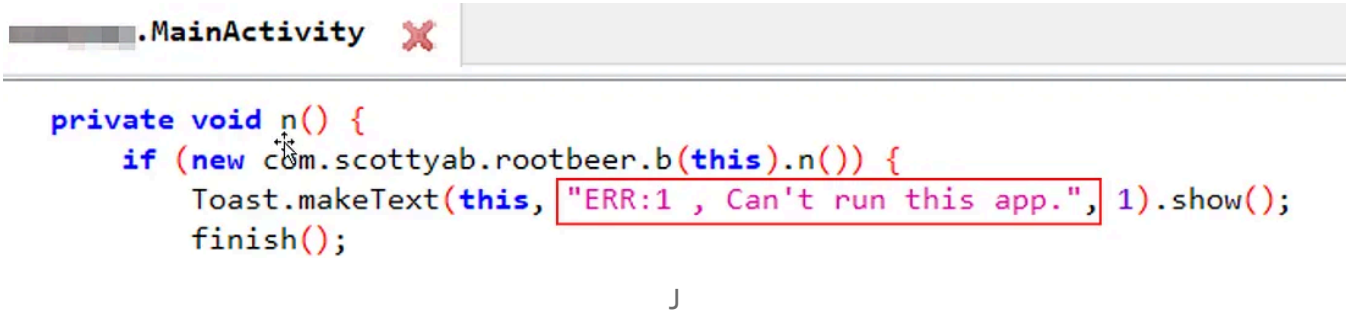


3. Tampering the Smali Code

Yes, sometimes we can do a spell with the application smali code which can be obtained after decompiling the application with **Apktool.**

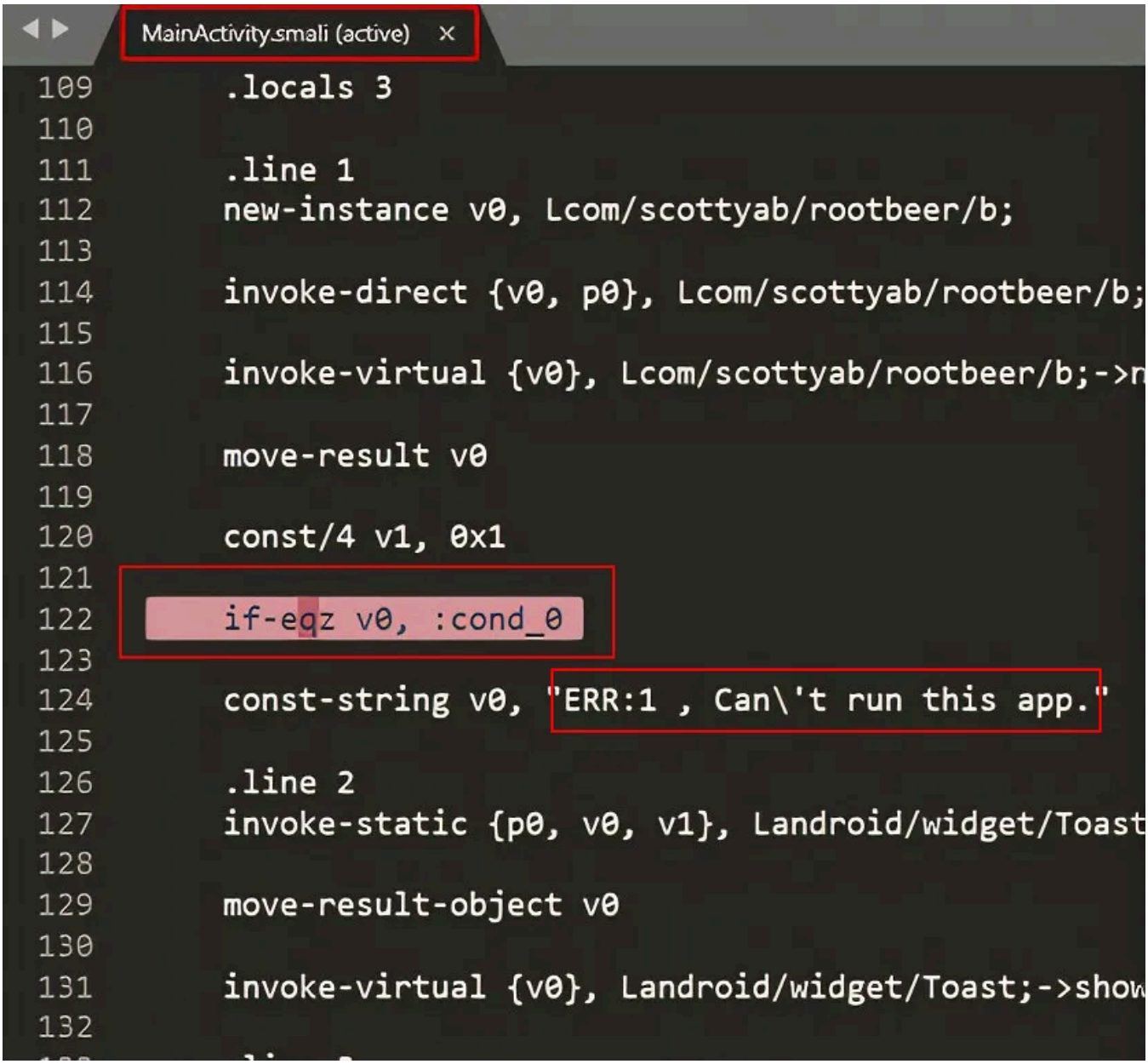
Step 1*: Decompile the apk file using* ***JADX-GUI*** *or any other alternative.*

Step 2: *Identify the code which is in charge of the root detection process:*

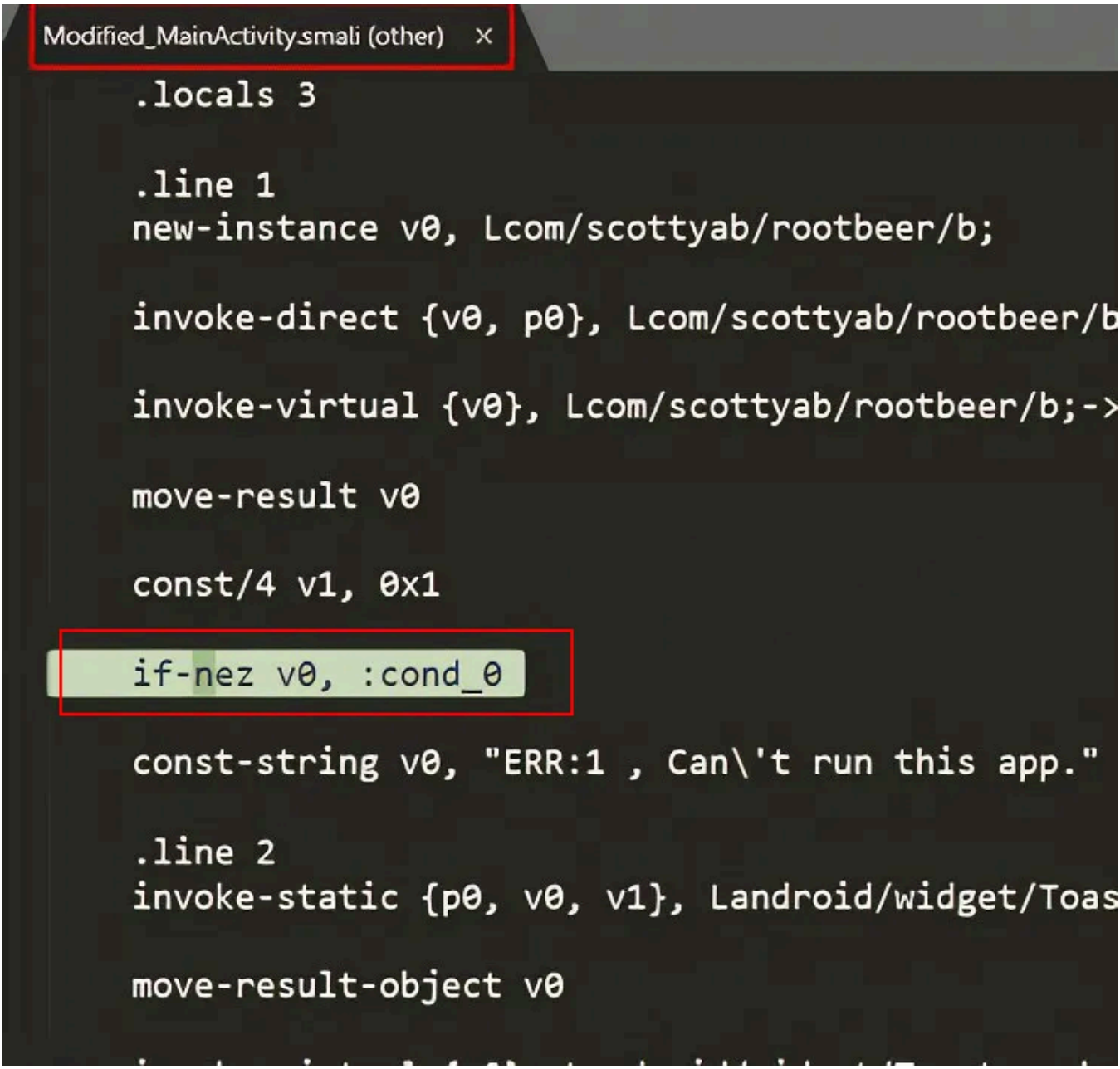


Step 3: *In this case, the application is using the* ***rootbeer*** *library for the root detection. And we can see there is a "****if****" condition is the decision maker element that decides whether the application is rooted or a Non-rooted device.*

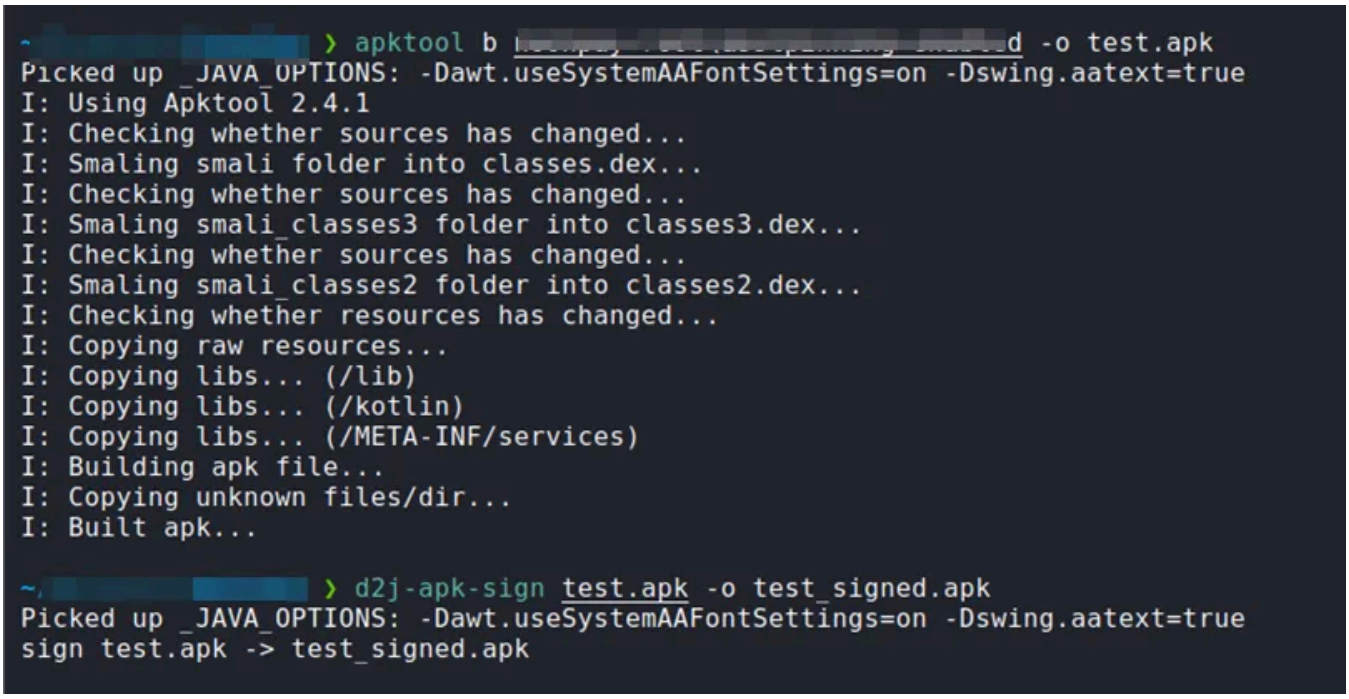
Step 4: *Now decompile the APK with Apktool and and find smali code for the above "****if*** *"statement.*



Step 2: *Modify the condition statement as illustrated in the below picture.*



Step 3: *Now save the file and re-build the application, don't forget to sign the apk file.*



Re-packing the application

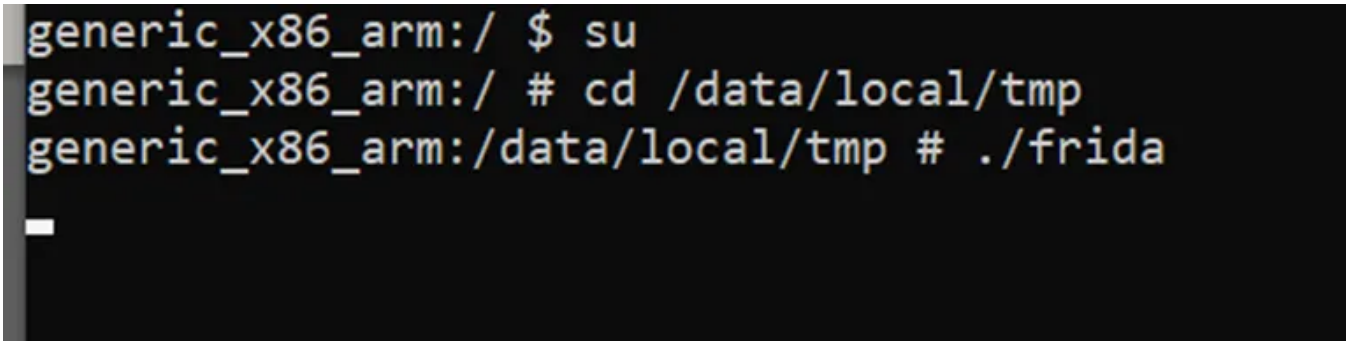
This will change the application logic and bypass the root detection.

4. Using Objection's Common method

I hope u guys are familiar with the objection tool, if not, pull it out using pip

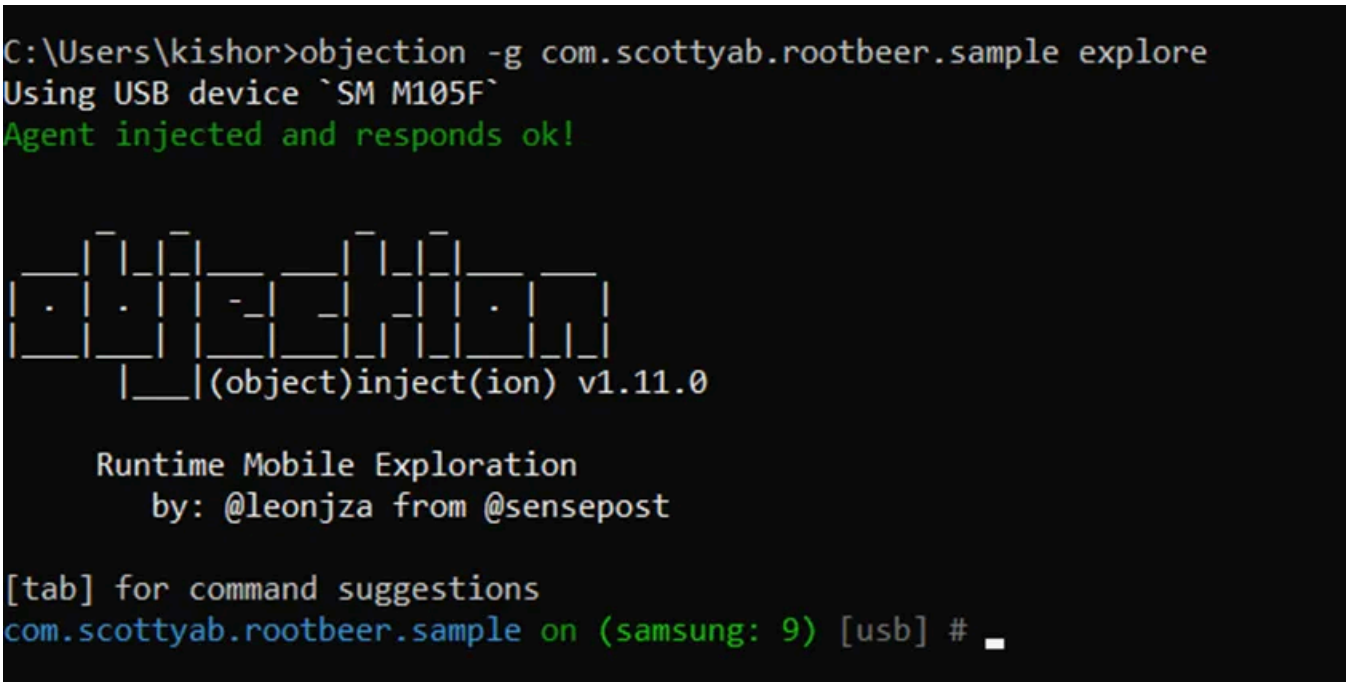
pip3 install objection

Step 1*: start the frida server on the android device/ Virtual device.*

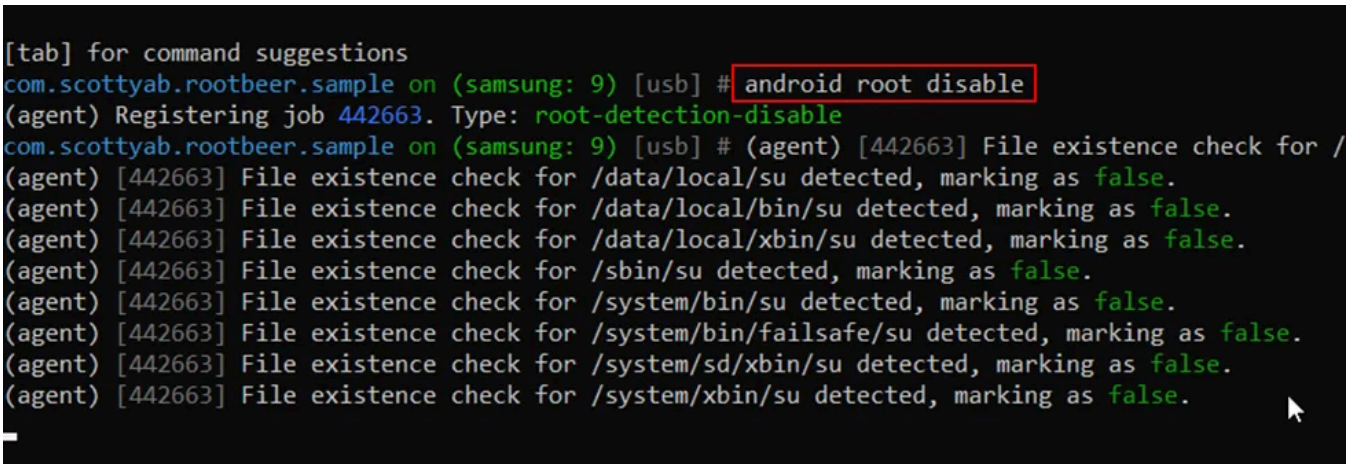


Step 2: *Now, launch the application with the following Objection command.*

objection -g com.test.app explore



Step 3: *now the application will be launched on your device, then execute the following objection command.*



This module will modify the values of the class methods of the root detection library in order to bypass the root detection

Step 4*: If the root detection prompt is still there on the application, just press the back button and exit from the application (don't kill the app from background) and open it again.*

The root detection will disappear :)

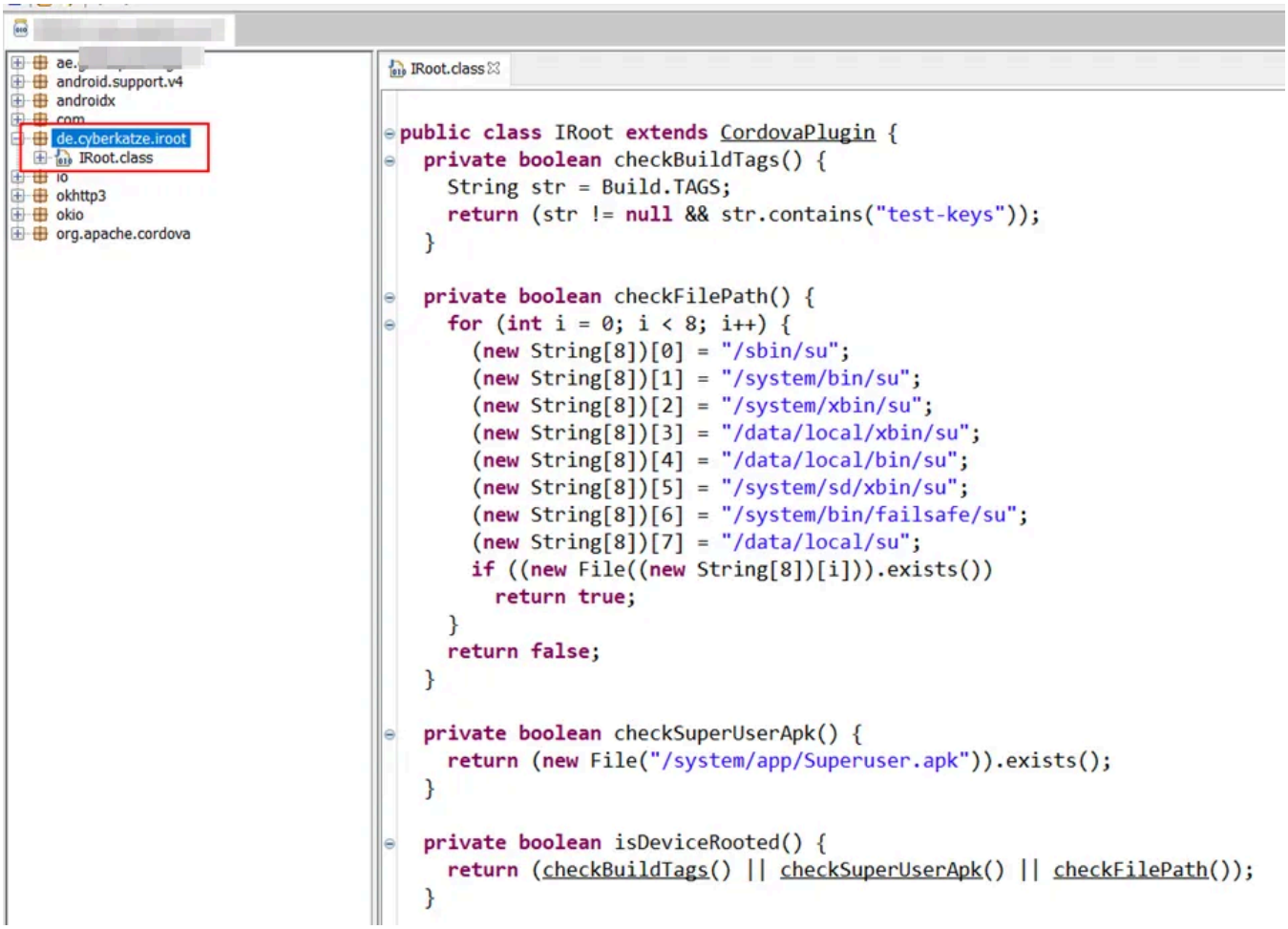
5. Bit Manual with Objection tool

This method is a kinda tricky.

Following are my steps.

Step 1: *First convert the apk file into class files using* ***dex2jar***

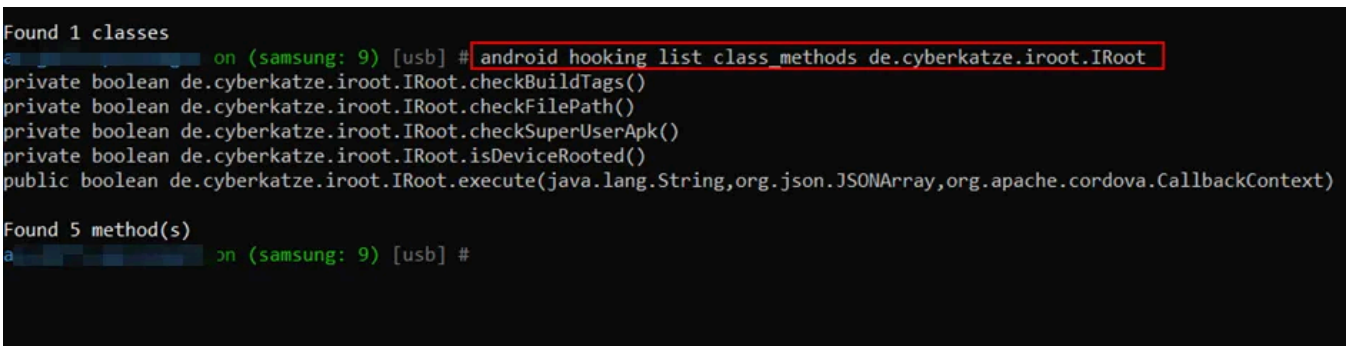
Step 2: *Analyse the class files and identify which library is being used for the root detection*



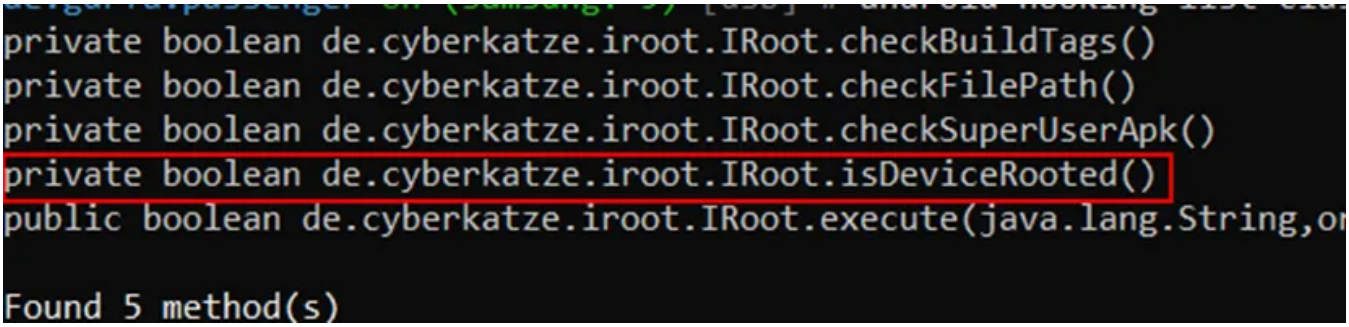
Step 3: *Now connect the app with objection (objection -g pkg_name/processID explore)*

Step 4: *Execute the following command*

android hooking list class_methods <root detection class>

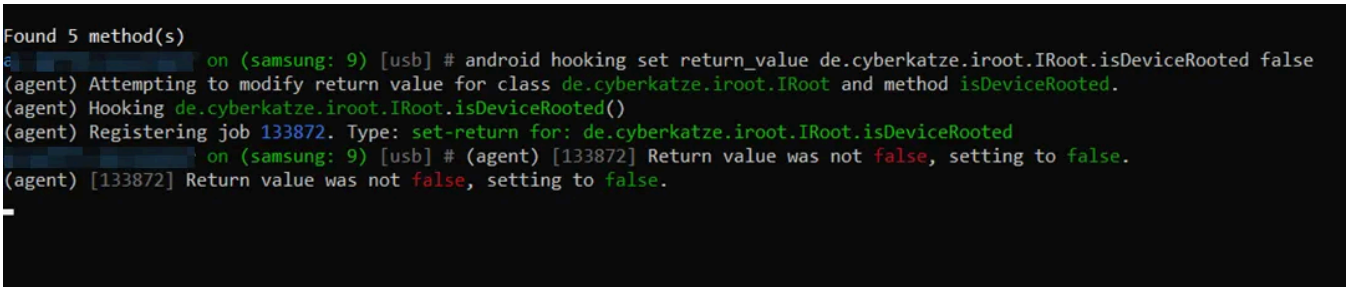


Step 5: *Now we can identify that the following boolean method* ***isDeviceRooted()*** *(may differ in other libraries) is in charge of root detection.*



Step 5: *We can change its return value* ***true*** *to* ***false*** *with the following command:*

android hooking set return_value <root_detect_class.method> false



Step 6: *Thats it, here we have changed the boolean value returned by the root detection class method, and this will change the application logic for root detection process and bypass the root detection.*

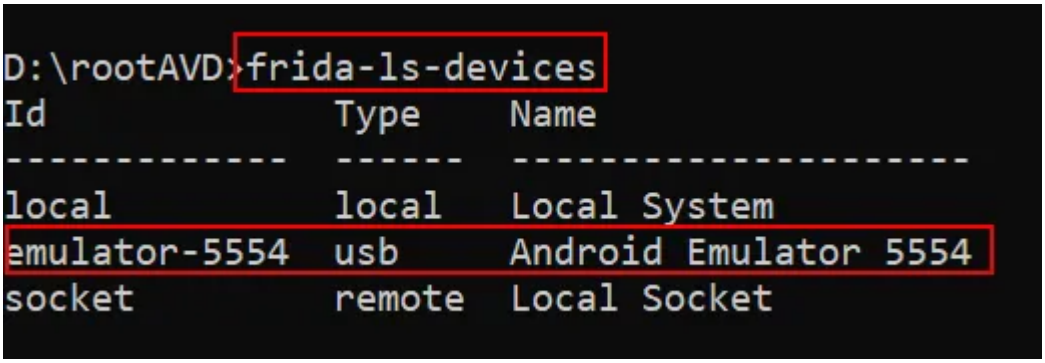
6. Using Frida scripts

Here comes the most and widely used root detection bypass method.

Step 1: *Make sure the frida server is running on your device and the USB cable is properly connected (if it is a physical device)*

Step 2: *Also make sure that your PC and the device have established a proper frida connection.*

Step 3: *You can verify the frida connection by using the command, shown in the below image.*

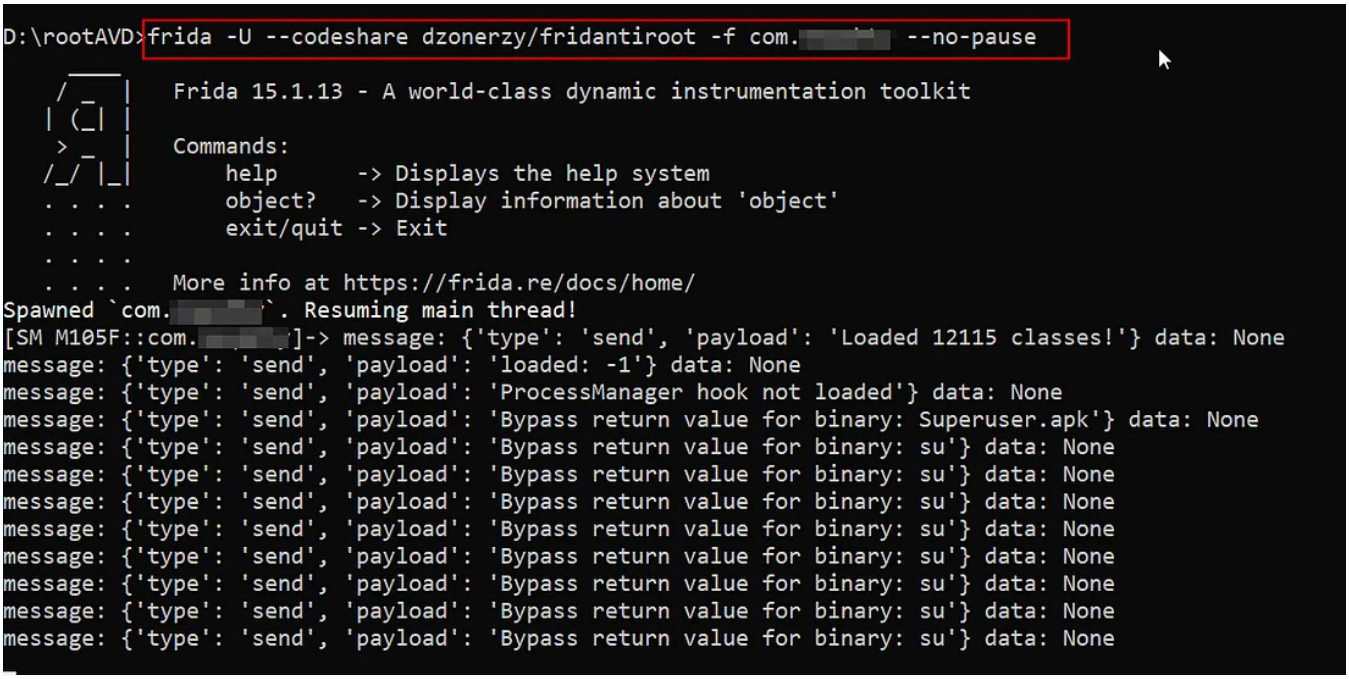


Step 4: *Now we have to find a good and popular frida scripts from the Codeshare or any other git repositories*

My fav one is:

i) https://codeshare.frida.re/@dzonerzy/fridantiroot/

Step 5: *Now connect your application with frida and load the frida script with the command, shown in below image.*



7. Medusa Framework

Similar to Objection tool, MEDUSA is also a flexible framework that automates the procedures and methods used in the dynamic analysis of Android applications.

Git repo:

Ch0pin/medusa: Binary instrumentation framework based on FRIDA (github.com)

Step 1: *Launch medusa*

Step 2: *We have to use the* ***helpers/anti_debug*** *module for bypassing the root detection*

Step 3: *Follow the steps illustrated in the following screenshots.*

```
medusa> search anti_debug
helpers/anti_debug
medusa> use helpers/anti_debug

Current Mods:
0) helpers/anti_debug


medusa> compile

Script is compiled

medusa> run -f ae.[redacted]er
Spawned package : [redacted] on pid 26917
[in-session] |c:clear |e:exit |r:reload |
| rs:reset scratchpad |i:info |t:trace  |?:help |:
-------------------Application Info--------------------

- Application Name: android.app.Application
- Files Directory: /data/user/0/[redacted]r/files
- Cache Directory: /data/user/0/[redacted]r/cache
- External Cache Directory: /storage/emulated/0/Android/data/[redacted]r/cache
- Code Cache Directory: /data/user/0/[redacted]/code_cache
- Obb Directory: /storage/emulated/0/Android/obb/[redacted]r
- Package Code Path: /data/app/[redacted]r-yLTMCanVbK7g-bdT-YkiAw==/base.apk
```

```
-------------------EOF Application Info-----------------


---------LOADING ANTI ROOT DETECTION SCRIPT-------------------
Loaded 11616 classes!
loaded: -1
ProcessManager hook not loaded
[!] overwriting is debugger connected
Bypass test-keys check
Bypass return value for binary: Superuser.apk
Bypass return value for binary: su
Bypass return value for binary: su
Bypass return value for binary: su
Bypass return value for binary: su
Bypass return value for binary: su
Bypass return value for binary: su
Bypass return value for binary: su
Bypass return value for binary: su
Bypass test-keys check
Bypass return value for binary: Superuser.apk
Bypass return value for binary: su
Bypass return value for binary: su
Bypass return value for binary: su
Bypass return value for binary: su
Bypass return value for binary: su
Bypass return value for binary: su
Bypass return value for binary: su
Bypass return value for binary: su
```

Bypassing root detection during the run time

That's it peeps, I'm sure there is a 90% possibility that any of these methods will help you to get rid of the root detection problem.

See you on next write up

References:

Ch0pin/medusa: Binary instrumentation framework based on FRIDA (github.com)

topjohnwu/Magisk: The Magic Mask for Android (github.com)

frida/frida: Clone this repo to build Frida (github.com)

Cybersecurity Android Infosec Bug Bounty Pentesting

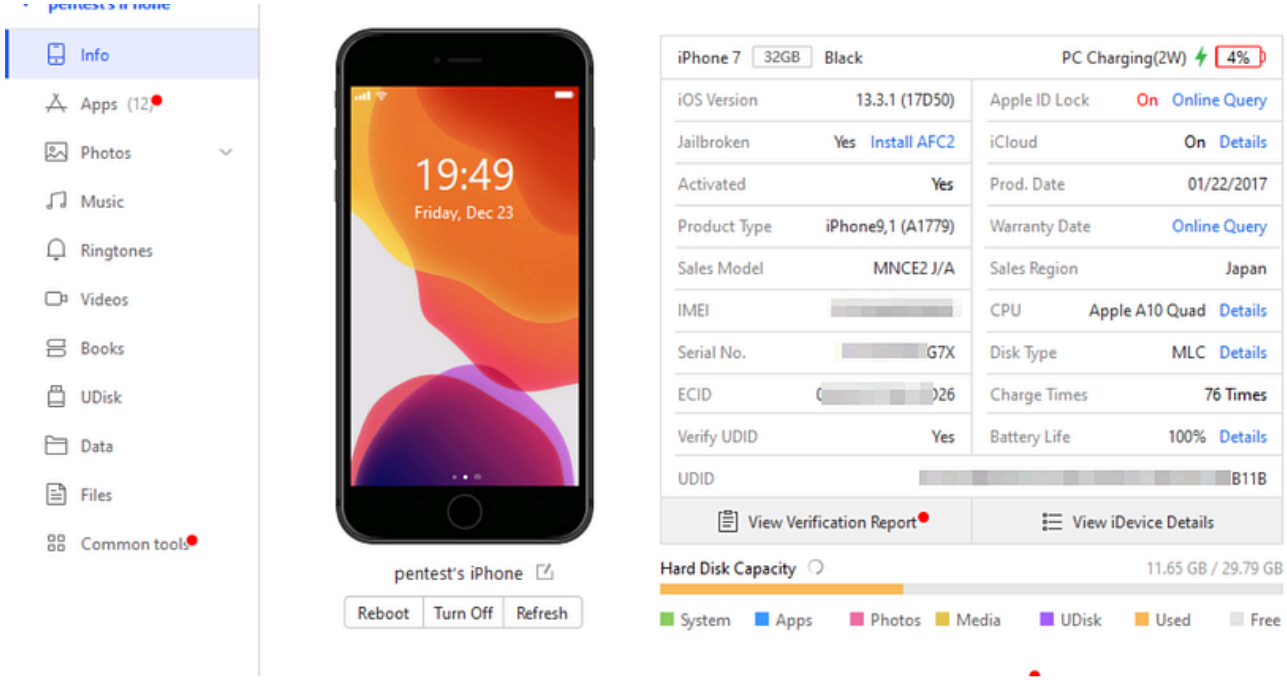
Following

Written by Kishor balan

418 Followers

Security Analyst || eWPTXv2 || eJPT

More from Kishor balan



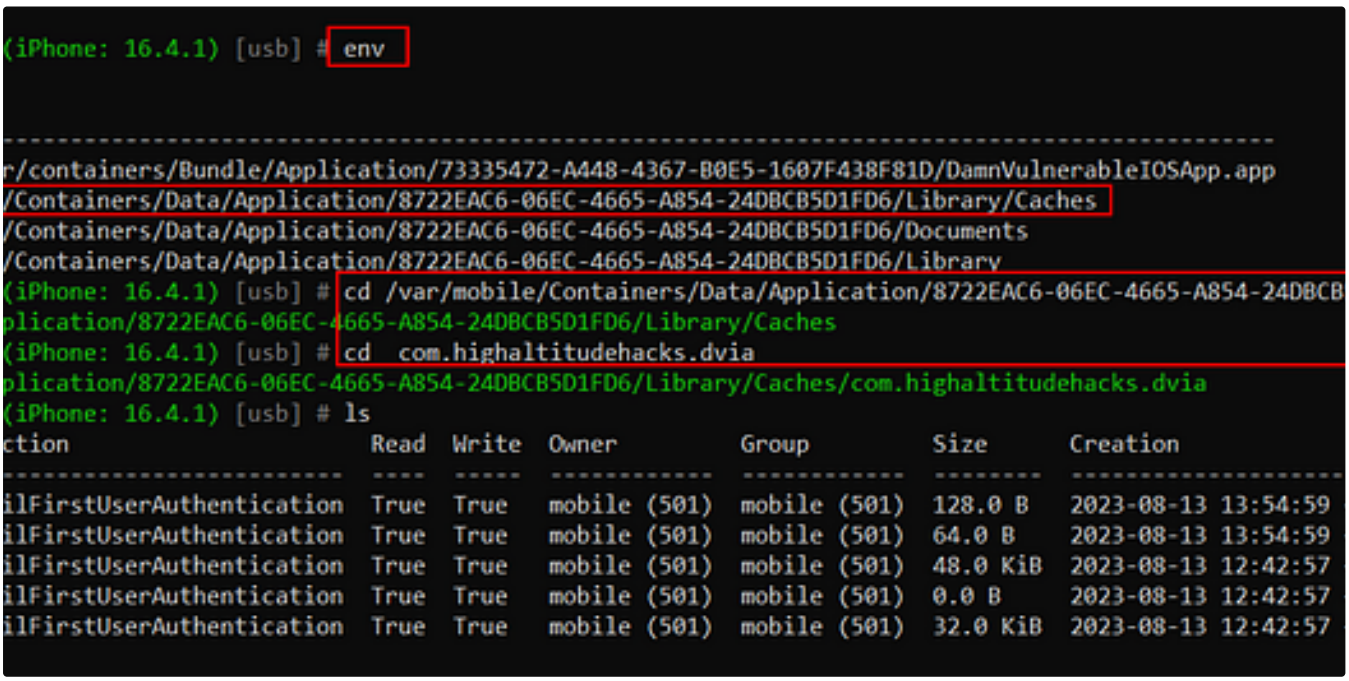
Kishor balan

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

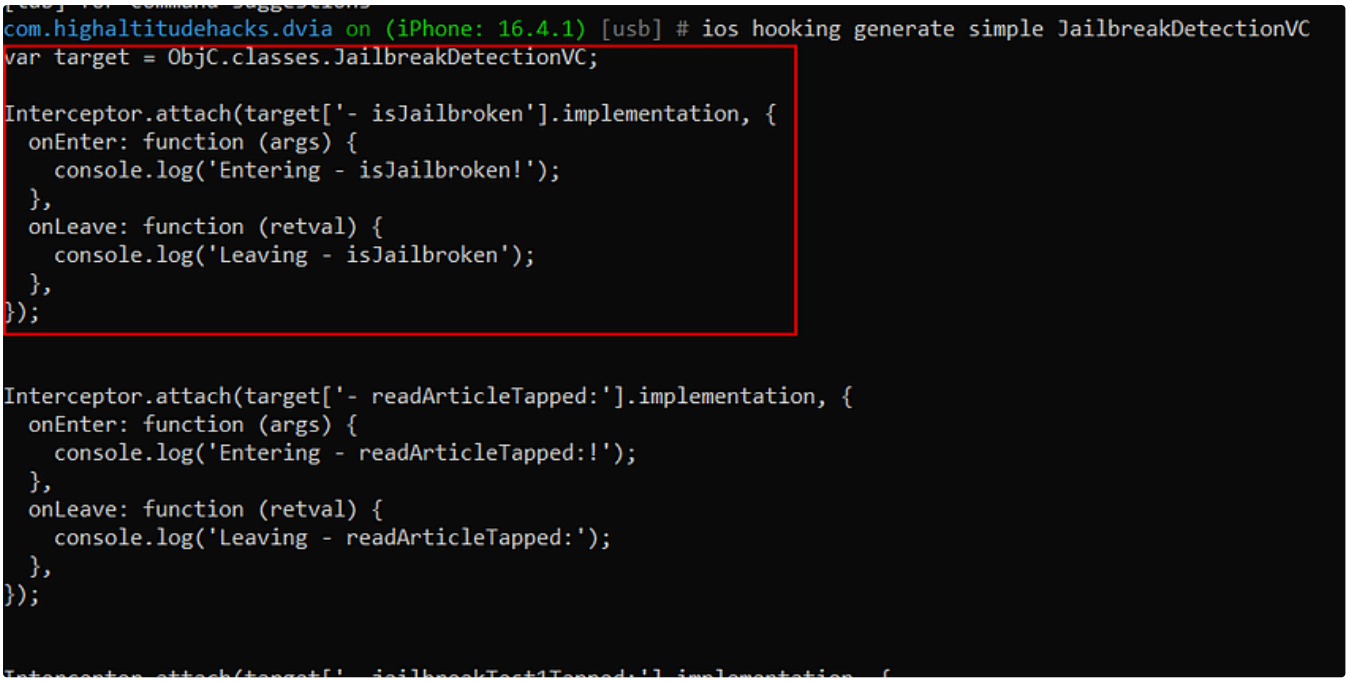
Kishor balan

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

Hola H3ckers,

6 min read · Nov 27, 2022

243 2



```
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:');
  },
});
```

Kishor balan

iOS Pentesting Series Part 3- The Ceasefire

Hola mates,

7 min read · Aug 19, 2023

22

See all from Kishor balan

Recommended from Medium

Aimar Sechan Adhitya

Bypassing Root Detection the Universal Way

Getting tired of reverse engineering obfuscated codes to bypass root detection? Time to learn the better way!

4 min read · Jan 12, 2024

41 1

Pawan Jaiswal

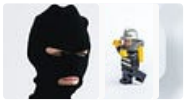
Android Penetration Testing with Frida: A Comprehensive Guide with Examples

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

Lists



Tech & Tools

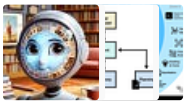
16 stories · 212 saves

Medium's Huge List of Publications Accepting Submissions

285 stories · 2528 saves

Staff Picks

630 stories · 922 saves



Natural Language Processing

1407 stories · 905 saves

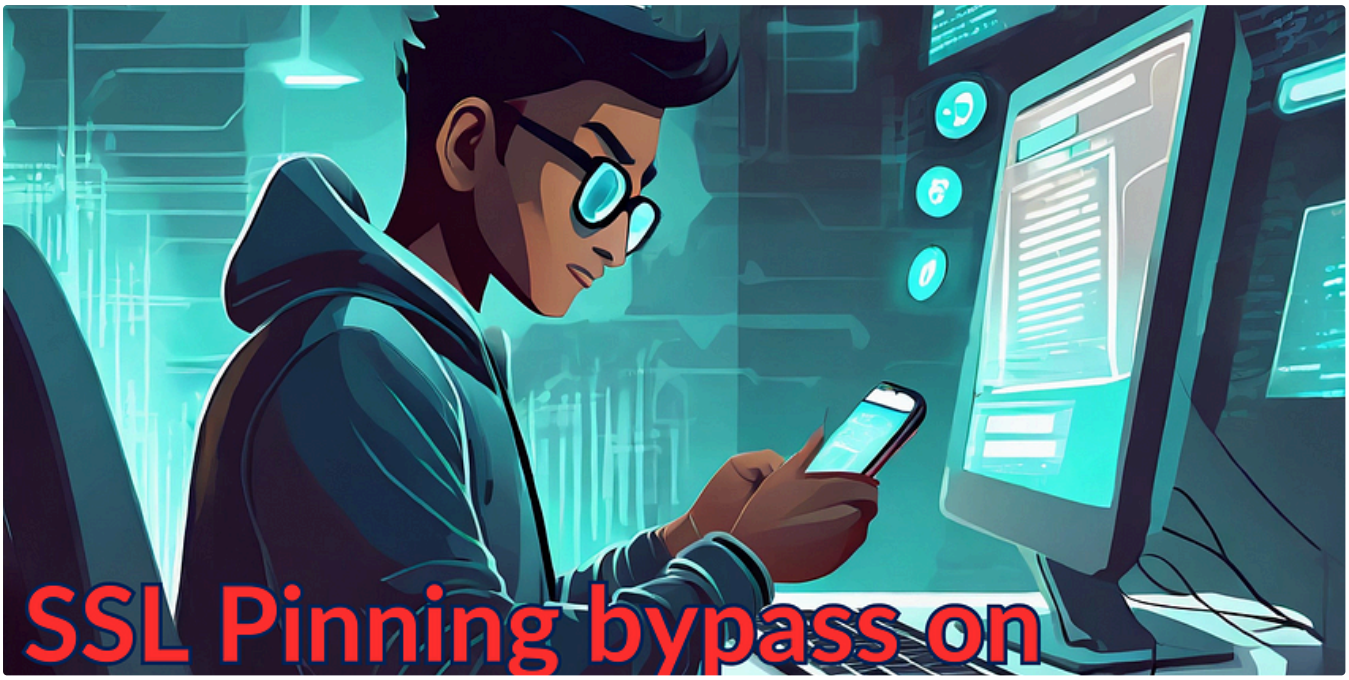
Sandeep Vishwakarma in InfoSec Write-ups

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



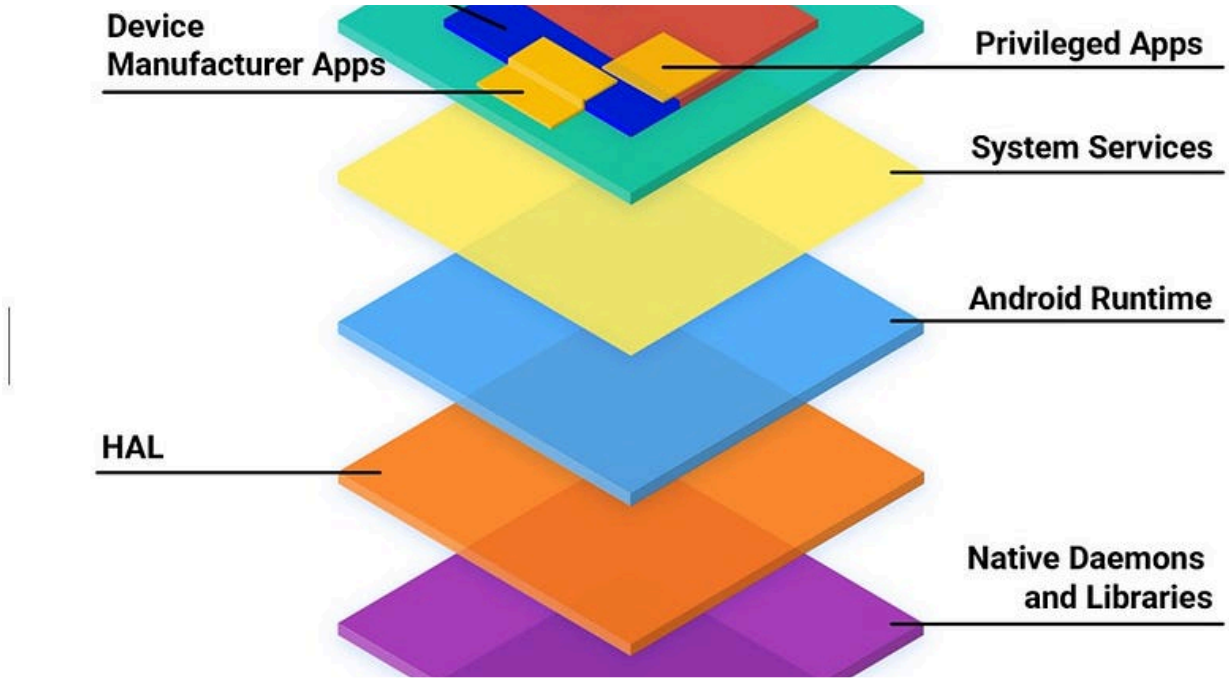
Sahil Choudhary

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



Rudranil Maity

Network Traffic Analysis of Android apps using MITMproxy and Frida

Abstract: With the overwhelmingly increasing number of applications in android marketplace, i.e., google play store; over time with the...

15 min read · Nov 22, 2023

Hacker's Dump

Diving Deep: A Comprehensive Guide to Android Penetration Testing — Part 3

Cracking the Code: A Beginner's Guide to Decoding Android Apps

8 min read · Nov 19, 2023

 2