

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



Kishor balan · Following

6 min read · Nov 27, 2022

Listen

Share

More

Hola H3ckers,

We all know there are a plenty of articles available for “How to intercept the HTTPS traffic of Android apps”, So here we are not going to cover them. If you have not found any, Refer the following:

[Configuring an Android Device to Work With Burp – PortSwigger](#)

Prerequisites:

Familiar with BurpSuite proxy, Basic Android Pentesting and tools such as adb, frida, Objection, Magisk application, Decompiling/Recompiling APK, and APK signing.

Table of Contents

1. Does my target app have SSL pinning?
2. Wait, How we can confirm the Pinning?
3. Time to Bypass
4. Why I am not able to intercept the app traffic even if the app is Working with HTTP

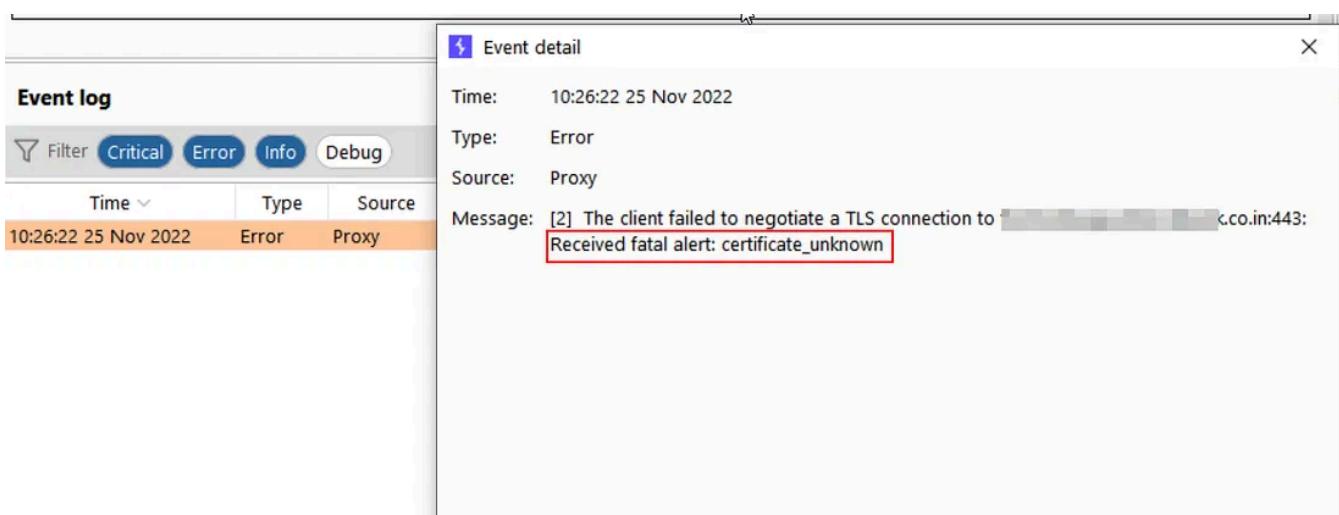
1. Does my target app has SSL pinning??

I got it, that sounds like a joke, because you guys know If the pinning is in place, then we won't be able to capture the HTTPS traffic of our target android application.

2. Wait, How we can confirm the Pinning?

After setting up the proxy in both the device and the proxy server (Burp), Fire up the target application, then do some activities that makes a communication between the target application and their server.

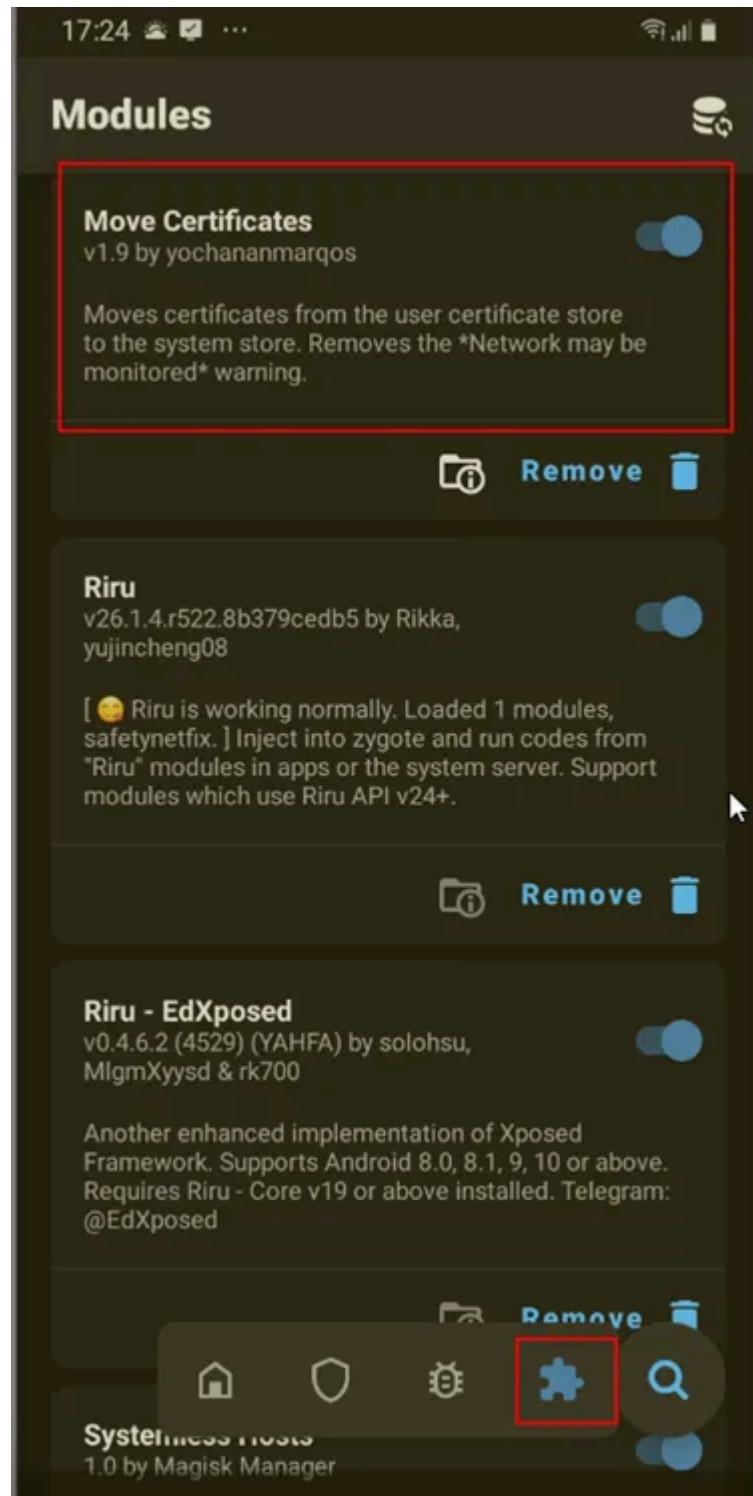
Time to monitor the Burp's dashboard, in specific, the Log section. If the Pinning is in place, then we will be able to see a Certificate error as follows:



2. Time to Bypass

2.1 Move Certificate — Magisk Module:

If your device is rooted with Magisk Application, Then Move Certificate module is one of good option.



This module will move the user trusted certificates to the system store , making the system (root) trust the Certificate which the user install (Our proxy CA certificate)

2.2 Objection tool

Repo: [sensepost/objection](#): 📱 objection – runtime mobile exploration (github.com)

Step 1: Make sure the frida server is running on the android device

```
generic_x86_arm:/ $ su
generic_x86_arm:/ # cd /data/local/tmp
generic_x86_arm:/data/local/tmp # ./frida
```

Step 2: Attach the target application with objection with the following command:

Objection -g <pkg name/ PID> explore

Then execute the “*android sslpinning disable*” command

```
C:\Users\kishor>frida-ps -Uai | findstr 13701
13701 [REDACTED]_demo [REDACTED].myapplication

C:\Users\kishor>objection -g 13701 explore
Using USB device `SM M105F`
Agent injected and responds ok!

[REDACTED]
[REDACTED] (object)inject(ion) v1.11.0

Runtime Mobile Exploration
by: @leonjza from @sensepost

[tab] for command suggestions
[REDACTED].myapplication on (samsung: 9) [usb] # android sslpinning disable
(agent) Custom TrustManager ready, overriding SSLContext.init()
(agent) Found com.android.org.conscrypt.TrustManagerImpl, overriding TrustManagerImpl.verifyChain()
(agent) Found com.android.org.conscrypt.TrustManagerImpl, overriding TrustManagerImpl.checkTrustedRecursive()
(agent) Registering job 693255. Type: android-sslpinning-disable
[REDACTED].myapplication on (samsung: 9) [usb] #
```

Thats it , the script will find the SSL pinning classes and hook them during the runtime in order to byass the Pinning.

2.3 Frida Framework

Repo: [Frida \(github.com\)](https://github.com/frida/frida).

Here comes the most popular and widely used method.

Step 1: Make sure the frida server is running on the android device

Step 2: Attach your target application with frida and run your favorite SSL bypassing script.

```
C:\Users\kishor>frida -l multiple-pinning-bypass.js -U -f [REDACTED] application --no-pause
    /_|  Frida 15.1.12 - 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/
Spawned ` [REDACTED] application` Resuming main thread!
[SM M105F] [REDACTED] application]->
=====
[#] Android Bypass for various Certificate Pinning methods [#]
=====
[-] OkHTTPv3 {1} pinner not found
[-] OkHTTPv3 {2} pinner not found
[-] OkHTTPv3 {3} pinner not found
[-] OkHTTPv3 {4} pinner not found
[-] Trustkit {1} pinner not found
[-] Trustkit {2} pinner not found
[-] Trustkit {3} pinner not found
[-] Appcelerator PinningTrustManager pinner not found
[-] Fabric PinningTrustManager pinner not found
[-] OpenSSLSocketImpl Conscrypt {1} pinner not found
[-] OpenSSLSocketImpl Conscrypt {2} pinner not found
[-] OpenSSLEngineSocketImpl Conscrypt pinner not found
[-] OpenSSLSocketImpl Apache Harmony pinner not found
[-] PhoneGap sslCertificateChecker pinner not found
[-] IBM MobileFirst pinTrustedCertificatePublicKey {1} pinner not found
[-] IBM MobileFirst pinTrustedCertificatePublicKey {2} pinner not found
[-] IBM WorkLight HostNameVerifierWithCertificatePinning {1} pinner not found
[-] IBM WorkLight HostNameVerifierWithCertificatePinning {2} pinner not found
[-] IBM WorkLight HostNameVerifierWithCertificatePinning {3} pinner not found
[-] IBM WorkLight HostNameVerifierWithCertificatePinning {4} pinner not found
[-] Conscrypt CertPinManager (Legacy) pinner not found
```

Following are my favorite scripts:

<https://codeshare.frida.re/@akabe1/frida-multiple-unpinning/>

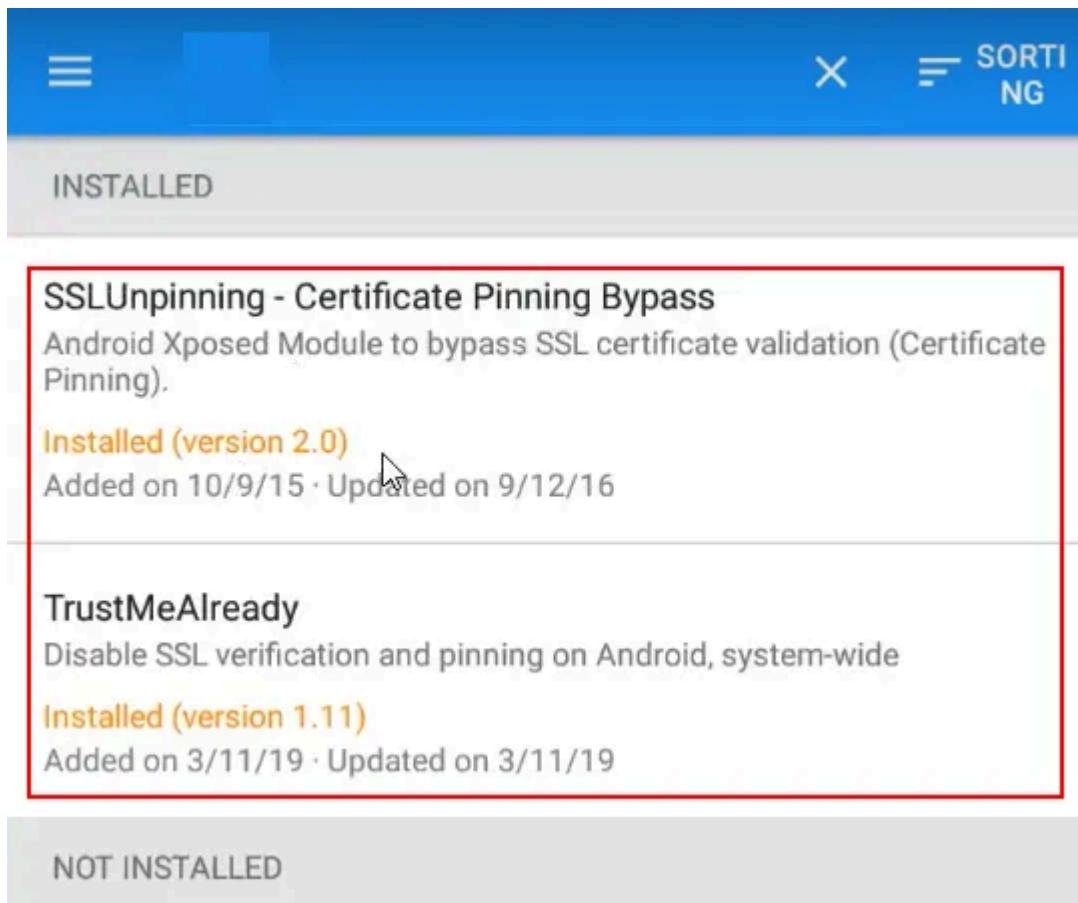
<https://codeshare.frida.re/@pcipolloni/universal-android-ssl-pinning-bypass-with-frida/>

2.4 Using Xposed Framework

If your device is rooted with **Xposed framework**, then you can try the following modules to bypass the pinning

1. [ac-pm/SSLUnpinning_Xposed: Android Xposed Module to bypass SSL certificate validation \(Certificate Pinning\).](#) (github.com)

2. ViRb3/TrustMeAlready: 🔒 Disable SSL verification and pinning on Android, system-wide ([github.com](#))



2.5 Using apk-mitm

apk-mitm is a CLI application that automatically prepares Android APK files for HTTPS inspection by modifying the apk files and repacking.

Repo:

[shroudedcode/apk-mitm: 🛡️ A CLI application that automatically prepares Android APK files for HTTPS inspection \(github.com\)](#)

apk-mitm can be pulled out using npm.

```
C:\Users\kishor>npm install -g apk-mitm
          ↗
added 142 packages, and audited 143 packages in 34s

18 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Step 1: Run the apk-mitm as shown in below.

```
c: \tools>apk-mitm app.apk
          ↗
apk-mitm v1.2.1
apktool v2.6.1
uber-apk-signer v1.2.1

Using temporary directory:
( [REDACTED] )-b0a74f454c40d4581567ba6f78b9ff9d

✓ Checking prerequisites
✓ Decoding APK file
✓ Applying patches
✓ Encoding patched APK file
✓ Signing patched APK file

Done! Patched file: ./app-patched.apk
```

That's it, apk-mitm has done its part. Now we can install the patched apk and intercept the application traffic.

2.6 Modifying the `network_security_config.xml` file

The Network Security Configuration lets apps customize their network security settings through a *declarative configuration file*. The entire configuration is contained within this XML file, and no code changes are required.

Source: [Network security configuration | Android Developers](#)

The Network Security Configuration works in **Android 7.0 or higher**.

Step 1: Decompile the android application with apktool or alternatives. And locate the `network_security_config.xml` file under `/res/xml`.

Step 2: The file may look like this if the app has pinned its own CA certificates.

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config>
        <domain includeSubdomains="true">example.com</domain>
        <pin-set expiration="2018-01-01">
            <pin digest="SHA-256">7HIpactkIAq2Y49orF00QKurWxmmSFZhBCoQYcRhJ3Y=</pin>
            <!-- backup pin -->
            <pin digest="SHA-256">fwza0LRMXouZHRC8Ei+4PyuldPDcf3UKg0/04cDM1oE=</pin>
        </pin-set>
    </domain-config>
</network-security-config>
```

Image source: developer.android.com

Step 3: Remove that <pin-set>... </pin-set> tag section and add the following:

```
PS C:\[REDACTED]\[REDACTED]\[REDACTED]> type network_security_config.xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config>
        <domain includeSubdomains="true">example.com</domain>
        <trust-anchors>
            <certificates src="user" />
            <certificates src="system" />
        </trust-anchors>
    </domain-config>
</network-security-config>
```

Step 4: Now save the file and Re-pack the application using apktool and uber-apk-signer (For signing the modified apk).

Thats it, install our new apk and your are good to go.

3. What if the application is not getting intercepted and also not showing any errors !!

Here the first thing pop-up in my mind is “Flutter”. The flutter based applications are basically “Proxy unaware”.

So here comes our hero “Reflutter” :

“This framework helps with Flutter apps reverse engineering using the patched version of the Flutter library which is already compiled and ready for app repacking.”

Repo: <https://github.com/Impact-I/reFlutter>

Step 1: install the reflutter using pip

Step 2: Follow the commands illustrated in the below screenshot.

The screenshot shows a terminal window with the following text:

```
reflutter <apk>.apk
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: <redacted>.104
Wait...
SnapshotHash: e4a09dbf2bb120fe4674e0576617a0dc
The resulting apk file: ./release.RE.apk
Please sign,align the apk file
Configure Burp Suite proxy server to listen on *:8083
Proxy Tab -> Options -> Proxy Listeners -> Edit -> Binding Tab
Then enable invisible proxying in Request Handling Tab
Support Invisible Proxying -> true
```

Step 3: Sign the application using uber-apk-signer or any alternatives and install it.

Step 4: Now in Burp proxy, Start listening the port 8083 and also enable “**Support Invisible Proxying**”.

The screenshot shows the Burp Suite interface. On the left, under 'Proxy Listeners', there are two entries: '127.0.0.1:8080' and '*:8083'. The second entry is selected. On the right, the 'Edit proxy listener' dialog is open with the 'Request handling' tab selected. It contains fields for 'Redirect to host' and 'Redirect to port', and a checkbox for 'Force use of TLS'. Below this, a note says 'Invisible proxy support allows non-proxy-aware clients to connect directly to the listener.' with a checked checkbox for 'Support invisible proxying (enable only if needed)'. The entire 'Support invisible proxying' section is highlighted with a red box.

That's it peeps, you are all good to go ... !

4. My application is using HTTP only but Still I am not able to Intercept!!

Hmm..That's a kinda weird , But it happens sometimes.

Applications with this behaviour, are basically called “Proxy Unaware” applications. Such applications route the traffic directly to the internet without cooperating with system wide Proxy settings.

Time to bypass:

For this method, I would like to thank brother Faris ❤.

[\(60\) Faris Mohammed | LinkedIn](#)

Step 1: Find out the domain address to which the App is communicating using Wireshark. Shown Below.

The screenshot shows a Wireshark capture. A specific HTTP request is highlighted with a red box. The URL in the request is `http://[REDACTED].com/services/ajax_v6.7.aspx?app=android&ver=6.64&lang=en&countryId=1&cat=authenticateUser]`. The text '[Full request URI: http://[REDACTED].com/services/ajax_v6.7.aspx?app=android&ver=6.64&lang=en&countryId=1&cat=authenticateUser]' is also highlighted with a red box. Below the main text, there are sections for 'File Data' (79 bytes) and 'HTML Form URL Encoded' (application/x-www-form-urlencoded), with a sub-item 'Form item: "password" = [REDACTED]'. The entire URL and its description are highlighted with a red box.

Step 2: Decompile the application using apktool

Step 3: Enter the decompiled folder and use the ack/grep tool to find out the file in which the domain name is mentioned.

```
(root💀aspire)-[~/mnt/c/Users/Faris/Downloads/New folder/app]
# ack pci.[REDACTED].com
smali_classes3/com/[REDACTED]/[REDACTED]ers/utils/GlobalFunctions.smali
303: const-string v0, "http://[REDACTED].com/"
```

Step 4: Replace the domain name with the IP address and Port of BurpSuite.

```
.line 52
new-instance v0, Lcom/nine[REDACTED]/[REDACTED]ers/utils/GlobalFunctions;

invoke-direct {v0}, Lcom/nine[REDACTED]/[REDACTED]ers/utils/GlobalFunctions;--><init>()V
    sput-object v0, Lcom/nine[REDACTED]/[REDACTED]ers/utils/GlobalFunctions;-->INSTANCE:Lcom/nine[REDACTED]/[REDACTED]ers/utils/GlobalFunctions;
        const-string v0, "http://192.168.137.1:8080/"

.line 76
```

Step 5: Re-pack the application, sign it and install it on the android device.

Step7: In the BurpSuite proxy, From the Request handling tab, give redirect host and port as the original domain address which was used by app in the first place.

Edit proxy listener

Binding Request handling Certificate TLS Protocols

?

These settings control whether Burp redirects requests received by this listener.

Redirect to host: [REDACTED].com

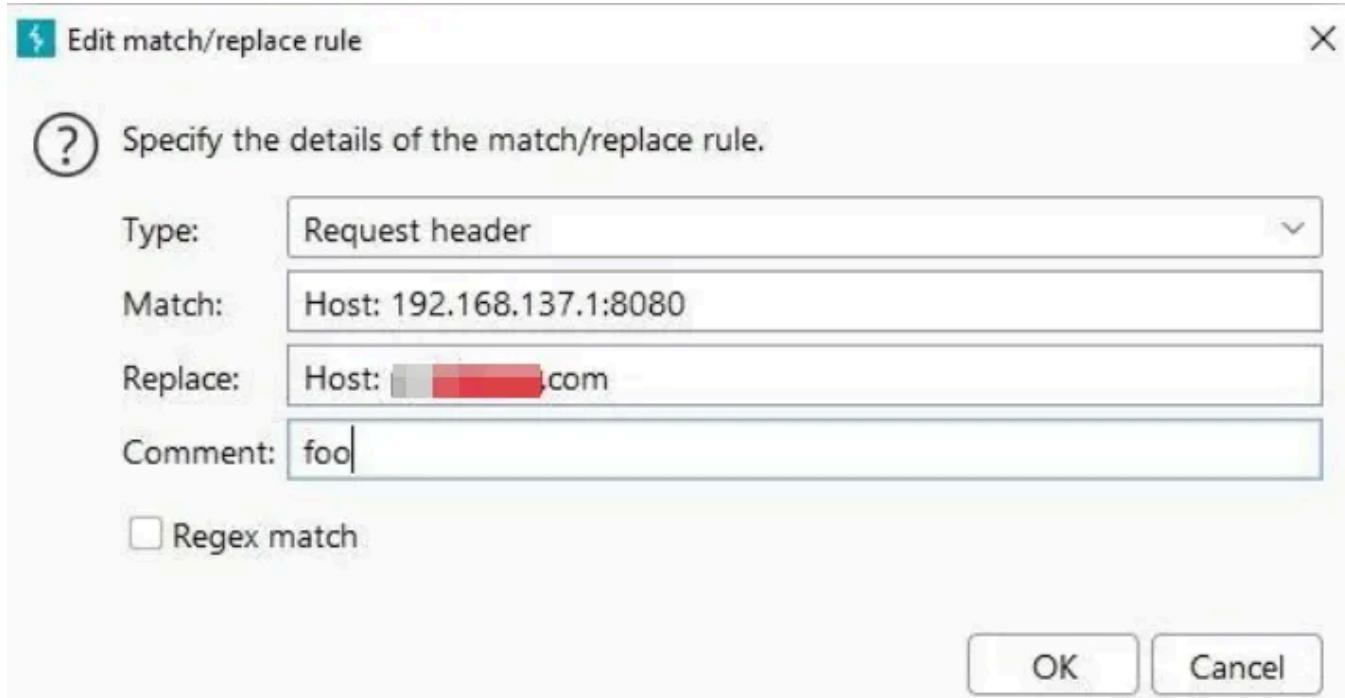
Redirect to port: 80

Force use of TLS

Invisible proxy support allows non-proxy-aware clients to connect directly to the listener.

Support invisible proxying (enable only if needed)

Step 8: Setup match and replace in the proxy options to replace the Host header value from the burp listener IP address to original domain address of the application



Thats it. Now the application's HTTP traffic will be captured in our Burp proxy.

Note: — Here, Since we replace the hardcoded application domain, we don't need to setup device proxy since the application directly communicates with the hardcoded domain (We have replaced it with our proxy IP).

So I think I've done for now. In fact, there are some more other methods that we can use to bypass the android SSL Pinning. I will try to make a Part 2 with that based on your responses.

Thanks peeps, Happy hacking :)

Cybersecurity

Android

Security

Mobile Pentesting

Hacking



Following


[Open in app ↗](#)


Search



More from Kishor balan

pentest's iPhone

Info

- Apps (12)
- Photos
- Music
- Ringtones
- Videos
- Books
- UDisk
- Data
- Files
- Common tools (1)

pentest's iPhone ⓘ

[Reboot](#) [Turn Off](#) [Refresh](#)

iPhone 7 [32GB] Black		PC Charging(2W) ⚡ [4%]	
iOS Version	13.3.1 (17D50)	Apple ID Lock	On Online Query
Jailbroken	Yes Install AFC2	iCloud	On Details
Activated	Yes	Prod. Date	01/22/2017
Product Type	iPhone9,1 (A1779)	Warranty Date	Online Query
Sales Model	MNCE2 J/A	Sales Region	Japan
IMEI	[REDACTED]	CPU	Apple A10 Quad Details
Serial No.	[REDACTED] G7X	Disk Type	MLC Details
ECID	[REDACTED] 026	Charge Times	76 Times
Verify UDID	Yes	Battery Life	100% Details
UDID	[REDACTED] B11B	View Verification Report View iDevice Details	
Hard Disk Capacity		11.65 GB / 29.79 GB	
■ System ■ Apps ■ Photos ■ Media ■ UDisk ■ Used ■ Free			

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



...

```
[Read 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:');
  },
});
```

 Kishor balan

iOS Pentesting Series Part 3- The Ceasefire

Hola mates,

7 min read · Aug 19, 2023

 22

...



```
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,

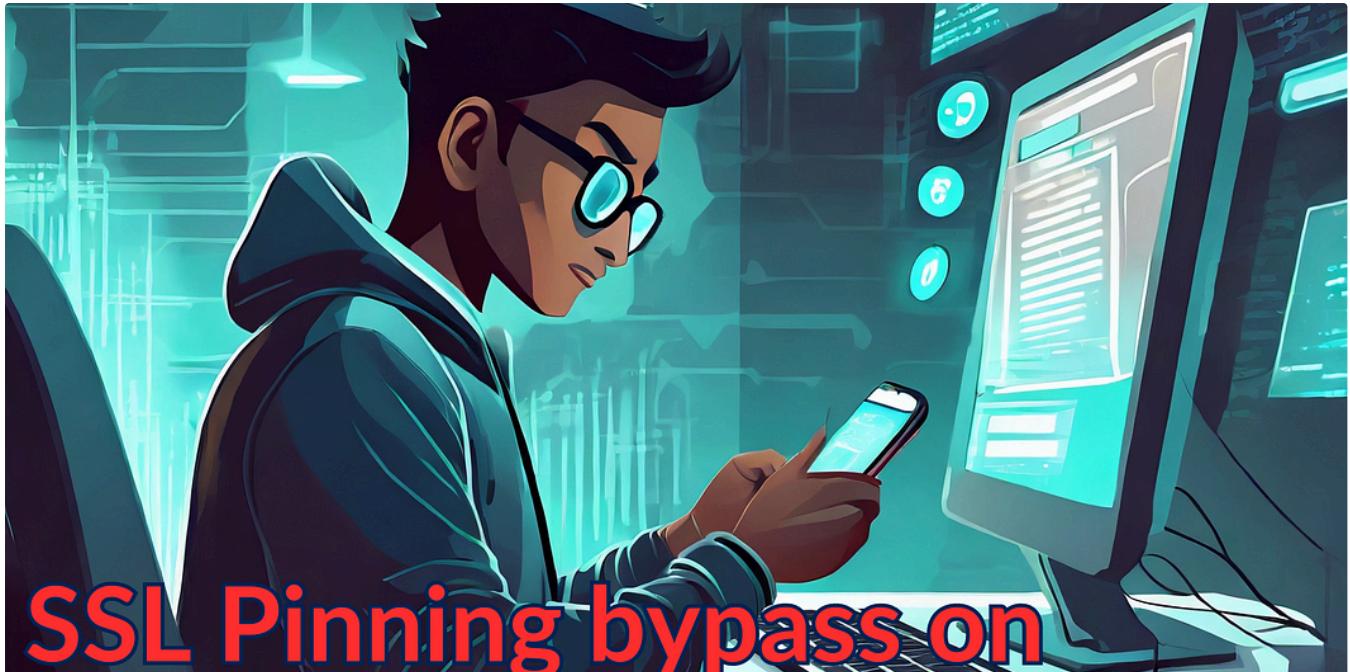
6 min read · Oct 16, 2022

 185

...

[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  +

...

 Luxeedd

Mobile Pentest Dynamic Analysis - Hooking with Frida

Introduction

3 min read · Nov 22, 2023



...

Lists



Tech & Tools

16 stories · 212 saves



Medium's Huge List of Publications Accepting Submissions

285 stories · 2528 saves



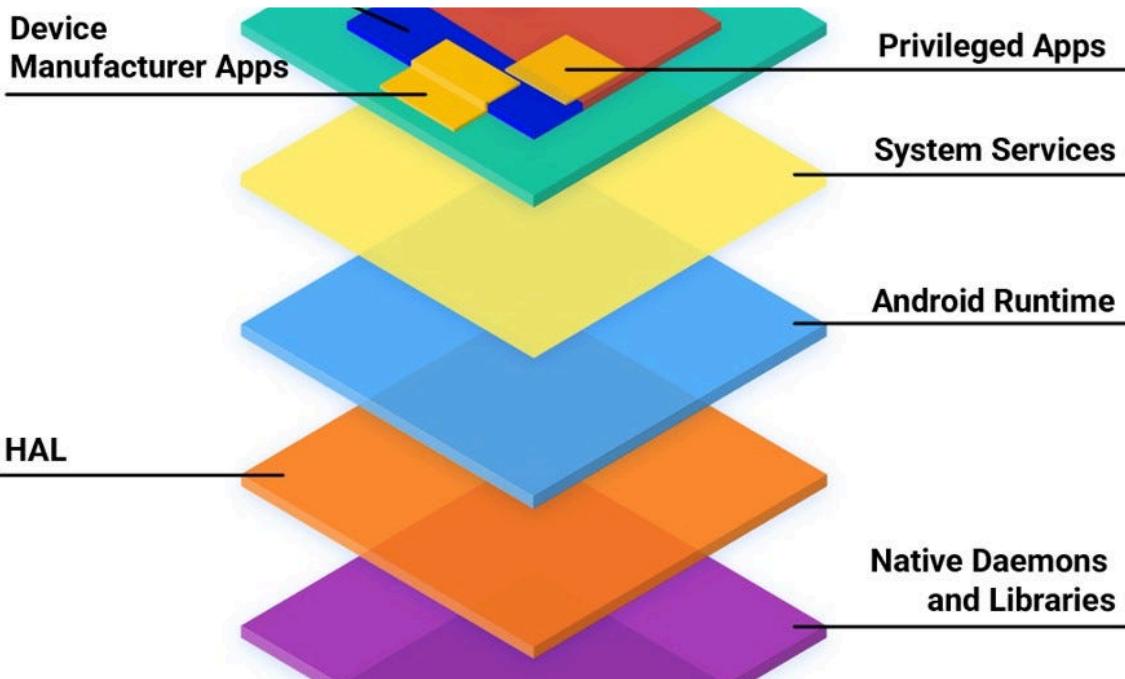
Staff Picks

630 stories · 922 saves



Natural Language Processing

1407 stories · 905 saves



Rudrani 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



...

Web links

Handle HTTP / HTTPS schemes

Android
App Links

Z4ki

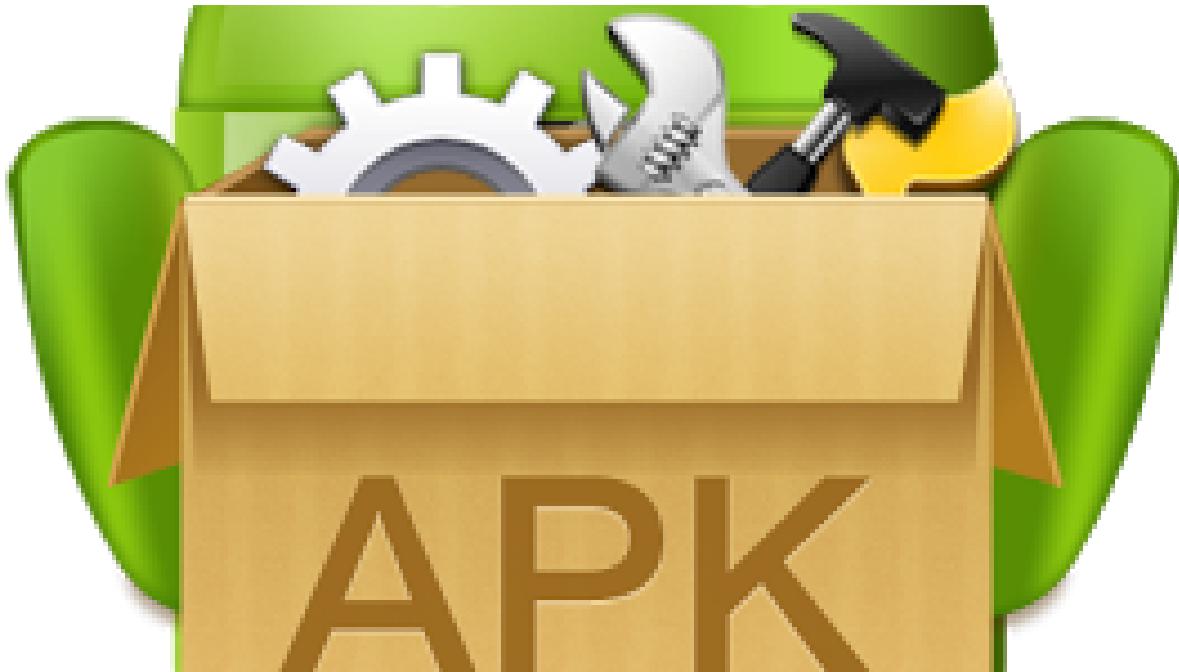
Android Deep Links exploitation

Recently, I have been studying Android penetration testing, and I came across a new topic called 'Deep Links'. Since it is a new topic to...

5 min read · Oct 30, 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

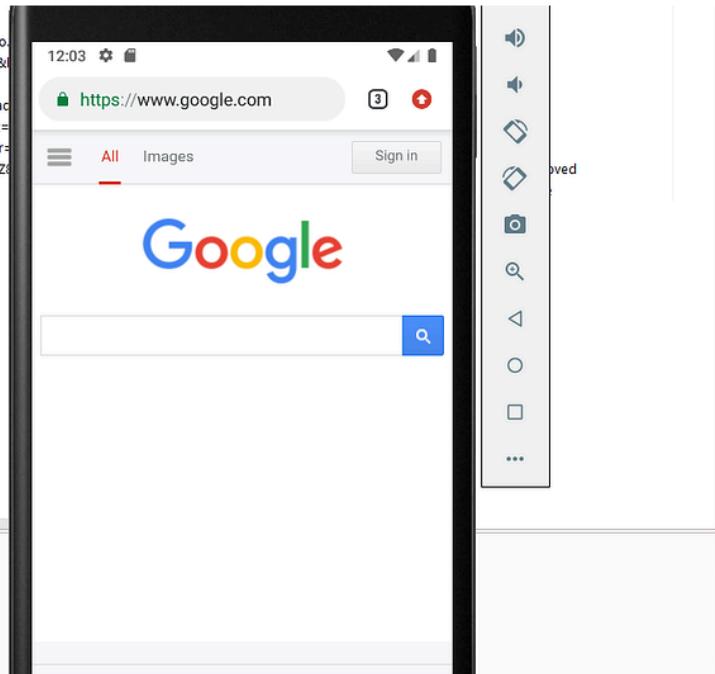


2



...

```
119 http://connectivitycheck.gstatic.com      GET  /generate_204
118 https://mtalk.google.com:5228             '000  Ogms-20.24.14-00000mcs.android.co...
122 https://translate.googleapis.com        GET  /translate_a/?client=chrome&hl=en&l...
120 https://www.google.com                  GET  /generate_204
121 https://www.google.com                  GET  /gen_204?atyp=i&ct=backbutton&cad...
123 https://www.google.com                  GET  /gen_204?atyp=i&ct=psnt&cad=&nrt...
124 https://www.google.com                  GET  /gen_204?atyp=i&ct=nrr&cad=&nrrr...
125 https://www.google.com                  GET  /setprefs?sig=0_vfoER4BEzit2-JkupRZ...
126 https://www.google.com                  GET  /
```



Shayan Ahmed Khan

RevEng! Reverse Engineering android apps to bypass SSL pinning for mobile app pen-testing

This is the second part of my 2 part blog series on mobile app pen-testing and reverse engineering. In the first part, I have explained how...

5 min read · Oct 31, 2023



...

See more recommendations