




# **HTTPS Decryption & SSL Unpinning for Android Apps**



**DISCLAIMER:** This guide is intended exclusively for lawful research, debugging, and development activities. The techniques described herein must only be applied to applications owned by the user or those with explicit, documented permission for testing. All information is to be used strictly in non-production environments.

Misuse of the material may breach laws or regulations, for which the user assumes full responsibility. The author disclaims liability for unauthorized or unethical application of these methods



# Required Tools

- Android Studio (<https://developer.android.com/studio>)
- Google Platform Tools (<https://developer.android.com/tools/releases/platform-tools>)
- CharlesProxy (<https://www.charlesproxy.com/>)
- rootAVD (<https://github.com/newbit1/rootAVD>)
- MagiskTrustUserCerts (<https://github.com/NVISOsecurity/MagiskTrustUserCerts>)



# Additional Tools

- android-unpinner (<https://github.com/mitmproxy/android-unpinner>)
- JustTrustMe (<https://github.com/Fuzion24/JustTrustMe>)

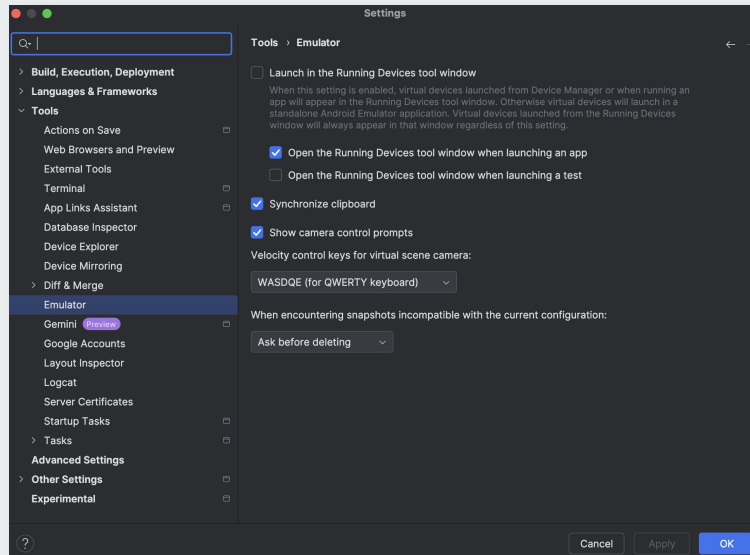


# Useful Resources

- ApkMirror.com
- Aurora Store App (<https://gitlab.com/AuroraOSS/AuroraStore/-/releases>)
- Apkeditor.io

# Step 1: Installing Android Studio

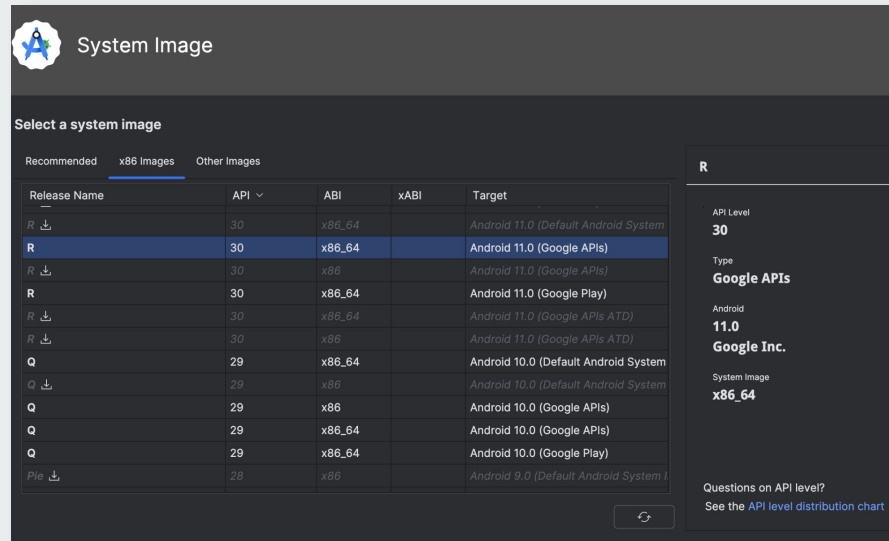
- Download and install Android Studio.
- Launch it, head over to *Settings>Tools>Emulator* and uncheck: *“Launch in the Running Device tool window”*



## Step 2: Configuring the Emulator

From Device Manager tab, configure the the virtual device and launch it:

- Pixel 7
- SDK 30 x86\_64 (R)
- Google API version



System Image

Select a system image

Recommended x86 Images Other Images

Release Name	API	ABI	xABI	Target
R	30	x86_64		Android 11.0 (Default Android System)
R	30	x86_64		Android 11.0 (Google APIs)
R	30	x86		Android 11.0 (Google APIs)
R	30	x86_64		Android 11.0 (Google Play)
R	30	x86_64		Android 11.0 (Google APIs ATD)
R	30	x86		Android 11.0 (Google APIs ATD)
Q	29	x86_64		Android 10.0 (Default Android System)
Q	29	x86		Android 10.0 (Default Android System)
Q	29	x86		Android 10.0 (Google APIs)
Q	29	x86_64		Android 10.0 (Google APIs)
Q	29	x86_64		Android 10.0 (Google Play)
Pie	28	x86		Android 9.0 (Default Android System)

R

API Level  
30

Type  
Google APIs

Android  
11.0

Google Inc.

System Image  
x86\_64

Questions on API level?  
See the API level distribution chart



## Step 3: Rooting the Emulator

- Start the emulator and navigate to the rootAVD folder in your terminal.
- List all available AVDs by running:  

```
$ ./rootAVD.sh ListAllAVDs
```
- Locate the relevant ramdisk.img file for your emulator from the displayed list. Replace {ramdisk-path} with its path and execute:  

```
$ ./rootAVD.sh system-images/android-30/google_apis/x86_64/ramdisk.img
```
- Reboot the emulator. The device should now be rooted, and the Magisk app will be installed automatically."





## Step 4: Configuring Charles Proxy

Download and launch Charles Proxy.

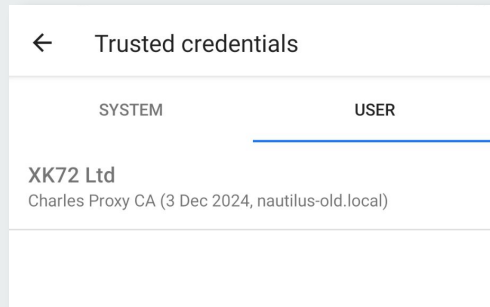
- Set the port to 8888
  - *Charles>Proxy>Proxy Settings> Port 8888*
- On the emulator, configure proxy in *Settings>Advanced* using local IP
  - *Charles, Help>Local IP Addresses*



## Step 5: Installing Charles Certificate

With Android 11, certificates must be installed via Settings.

- On emulator, visit <http://chls.pro/ssl> and install the certificate (if you are unable to download the certificate, set proxy in wifi settings then turn on and off airplane mode).
- Open Settings app and search for “CA certificate”
- Install the downloaded .crt file
- Search for “Trusted credentials” and check the installed Charles certificate.





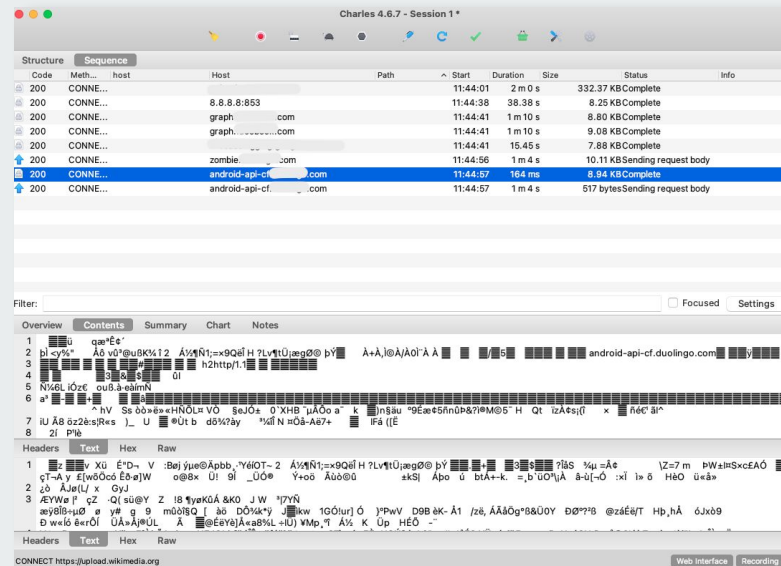
## Step 6: HTTPS Encrypted Traffic

Let's now check the traffic a Language learning app generates on Charles.

- Install the app .apk file on the Emulator.
- On Charles, go to *Proxy>Stop SSL Proxying*.
- Open the installed app and check Charles for the corresponding requests.

## Step 6: HTTPS Encrypted Traffic

As expected, HTTPS content appears encrypted.





## Step 7: Trusting User Certificate

We now need to convert the user-installed certificate to a System one.

In the emulator, open the Magisk app, let it perform initial actions and reboot the device if required.

- Copy the *AlwaysTrustUserCerts.zip* file to the emulator's internal storage:

```
$ adb push AlwaysTrustUserCerts.zip /sdcard/
```

- In Magisk app, navigate to Modules>Install from storage.
- Select the .zip file, then select *Reboot* after installation is complete.
- Verify that the Charles certificate is also present in System Trusted Credentials.



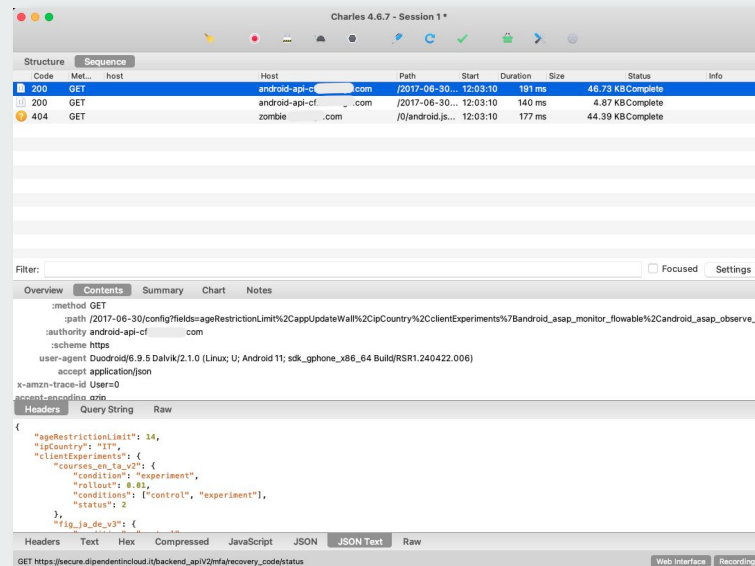
## Step 8: Decrypted HTTPS Traffic

Let's check requests on Charles with the System Certificate installed.

- Re-enable SSL proxying on Charles. *Proxy>Start SSL Proxying.*
- Open the target App on the emulator and check requests on Charles

# Step 8: Decrypted HTTPS Traffic

Congratulations! You are now able to access HTTPS requests content.





## Extra: SSL Unpinning

The *AlwaysTrustUserCerts* Magisk module is compatible with Android up to SDK 30. With more recent Android versions, I was not able to install certificates as System.

Furthermore, SSL pinning could be configured in the test app, only accepting certain certificate fingerprints. To bypass these limitation, SSL unpinning can be performed. This process modifies the APK file to remove certificate pinning, allowing you to trust custom user certificates.





## Extra: android\_unpinner

We can achieve SSL Unpinning leveraging android\_unpinner.

- Installation:

```
$ git clone https://github.com/mitmproxy/android-unpinner.git
$ cd android-unpinner
$ pip install -e
```

- With emulator running, run the following :

```
$ android-unpinner all {path-to-my-apk.apk}
```

- The unpinned application should be automatically installed on the emulator.

