Payatu

Device Setup and Barriers

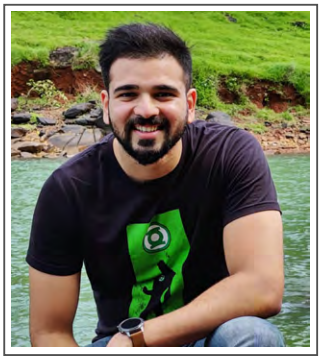
About the Author and Contributors

AUTHOR

AMIT KUMAR

Lead Security Consultant
Mobile Tower - Payatu

CONTRIBUTORS



VEDANT WAYAL

Security Consultant
Mobile
Payatu

ALI JUJARA

Security Consultant
Mobile
Payatu

KAPIL GURAV

Security Consultant
Mobile
Payatu

Table of Contents

1. SETTING UP ANDROID DEVICE

1.1 Genymotion 2

- Steps
- Troubleshooting

1.2 Ready to use Android Device Installation 4

- Device Description
- Pre-requisites
- Steps

1.3 Genymotion QEMU Hypervisor Mode. 7

- Steps

1.4 Android Studio Emulator. 9

- Steps
- Connect Android Virtual Device to VirtualBox Linux VM
- Troubleshooting

1.5 NoxPlayer Emulator 15

- Installation
- Working
- Troubleshooting

1.6 Ninjitsu Penetration Testing Environment 18

1.7 Setting Up the Android Physical Device 19

- Rooting an Android Device

1.8 Mobexler OS Installation 23

- Steps

2. TOOL SETUP

2.1 Magisk Installation 26

- Installing Magisk in the Genymotion device
- Installing magisk in AVD

- Installing Magisk on Physical Device
- Drozer via Mobexler
- One Click Proxy
- Setting up BurpSuite with Android Device

2.2 Useful Magisk Modules & Tools 47

3. SETTING UP THE iOS DEVICE

3.1 Initial Steps 49

3.2 Jailbreak 50

- Types of Jailbreak
- Which Jailbreak technique is better?

3.3 Installing Tools on iOS 53

- SSH
- Add Sources in Cydia

3.4 Installing Tools on Mac 59

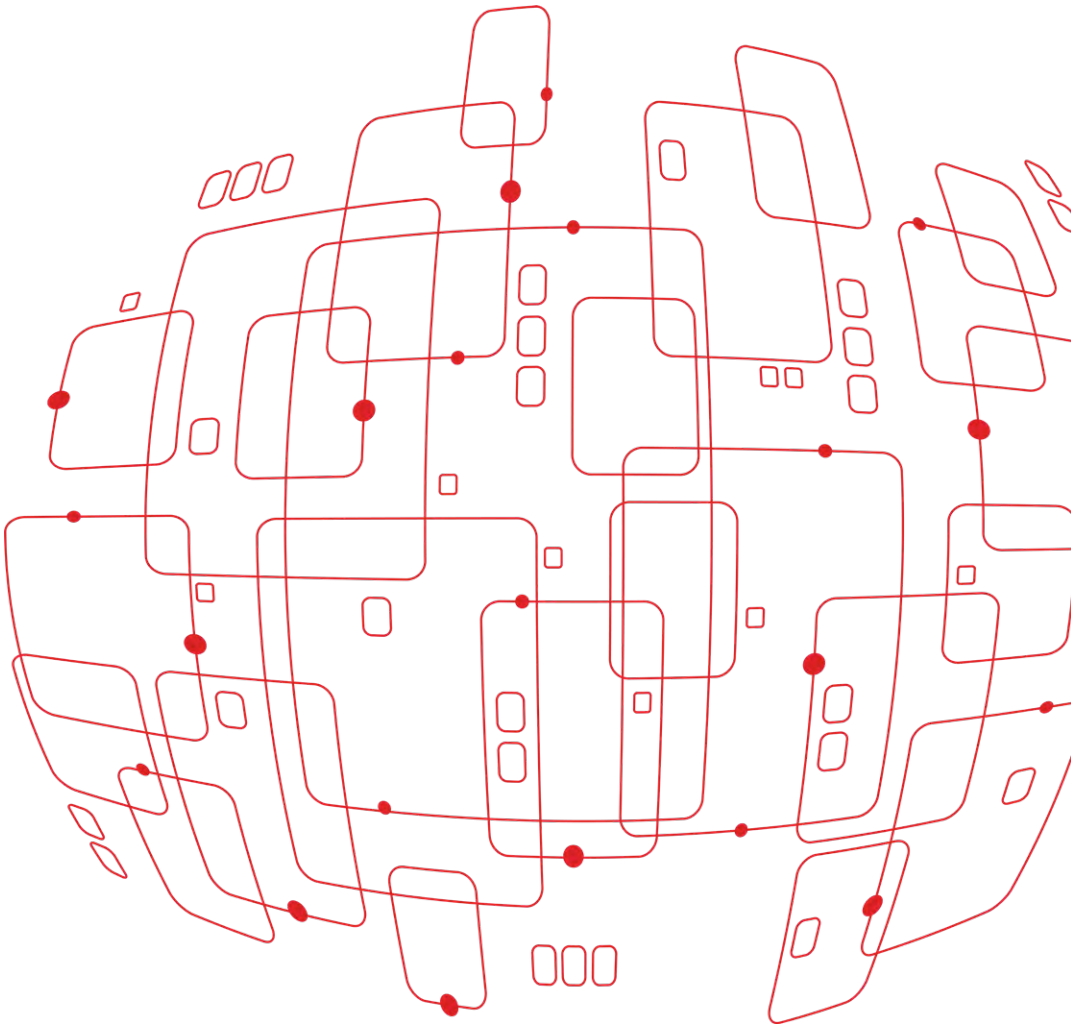
- Xcode
- Frida
- Ghidra
- Objection
- KeyChain Dumper
- Passionfruit
- SQLite Browser
- Realm Browser

3.5 Installing Tools on Windows 62

3.6 Useful tweaks & tools for iOS Device 63

References 67

About Payatu 68



1. SETTING UP ANDROID DEVICE

1.1 Genymotion

STEPS:

1. Download the Genymotion installer from here: Latest Genymotion
2. Select the installer platform as per your OS
3. Click on the setup file and proceed with your installation

TROUBLESHOOTING:

1. Unable to start a virtual device:
    a. Go to “Network and sharing settings”>>“Change adapter settings” >>Enable the “VirtualBox Host-Only Ethernet Adapter #2” adapter.

2. Device stuck while booting up:
    a. Uninstall VirtualBox 6.1.14. Reboot your PC.
    Download the VirtualBox 6.1.30 installer for Windows at
    https://download.virtualbox.org/virtualbox/6.1.30/VirtualBox-6.1.30-148432-Win.exe
    b. Install VirtualBox 6.1.30 Launch Genymotion. If you get the “unable to start the virtual device” error, reboot your PC.

3. Upon clicking on the device, it’s not booting up. (The booting screen is not visible)
    a. Open Virtualbox and check the device status. (It should be Fresh or Powered Off.)
    b. Double-click on the device and start the device in Virtualbox.
    c. Power off the device from Virtualbox and now open the device in Genymotion.

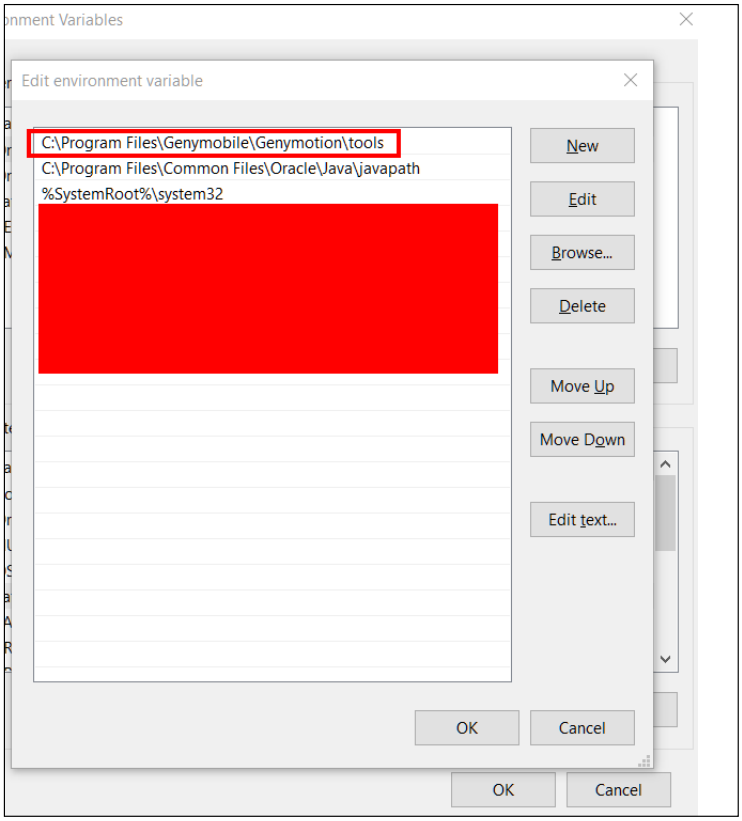
4. Device not connecting to local wifi, i.e. ***AndroidWifi***
    a. Open Virtualbox and open the settings of the device
    b. Go to the “Network” tab and open the “Adapter2” tab.
    c. Select the “Attached to:” option to “Nat” and save this setting.
    d. Now boot the device again, and you will be able to connect to “***AndroidWifi***”.

5. Getting
```
adb server version (41) doesn't match this client (39); killing...
```
error while connecting the Genymotion device via adb
    a. Genymotion has its own set of ADB tool locations in the Genymotion directory. The error occurs when you already have separate ADB tools in the system and have

environment variables set for them.

b. Solution:

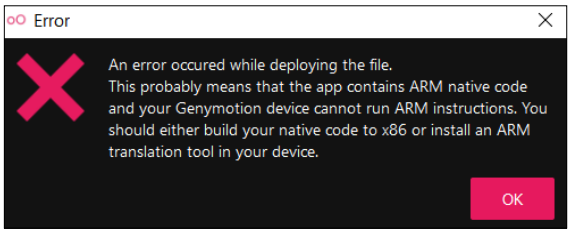
i. Remove the path of ADB from environment variable settings and replace the path with Genymotion's ADB tools which might be at `C:\Program Files\Genymobile\Genymotion\tools`



ii. ADB should be able to connect to the Genymotion device now. If the error still continues, restart the system.

iii. You can copy all other SDK tools, such as fastboot, into this folder.

6. Error while installing APK into the Genymotion device:



- Download the correct ARM translation archive for your device's Android version. For Android 8, we have used ARM_Translation_Oreo.zip. (Download Link)
- Drag and drop the .zip file to the device's unlocked screen and click on OK when asked for confirmation
- Restart the device and install the APK now.

1.2 Ready to Use Android Device Installation:

DEVICE DESCRIPTION:

1. Name:	Google Pixel 2XL
2. Android Version:	9.0
3. Processors:	4
4. Memory Size:	6096
5. Display:	1440*2880
6. Gapps:	Installed.

PRE-REQUISITES:

The following applications must be installed and should be working correctly.

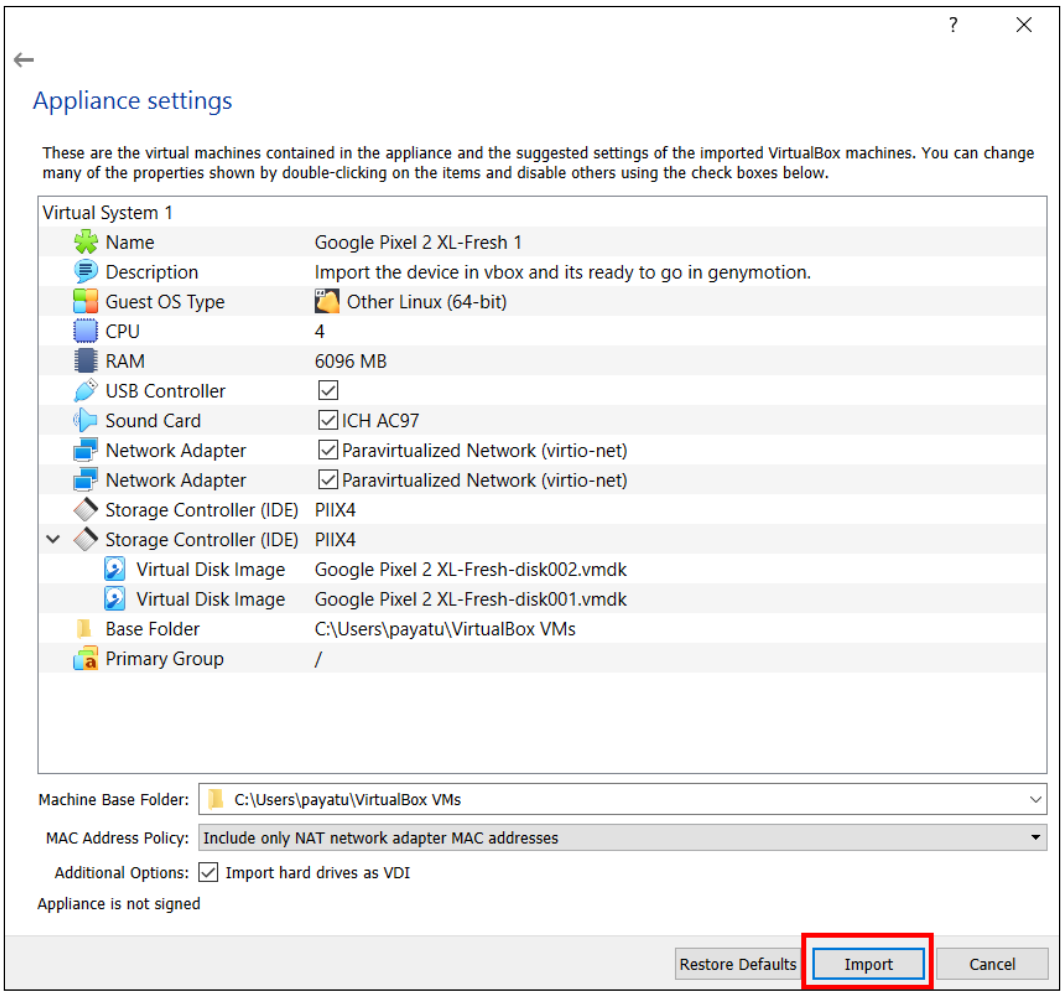
- VirtualBox
- Genymotion

STEPS:

1. Download the .ova file from here: Google Pixel 2XL
2. Close Genymotion software if it is already open.
3. Open VirtualBox and click on “Tools>>Import”.

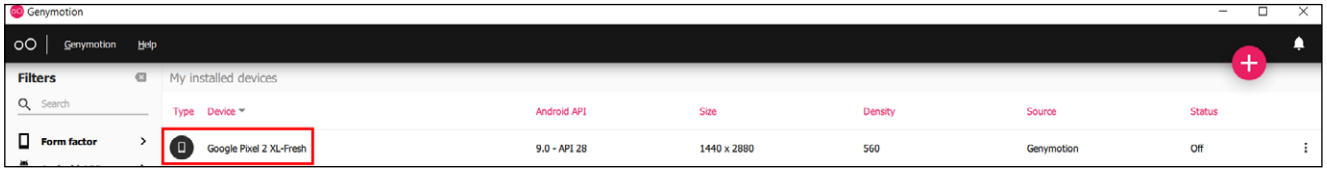
Import Option

4. Select the downloaded .ova file and click on “Next”.
5. Do not change any preloaded settings, and click on “Import”.
   This will import the .ova file.



Importing .ova file

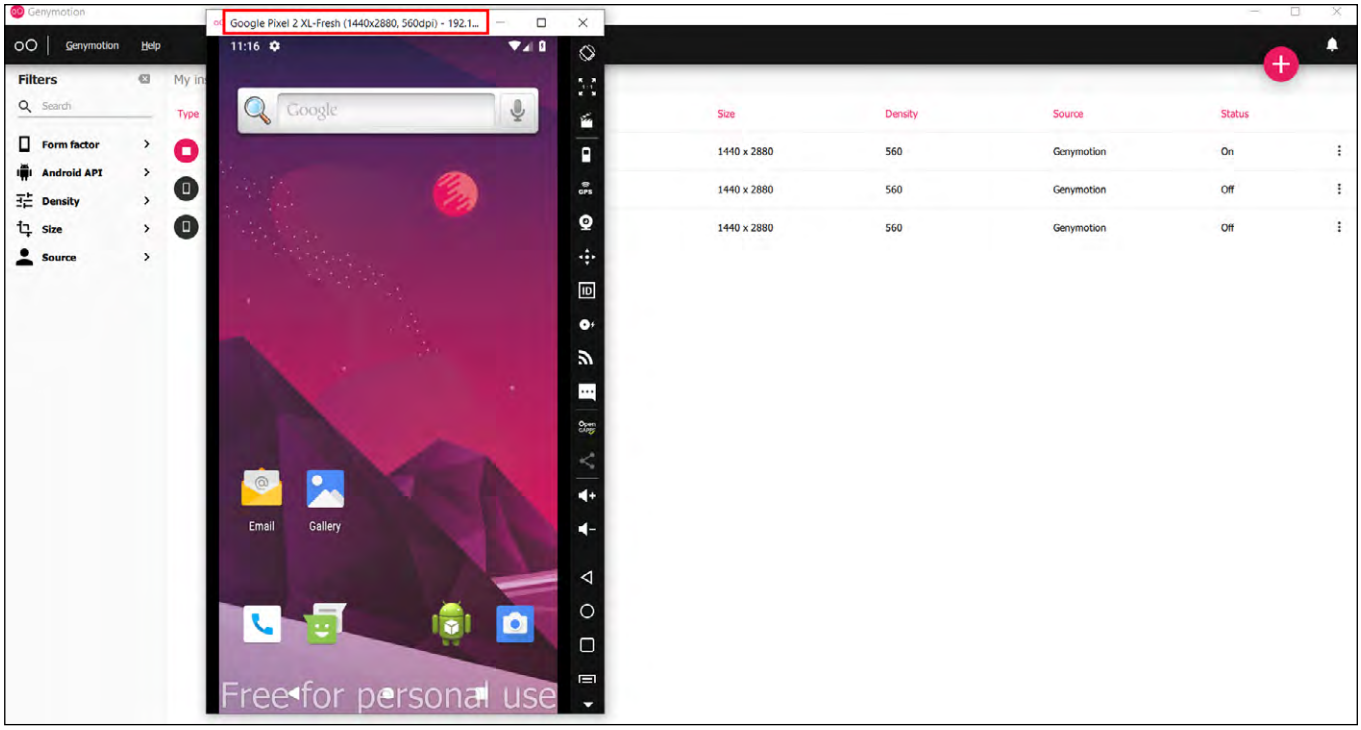
6. The device import will take some time. Once the import is finished, open Genymotion.
7. You will be able to see the imported device in Genymotion's "My installed devices" list.



Installed devices

8. Start this device. You will get some errors during these steps, or the device will not boot up. Just close the error window and start the device again. Do this 4-5 times.
    a. ***Why does this happen?***
    >> The imported device's adapter and your adapters will initially not match. Thus, VirtualBox will automatically create a new "Host-Only" adapter, which is compatible with the imported device during initial boot sessions. Once this adapter is created, the device will start successfully.
    b. If the device still does not boot up, reboot the system and try again.

9. The imported device will get booted up, and you can work with this device.



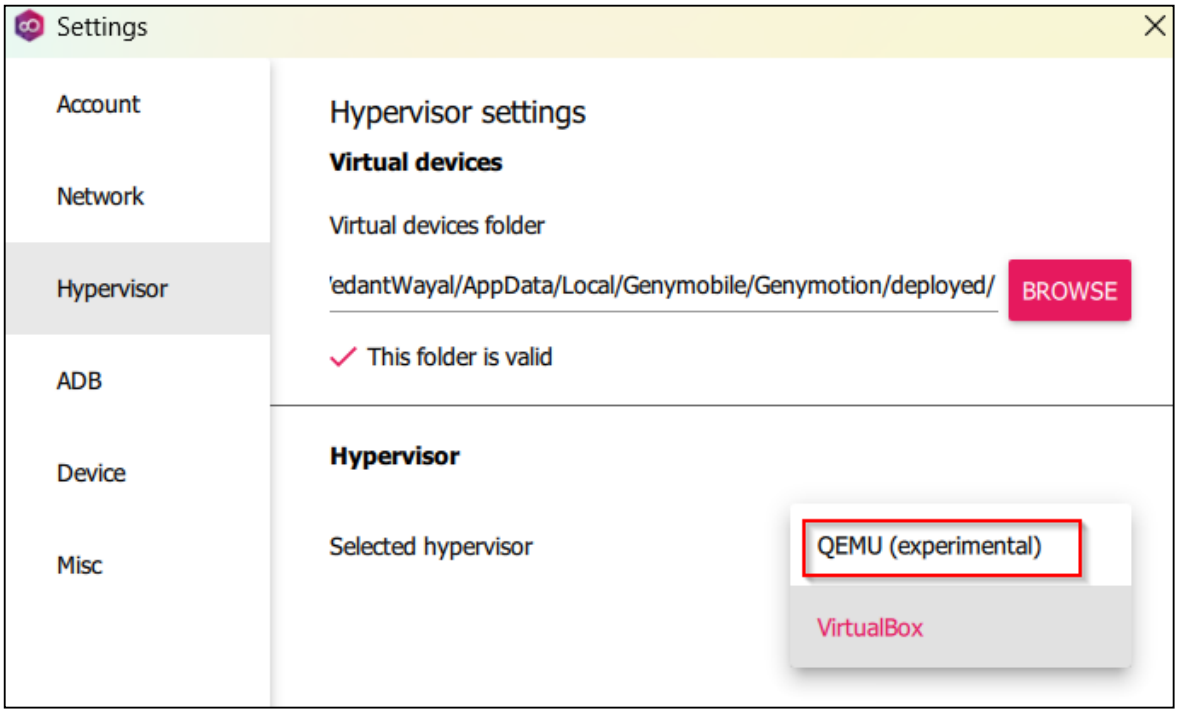
1.3 Genymotion QEMU Hypervisor Mode:

From Genymotion Desktop 3.3.0 onwards, it is possible to choose between VirtualBox and QEMU hypervisor. It is recommended to use the QEMU hypervisor over VirtualBox to avoid any Virtualbox-related issues.

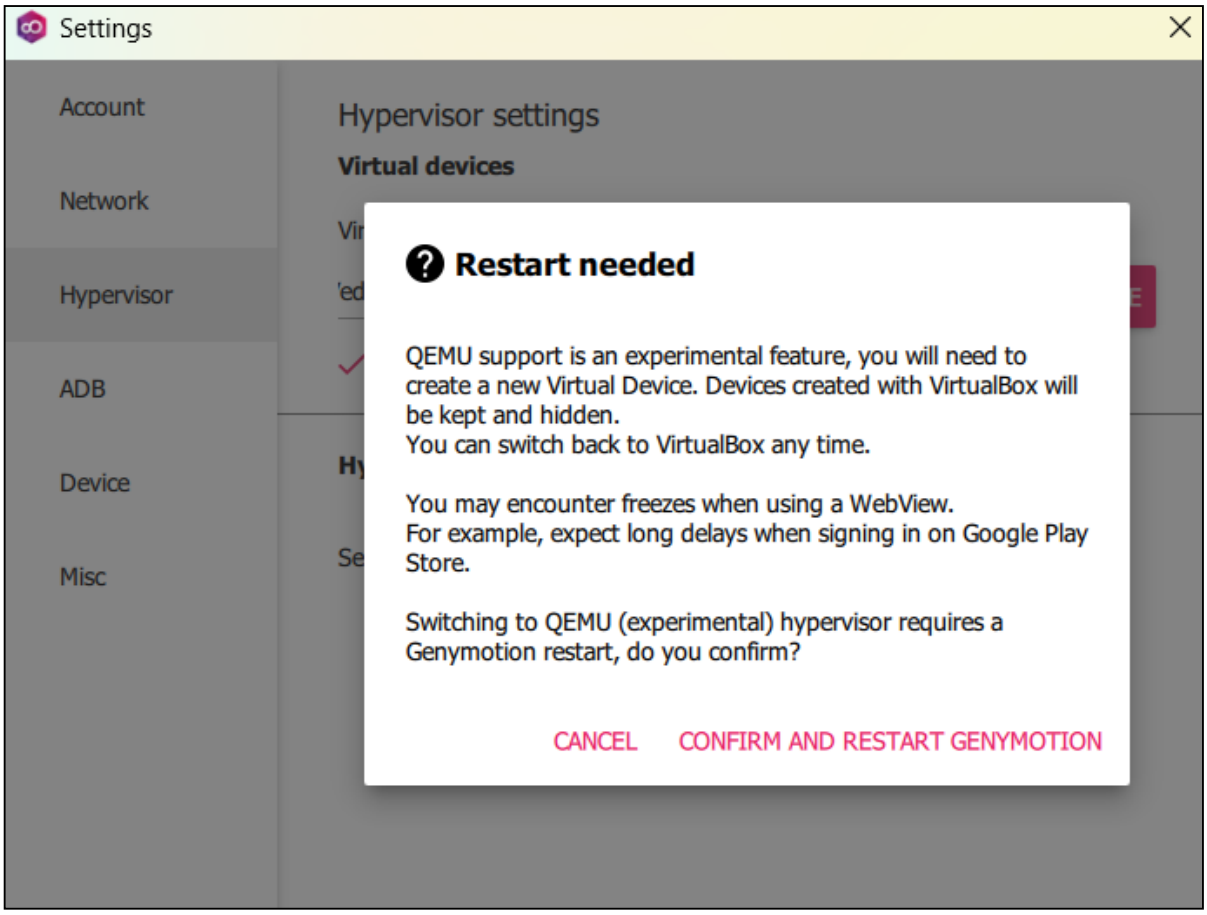
QEMU hypervisor setting currently supports Windows OS without any paid license. To use this setting on macOS or Linux OS, a paid license needed to be acquired.

STEPS:

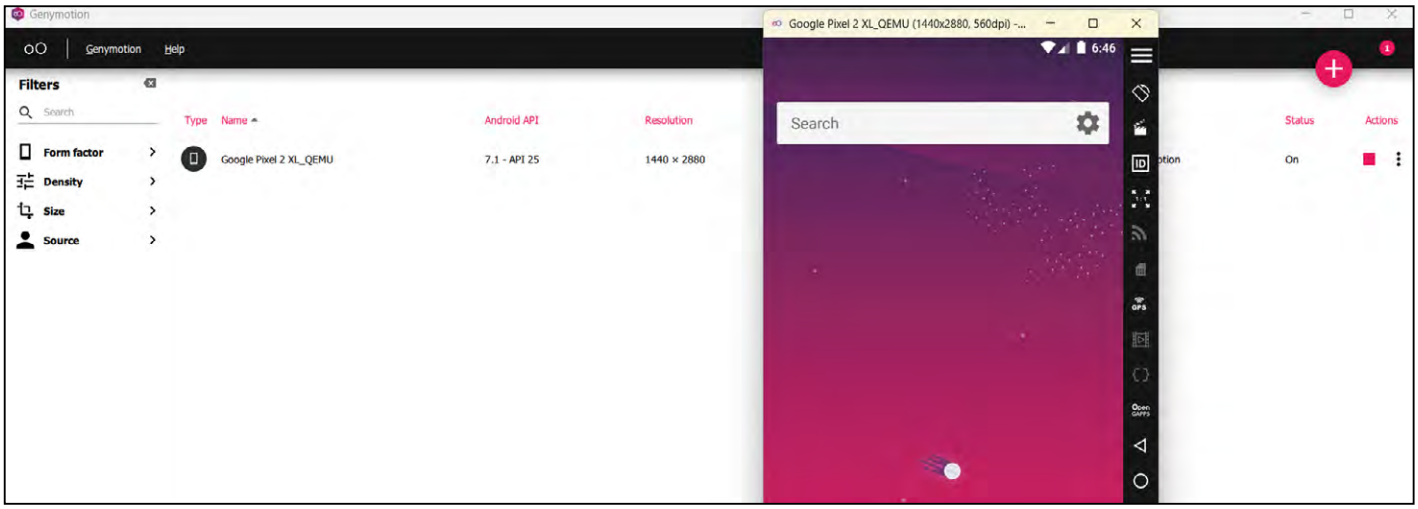
1. Download and install the latest version of Genymotion Dekstop (3.3.0 onwards).
2. Go to “Genymotion>>Settings>>Hypervisor” and select “QEMU (Experimental) as hypervisor instead of Virtualbox



3. Confirm and restart the Genymotion



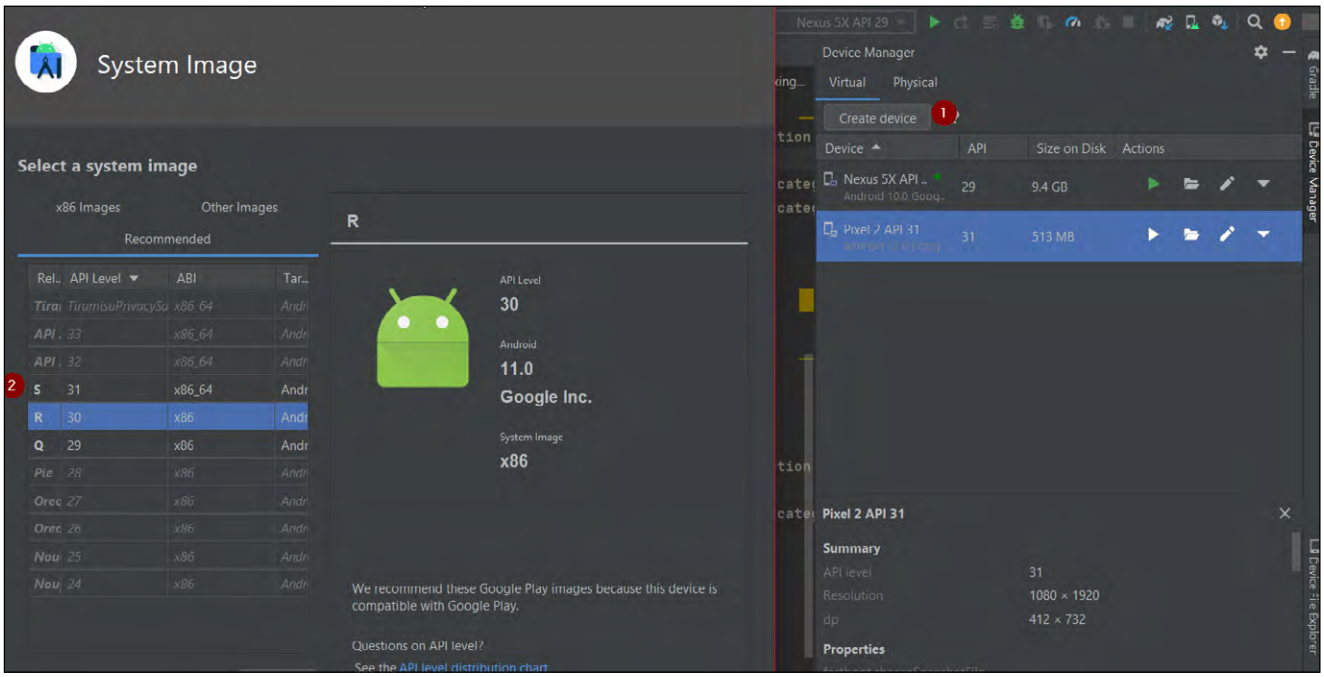
4. You can use Genymotion with QEMU hypervisor mode.



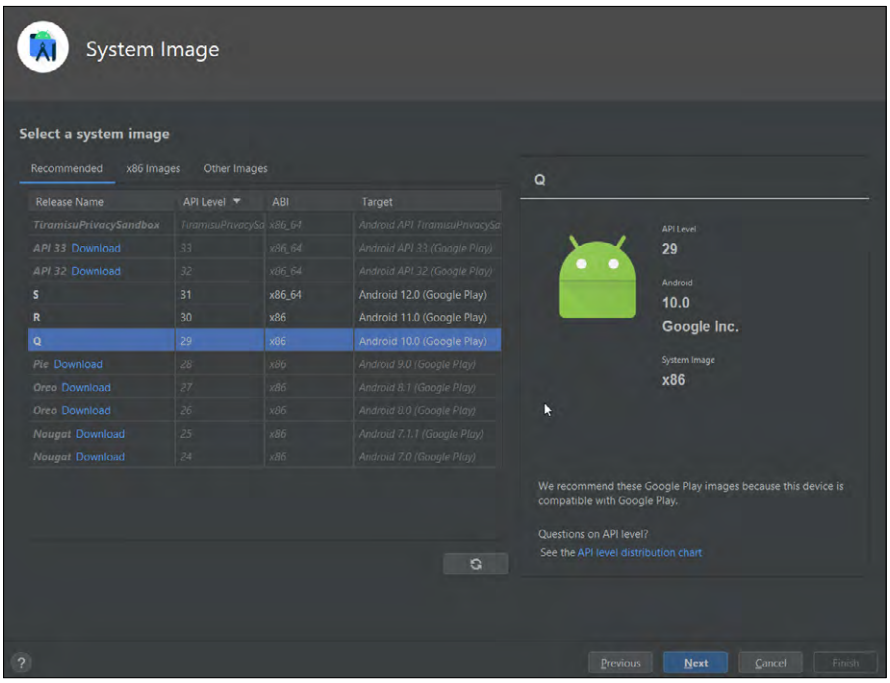
1.4 Android Studio Emulator:

STEPS:

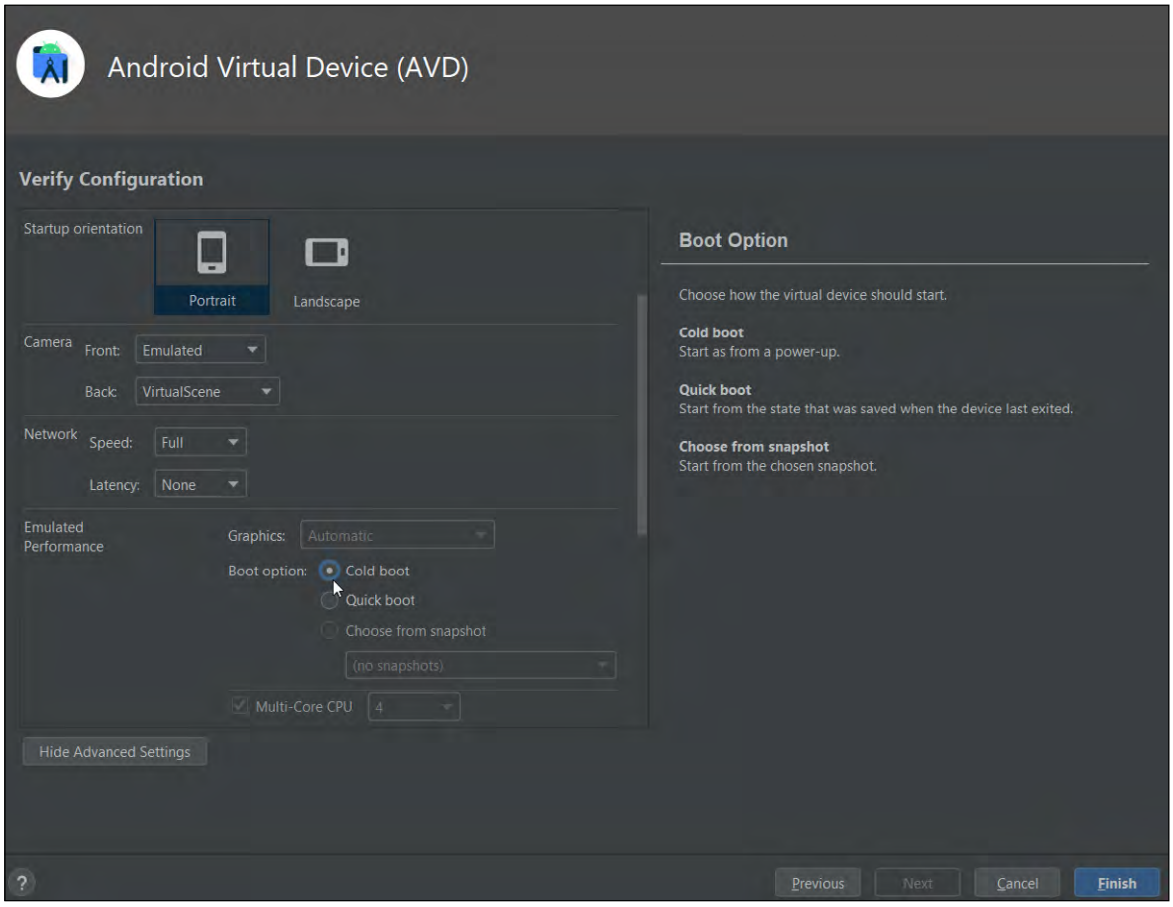
1. Download Android Studio and set up the Android Studio by following the default steps.
2. Click on “create image” and download the system images among (27,29,28,30) because, on other API versions, Magisk installation might not work correctly.



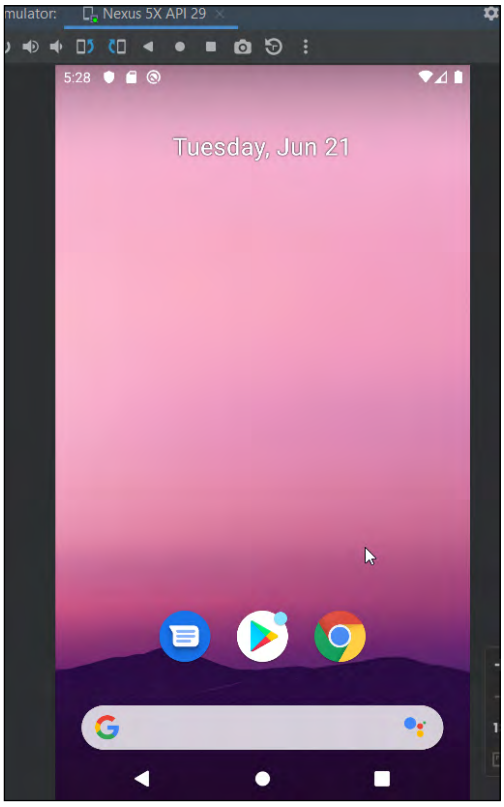
3. After successfully downloading system images, create the Emulator with the downloaded API version.



4. Go to advanced settings and choose “Cold Boot”.



5. Launch the Emulator.

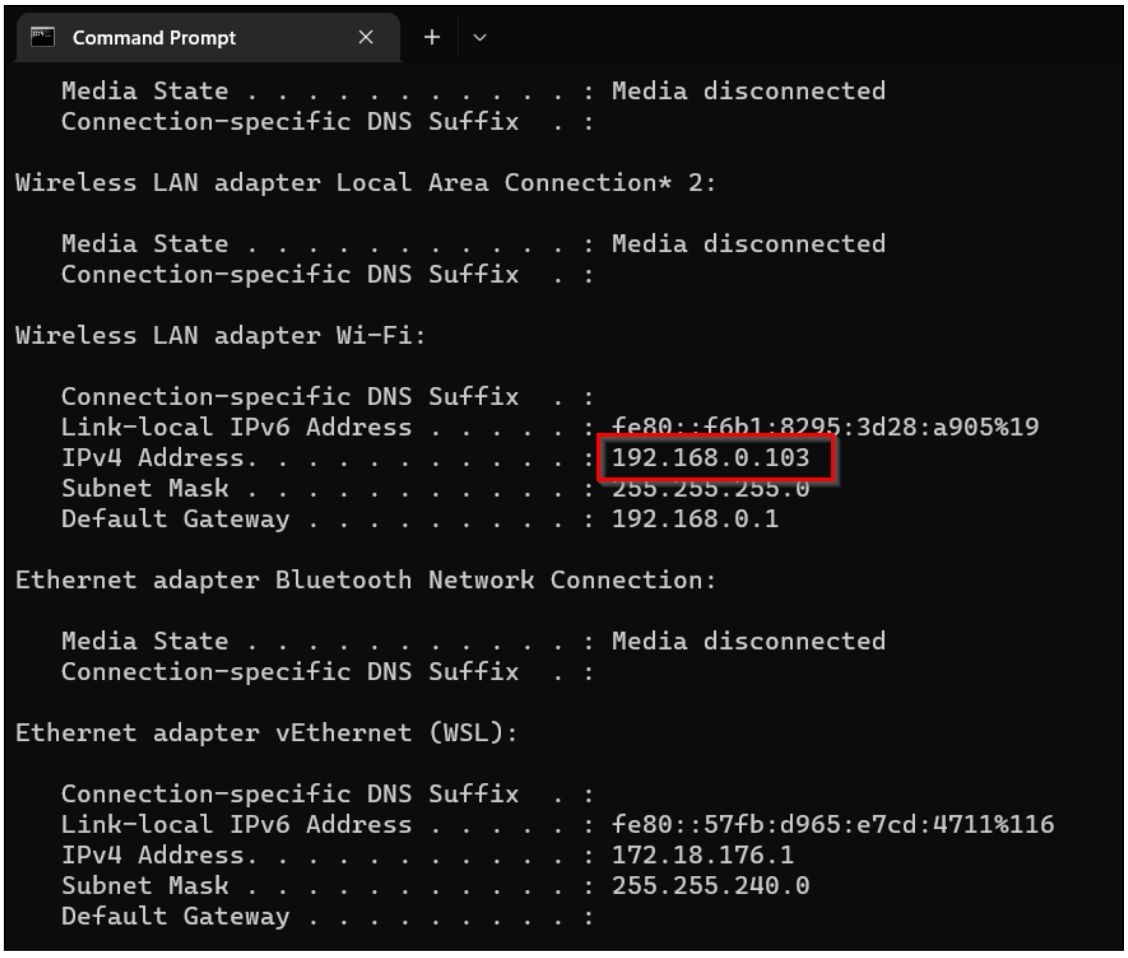


Connect Android Virtual Device to VirtualBox Linux VM

1. Start the AVD emulator.
2. Open cmd and type `adb kill-server`
3. Now type `adb -a nodaemon server` and let it run there.

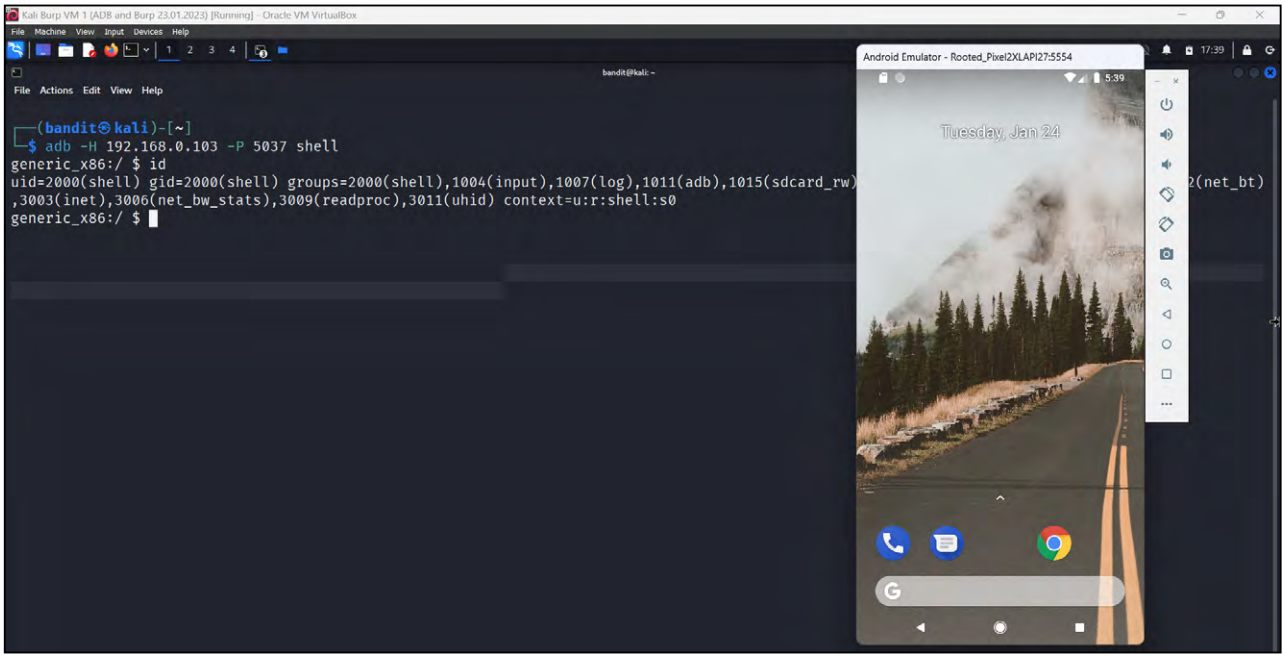
```
C:\Users\VedantWayal>adb -a nodaemon server
adb I 01-24 17:36:33 31328  4452 auth.cpp:417] adb_auth_init...
adb I 01-24 17:36:33 31328  4452 auth.cpp:152] loaded new key from 'C:\Users\VedantWayal\.android\adbkey' with fingerpri
nt 99399119D2022432434D440B7CA5E37D8BE329F942A5A4C441C78E3464FB10CE
adb I 01-24 17:36:33 31328 35088 transport.cpp:332] emulator-5554: read thread spawning
adb I 01-24 17:36:33 31328   892 transport.cpp:304] emulator-5554: write thread spawning
adb I 01-24 17:36:33 31328  4452 adb.cpp:172] emulator-5554: already offline
```

4. Open a different cmd window and type `ipconfig` and note the IP of your Windows host.



5. Start your Kali VM instance and open the terminal.
6. Type the following command:

```
adb -H <windows-ip> -P 5037 shell
```

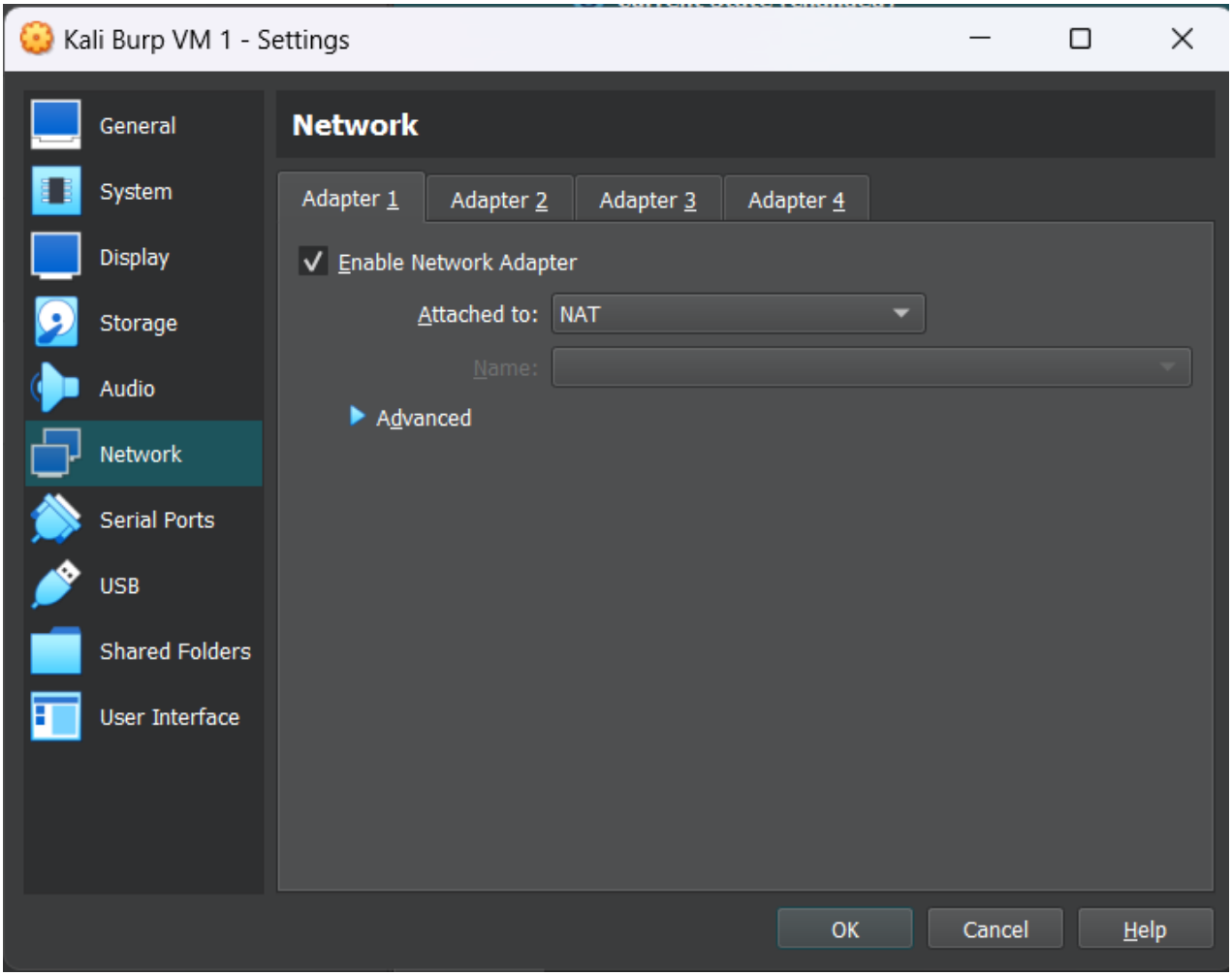


7. In case of multiple devices:

```
adb -H <windows-ip> -P 5037 -s <deviceid> shell
```

8. Your device will be connected. You can try other commands instead of shell.

Note: Set your Kali VBox image's network configuration as **NAT**.



TROUBLESHOOTING:

The emulator gets Killed Error

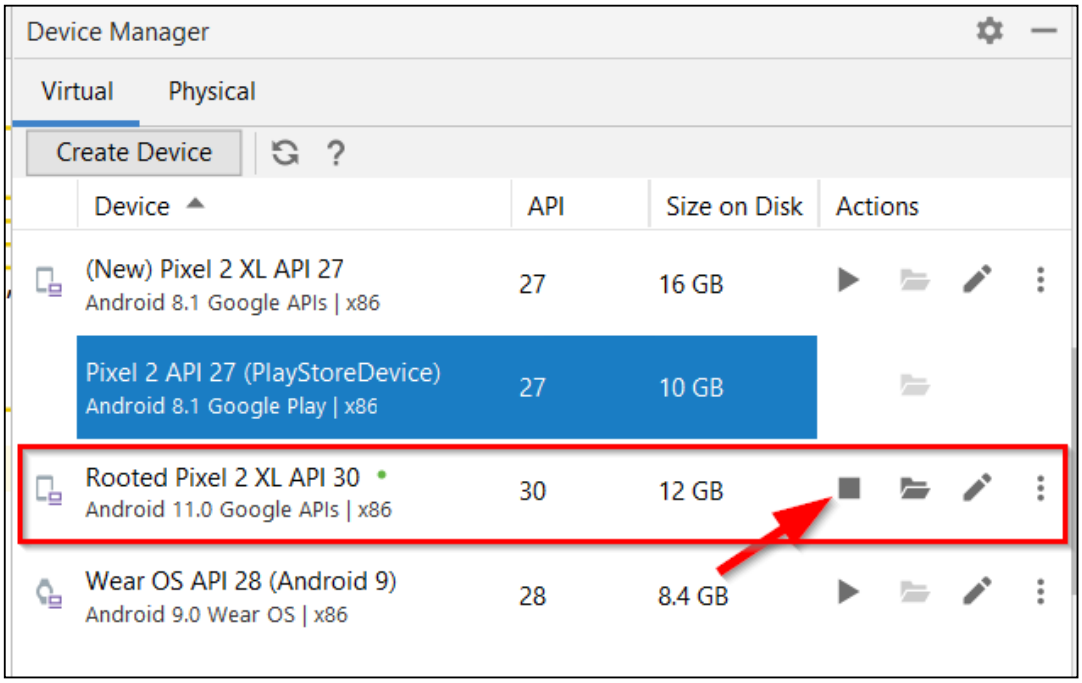
Method 1: Update the Android Emulator to the Latest Release
Method 2: Clear your Disk Space
Method 3: Install the Intel x86 Emulator Accelerator on your system
Method 4: set sdk path to env variable - %localappdata%/android/sdk/emulator/

```
1. Open cmd and run   emulator -list-avds

2. Launch emulator - emulator -avd <emulator_name>
```

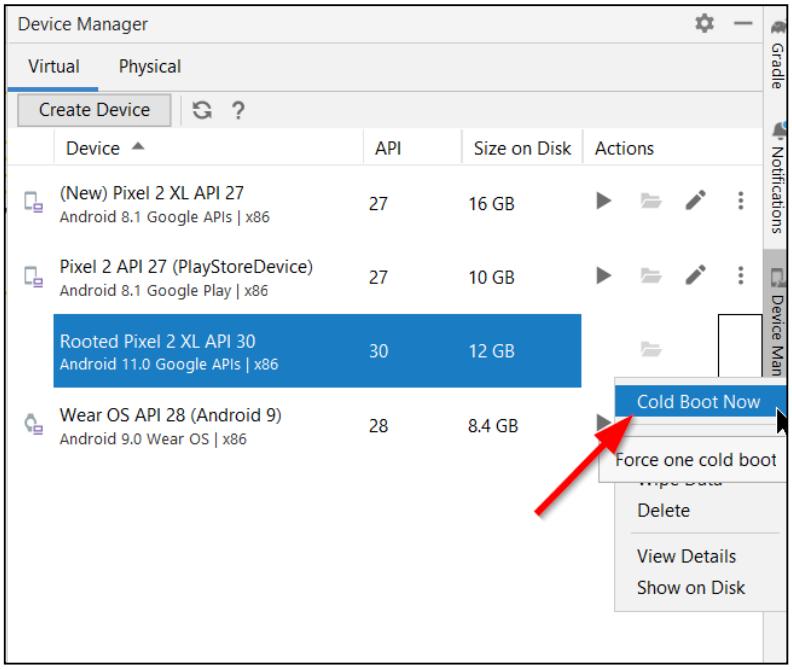
THE EMULATOR IS NOT GETTING TERMINATED

When we stop the device from the device manager, it still does not get terminated. Hence, it becomes impossible to boot the device again without terminating it first. To resolve this, follow the steps below:

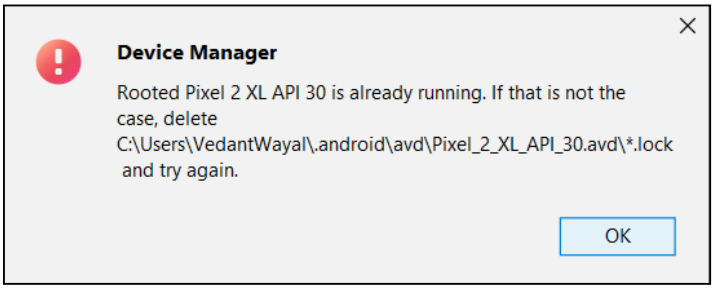
1. Stop the device from the device manager



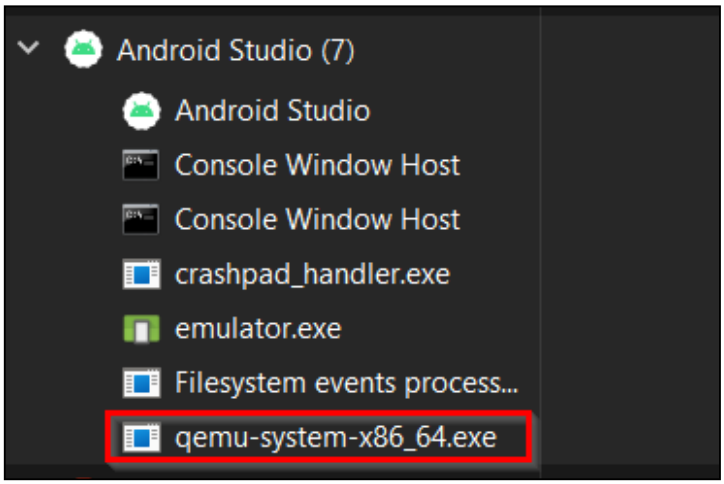
2. Cold reboot the device



3. You will get the following error



4. Open Task Manager and search for `qemu-system` process and end this process



5. You can start the device again.

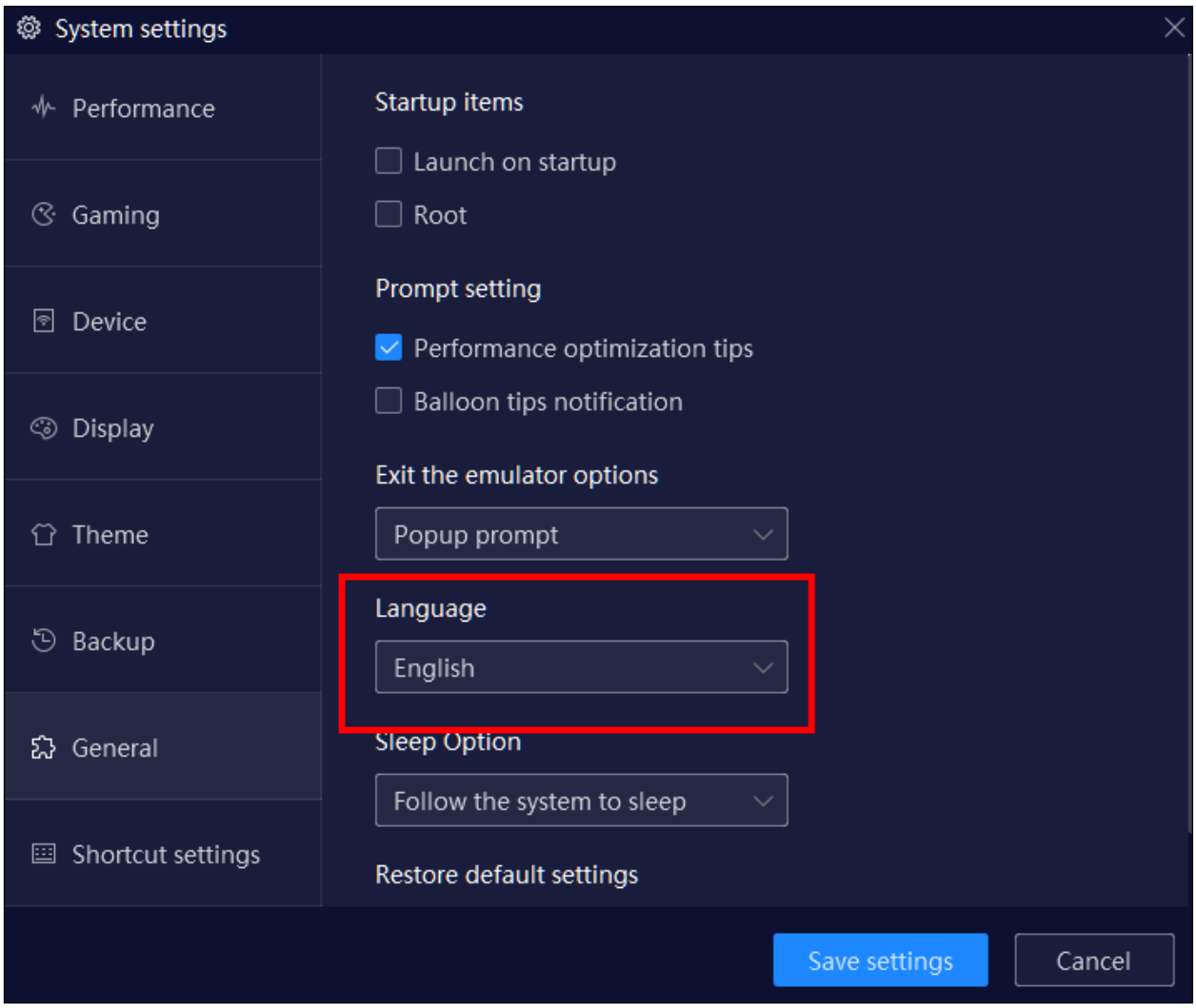
1.5 NoxPlayer Emulator:

DOWNLOAD:

Download NoxPlayer: https://www.bignox.com/

INSTALLATION:

1. Click on the installation file and complete the installation process.
2. Open the installed Noxplayer and click on the Settings icon.
3. Go to General>>Language and change language to English.



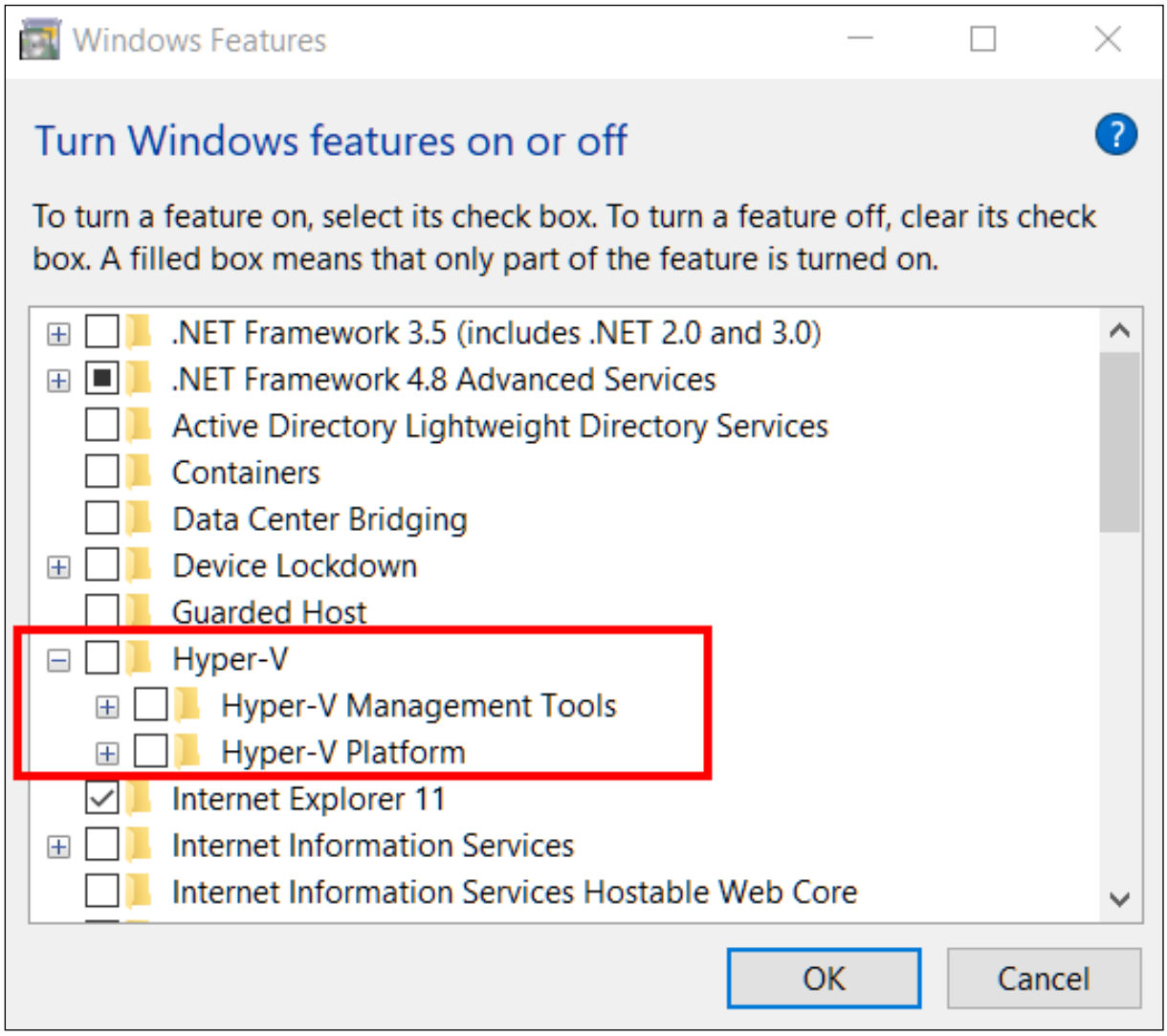
WORKING:

- Once Noxplayer is installed, you can open the player directly from the installation directory or desktop shortcut.
- The emulator will open directly.
- Use drag and drop to install any APK in the emulator.

TROUBLESHOOTING:

>> Noxplayer is stuck at 99% error.

1. Open Control Panel>>Programs>>Turn the Windows feature on or off
2. Uncheck the box in front of Hyper-V



3. Restart system
4. Now Noxplayer will load and start working

>> Disable Hyper-V error:

1. Open Control Panel>>Programs>>Turn Windows feature on or off
2. Uncheck the box in front of Hyper-V
3. Restart system
4. Now Noxplayer will load and start working

- In order to run NoxPlayer, you have to disable Hyper-V, which causes errors for VirtualBox VMs. Thus, enable this feature again to use VirtualBox VMs
- You can use either NoxPlayer or VirtualBox VMs

1.6 Ninjitsu Penetration Testing Environment:

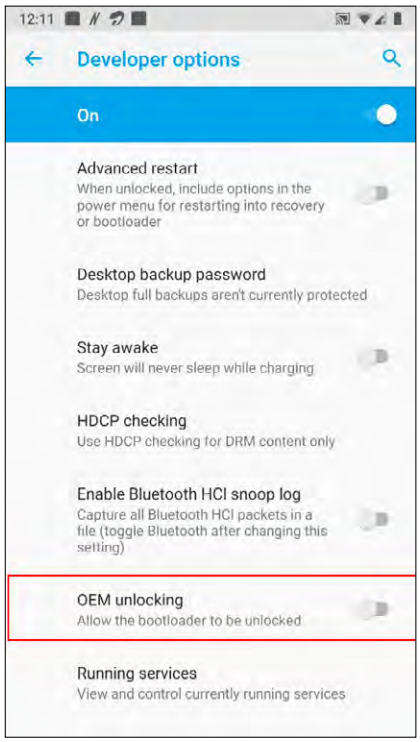
Ninjitsu works alongside the MEmu emulator, and it has a list of pre-installed tools that one would require in an Android application pentest. Check out this detailed article for steps on how to configure it.

1.7 Setting Up the Android Physical Device

ROOTING AN ANDROID DEVICE:

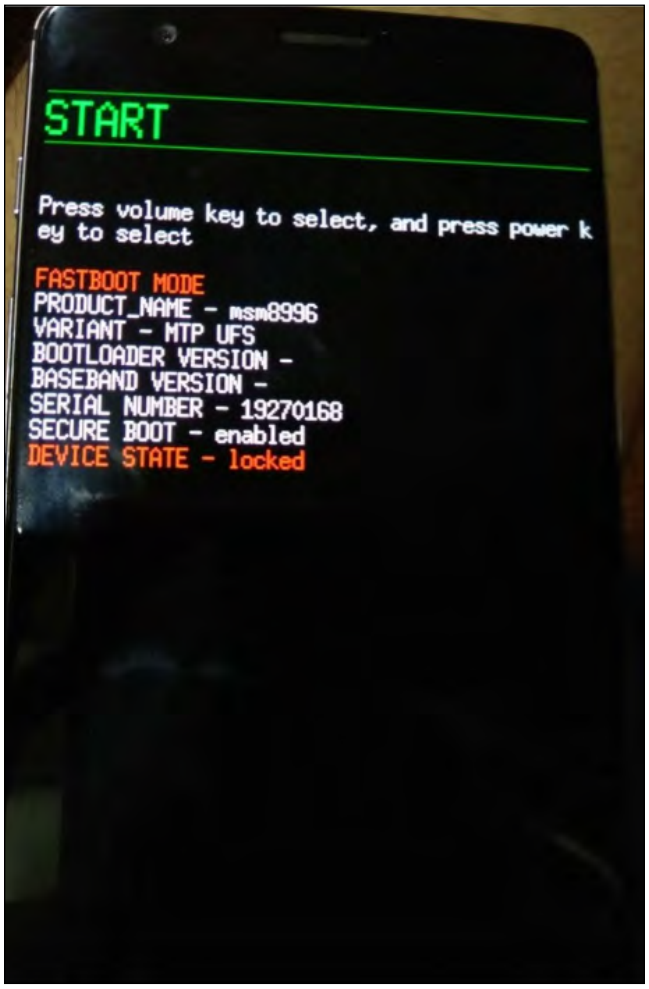
1. Navigate to the About device and tap on the build number 7 times to enable developer options.

2. Go to the developer options and enable “OEM Unlocking”.



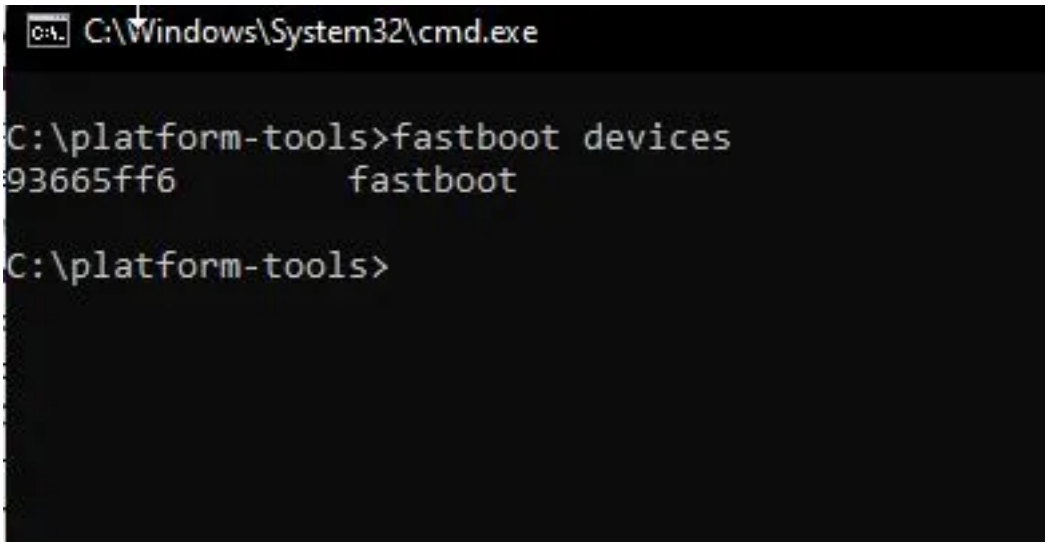
3. After this, reboot the device into the bootloader. You can do this using the adb command.

```
adb reboot bootloader
```



4. Once you are in fast boot mode, check if your system recognizes the device in fast boot mode. Use the following command to check that.

```
fastboot devices
```



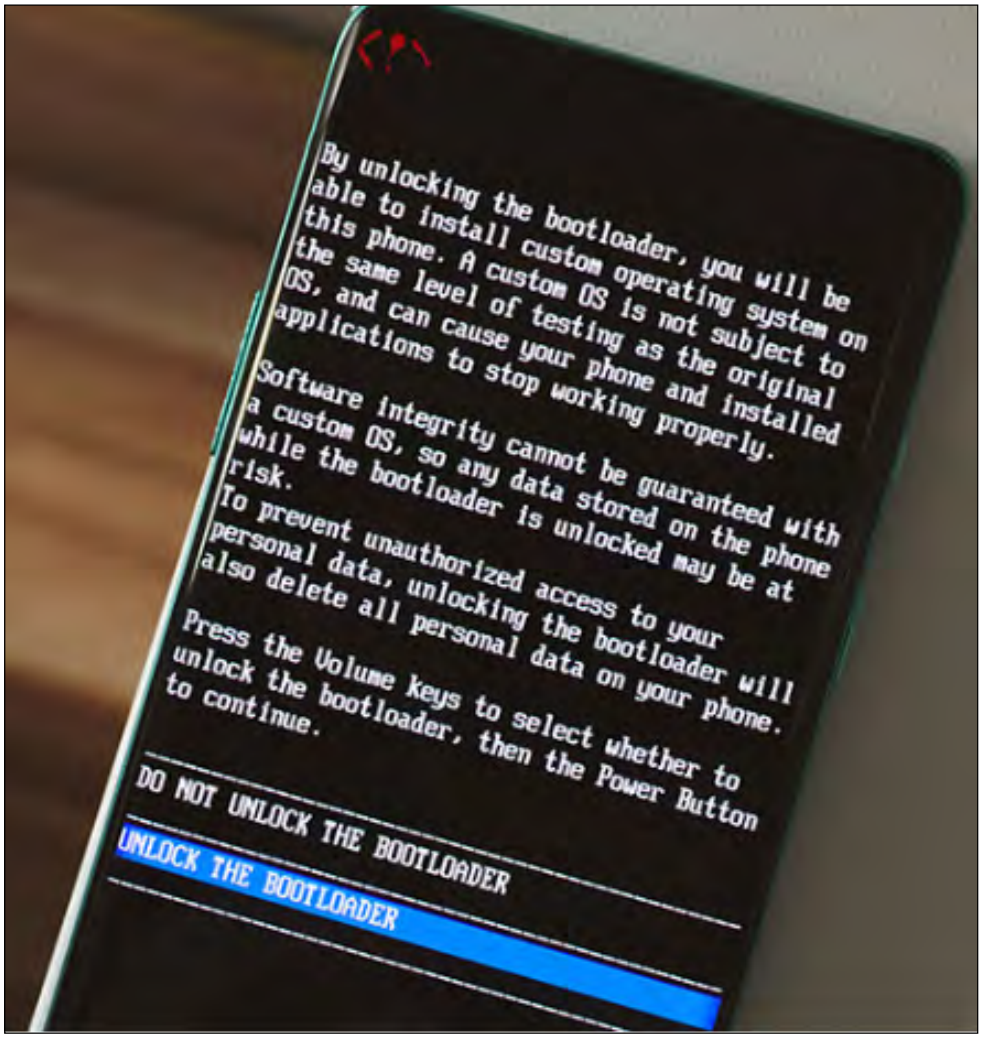
5. If the device is not detected at this stage, go back to your system and ensure that the correct drivers for your respective device have been installed. If not, install the drivers.
   Here are some links for popular drivers.

   Oneplus USB Drivers
   Samsung USB Drivers

6. Once the drivers are installed, resume step 4.
7. Run the following command.

```
fastboot oem unlock
```

8. Once you run the command, you will see the following screen.
   You can move across selections using Vol. Up/Vol. down buttons.
   You can use the power button to select either of the options.
   Please note that unlocking the bootloader will wipe out the entire device, so make sure your data is backed up properly before this step.



9. Once the bootloader is unlocked, the device will boot up in the factory state. At this stage, you will need to set up everything like you would on a new device (Setting up Gmail, Wifi, etc.). Once the device is reconfigured, reboot again into fast boot mode (refer to Step 3).
10. Download TWRP Custom Recovery for the testing device model from here.
11. Flash the custom recovery using this command. Rename the downloaded file to **twrp.img**.

```
fastboot flash recovery twrp.img
fastboot reboot
```

12. Many devices will replace a custom recovery automatically during the first boot. To prevent this, search for the proper key combo for your device to enter recovery. After typing `fastboot reboot` , hold the key combo, and boot to TWRP.
13. You can also try to temporarily boot the downloaded image using the following command:

```
fastboot boot twrp.img
```

    Once TWRP is booted, TWRP will patch the stock ROM to prevent it from replacing TWRP. If you don't follow this step, you will have to repeat the installation.

14. Congratulations, you now have TWRP installed on your device. However, the device is still not rooted.

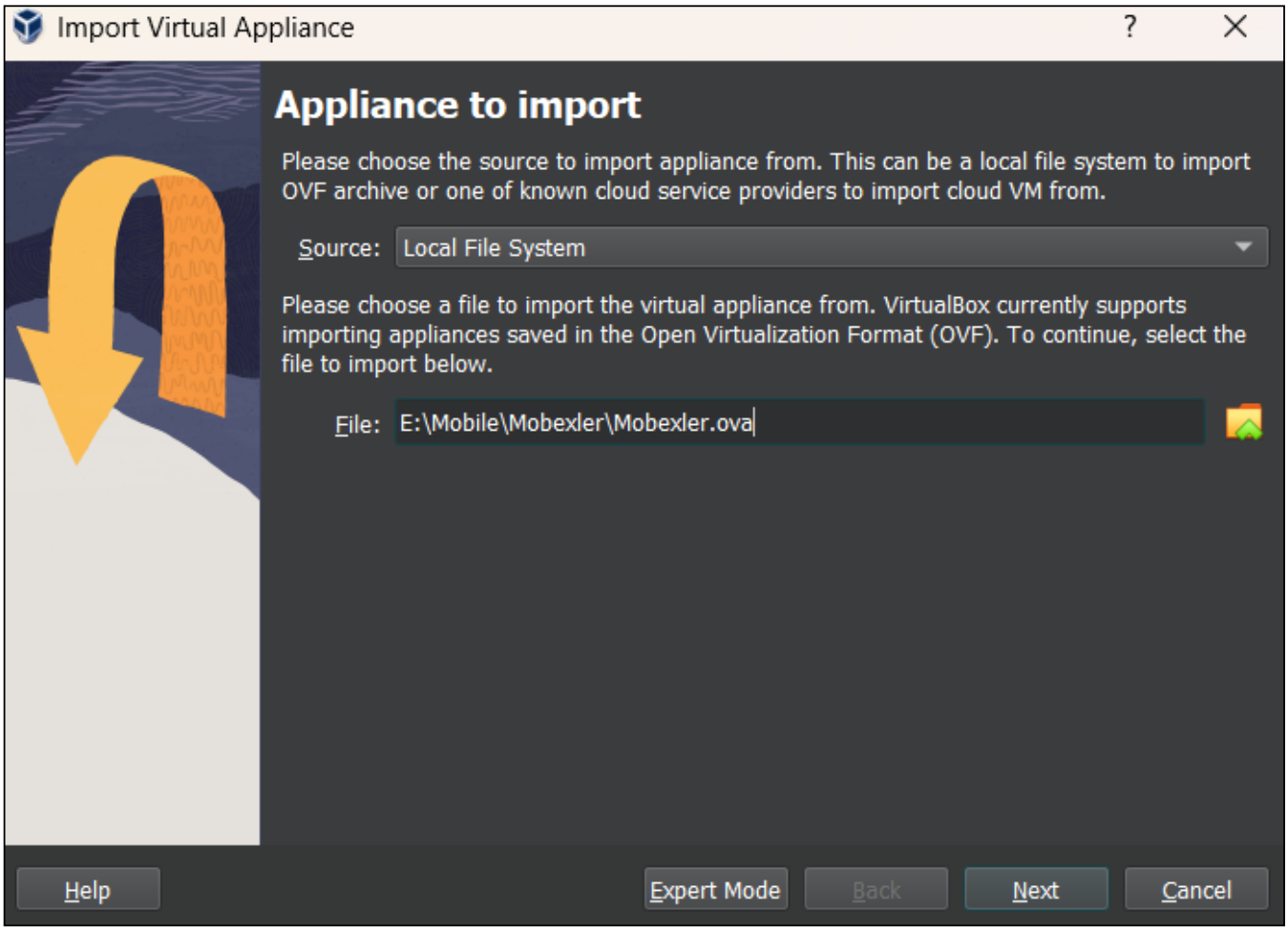
1.8 Mobexler OS Installation

Mobexler is a customized virtual OS, curated to help in the penetration testing of Android & iOS applications.

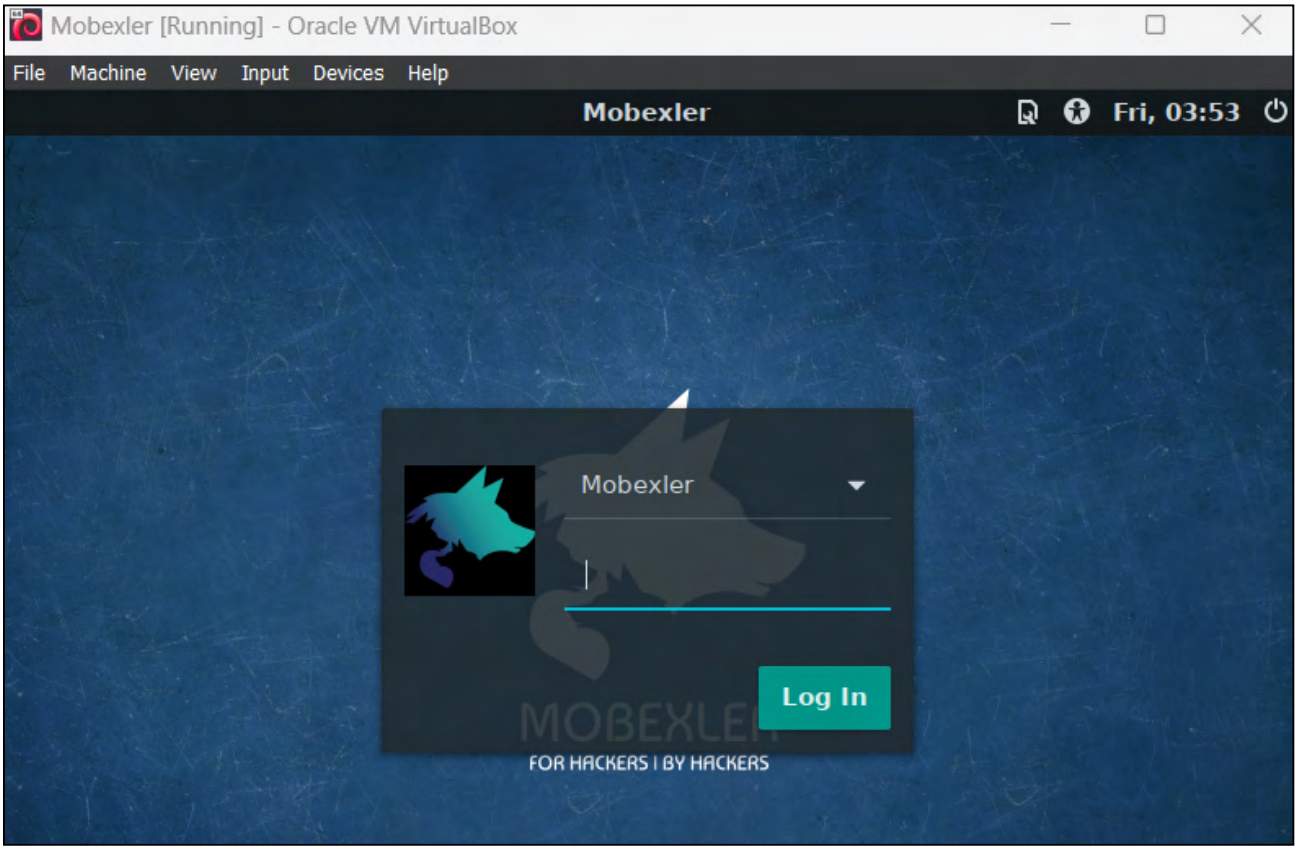
STEPS

1. Download the Mobexler .ova file from the following official website: https://mobexler.com/download.htm

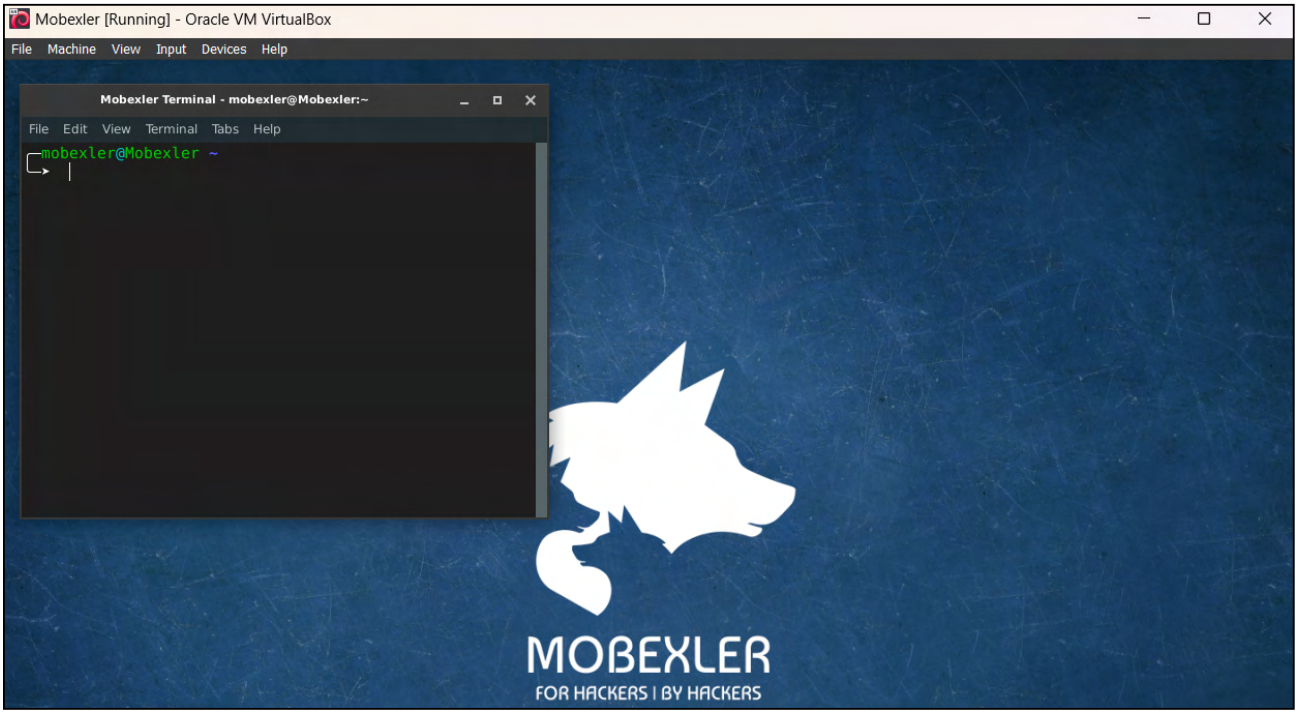
2. Import this downloaded .ova file into Virtualbox

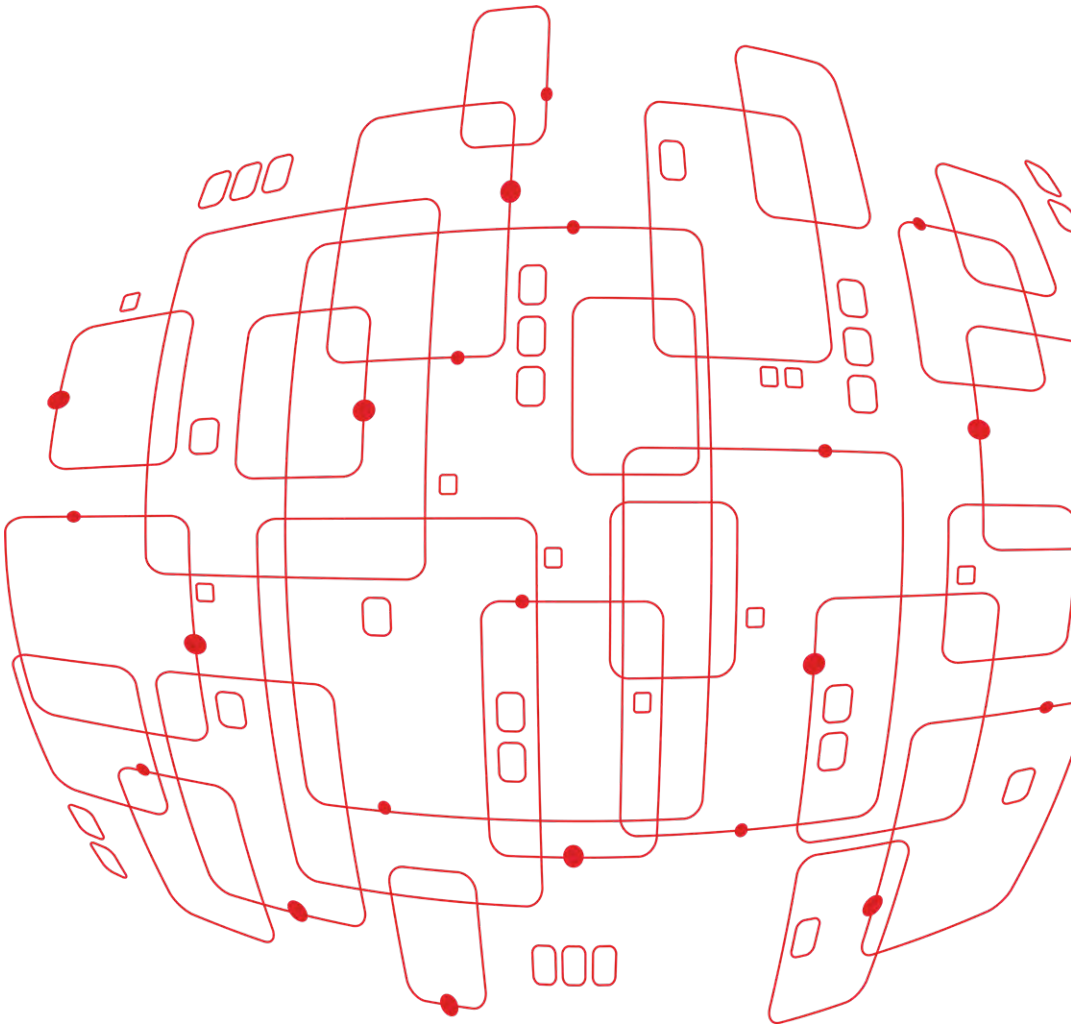


3. Once importing is finished, start this virtual OS.



4. Enter the password: `mobexler` and you can use Mobexler right away.





2. TOOL SETUP

2.1 Magisk Installation

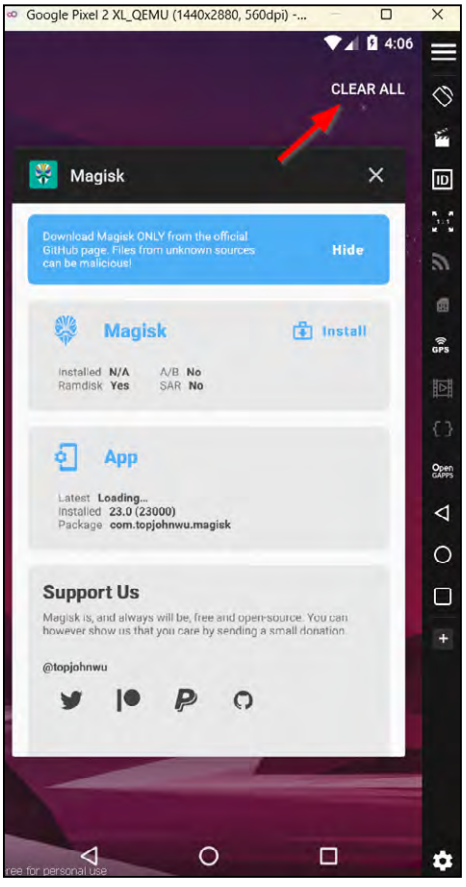
INSTALLING MAGISK IN THE GENYMOTION DEVICE

We recommend using "QEMU Hypervisor" to create/import a new Genymotion device. This setting works best with Magisk. We are using Google Pixel 2XL device with Android version 8.0

STEPS

1. Start the virtual device and download the Magisk.apk from the following link:
   Magisk Manager v23.0

2. Drag and drop the downloaded Magisk apk into the emulator device. The Magisk Manager will open. Just close it from the recent app screen as well.

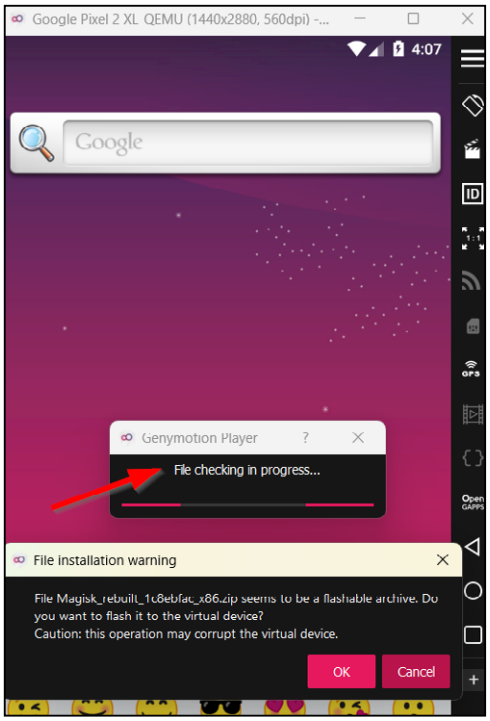


3. Now download the flashable Magisk rebuilt framework from the following based on your device arch type.
   Magisk framework for x86
   Magisk framework for x86_64
   Magisk framework for arm64

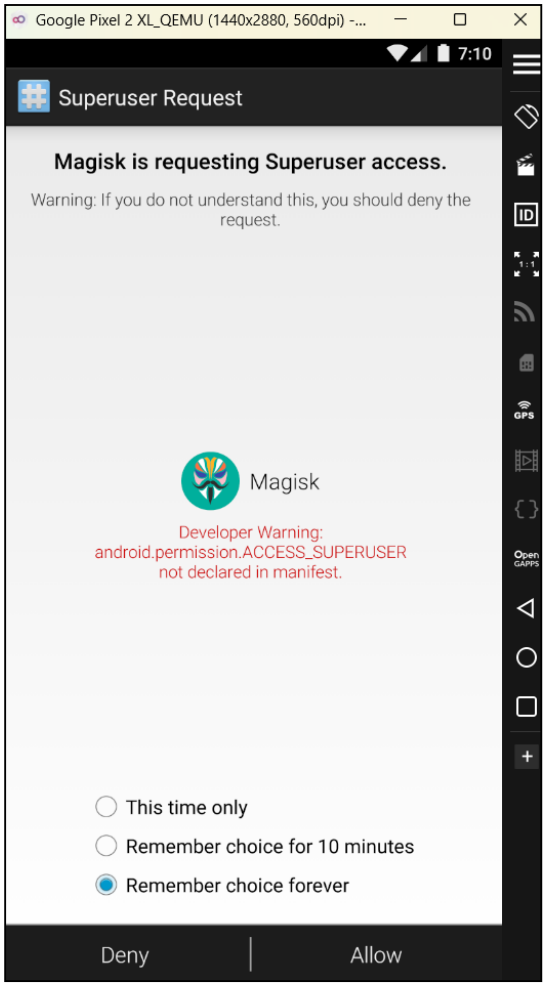
4. Check your device arch type with the following command:

```
adb shell getprop ro.product.cpu.abi
```

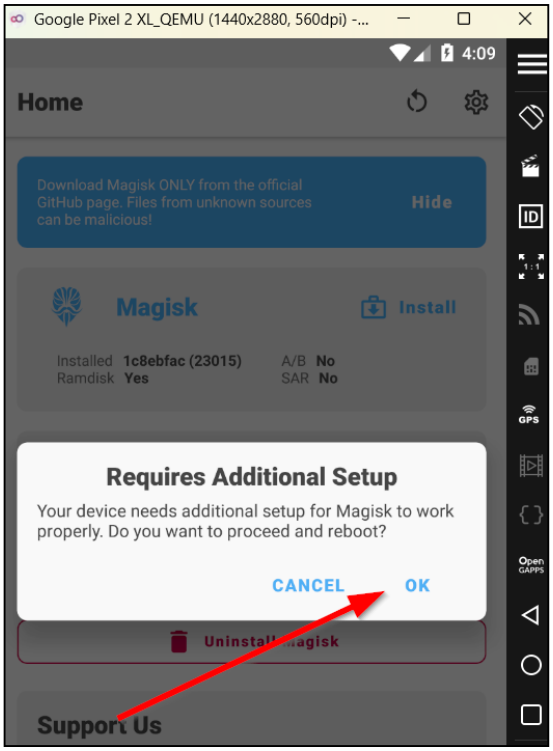
5. Drag and drop the Magisk .zip file into device and reboot the device.



6. Now open the installed Magisk application from the device and grant superuser access (Allow) with the “Remember this choice forever” option.

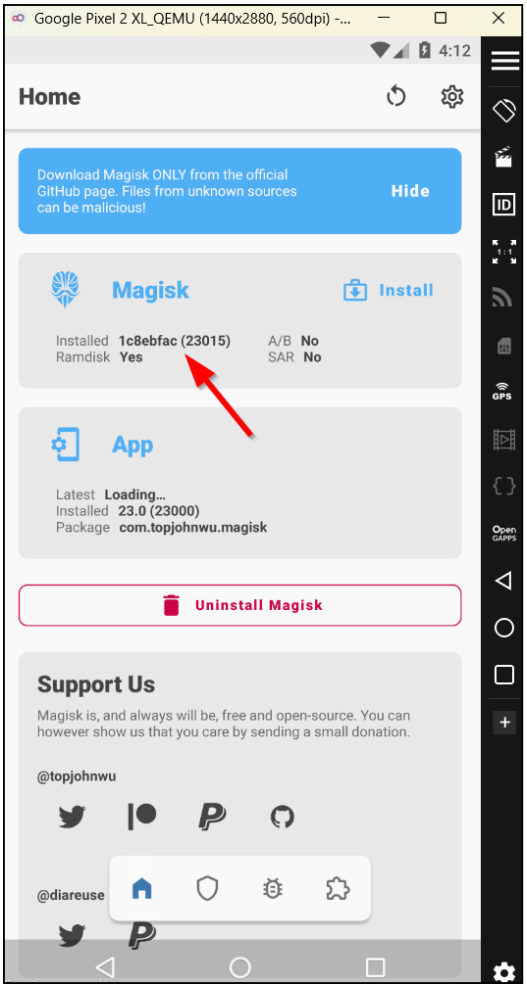


7. Close Magisk Manager and all its processes (from Recent Screen cards) and launch it again.



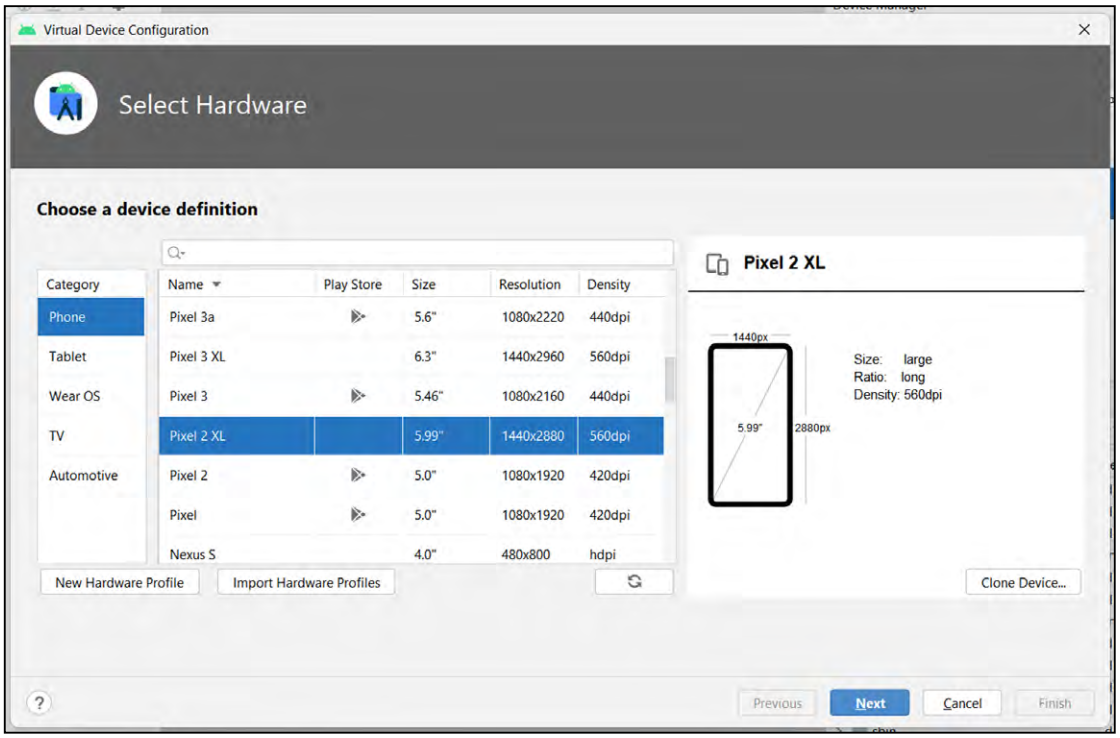
8. A pop-up will appear “Require Additional Setup”. Click on OK, and the device will reboot again.

9. Upon rebooting, open and check for the following value in Magisk Manager. If there is some value instead of N/A, then Magisk has been properly installed on your Genymotion Device.

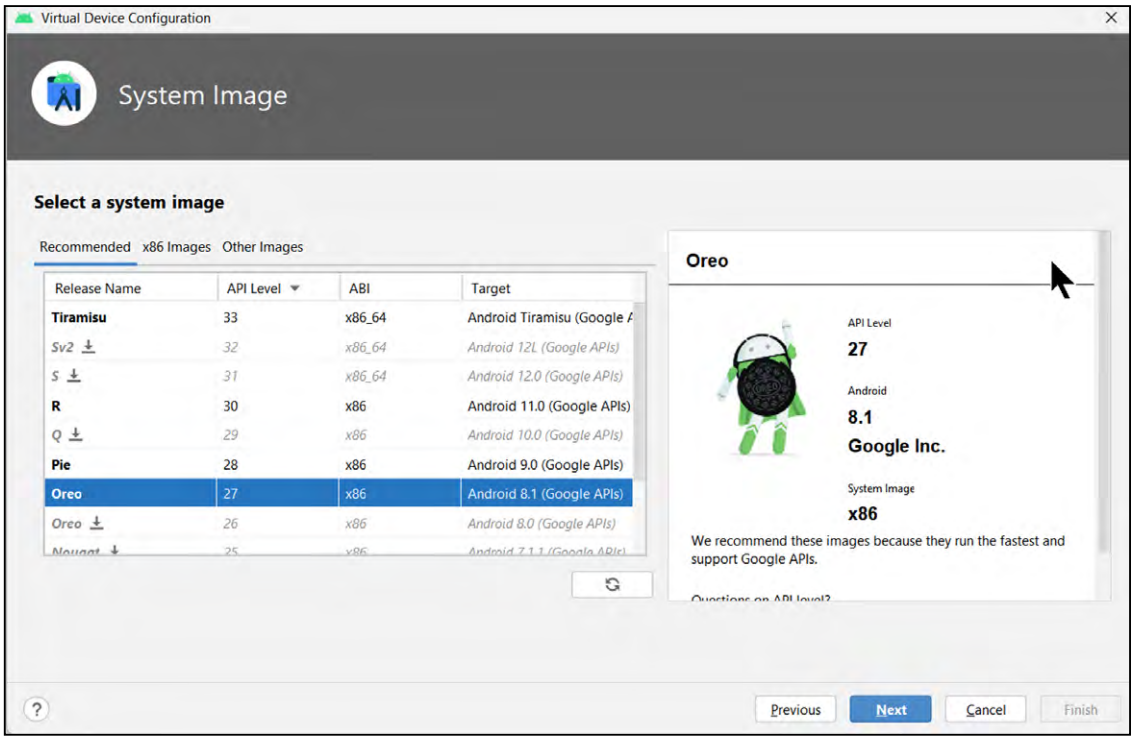


INSTALLING MAGISK IN AVD

1. Create a Pixel 2XL device.



2. Select API Level 27 and create a device.



3. Download the rootAVD repo from below and make sure “/APPS” folder has Magisk.apk
   https://github.com/newbit1/rootAVD

4. Extract the downloaded .zip in the simpler directory to avoid character issues. For example:
   `C:\Users\VedantWayal\Videos\rootAVD-master`

5. Go to `/rootAVD` and open cmd.

6. Type the following command:
   `rootAVD.bat ListAllAVD`

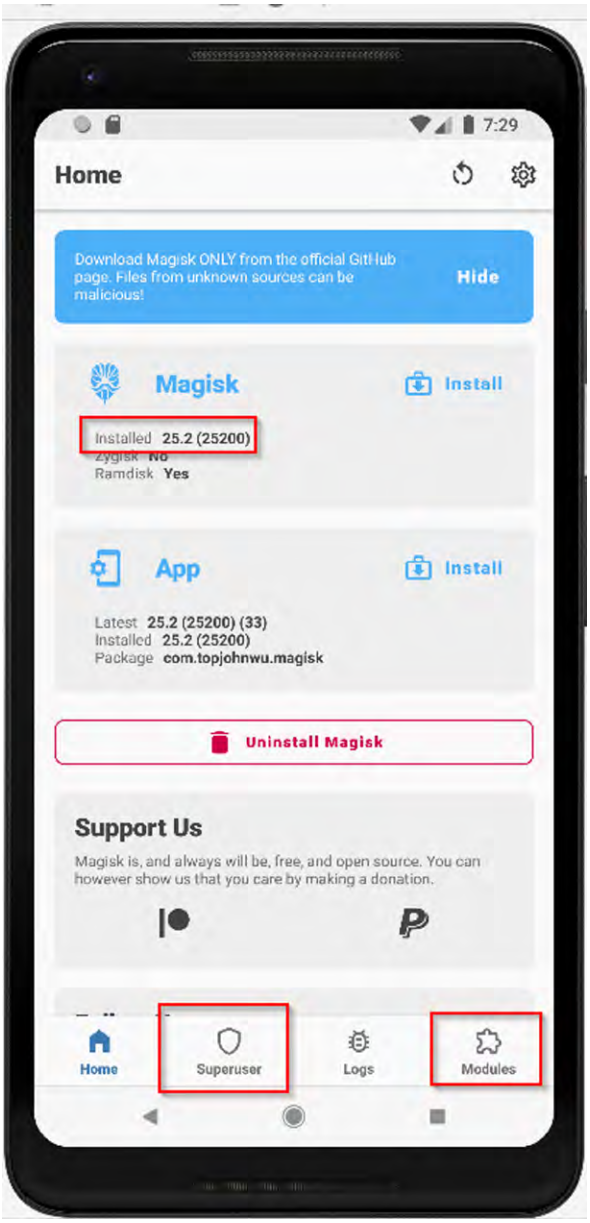
```
Command Examples:
rootAVD.bat
rootAVD.bat ListAllAVDs
rootAVD.bat EnvFixTask
rootAVD.bat InstallApps

rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-30\google_apis\x86\ramdisk.img
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-30\google_apis\x86\ramdisk.img DEBUG PATCHFSTAB GetUSBHPmodZ
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-30\google_apis\x86\ramdisk.img restore
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-30\google_apis\x86\ramdisk.img InstallKernelModules
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-30\google_apis\x86\ramdisk.img InstallPrebuiltKernelModules
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-30\google_apis\x86\ramdisk.img InstallPrebuiltKernelModules GetUSBHPmodZ PATCHFSTAB DEBUG

rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-27\google_apis\x86\ramdisk.img
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-27\google_apis\x86\ramdisk.img DEBUG PATCHFSTAB GetUSBHPmodZ
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-27\google_apis\x86\ramdisk.img restore
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-27\google_apis\x86\ramdisk.img InstallKernelModules
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-27\google_apis\x86\ramdisk.img InstallPrebuiltKernelModules
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-27\google_apis\x86\ramdisk.img InstallPrebuiltKernelModules GetUSBHPmodZ PATCHFSTAB DEBUG
```

7. Now run this command highlighted below:

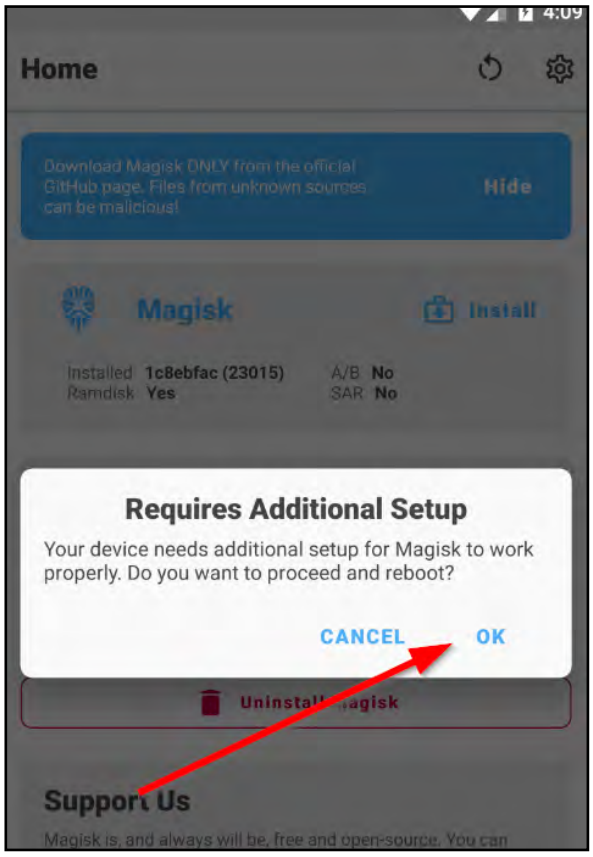
```
Command Examples:
rootAVD.bat
rootAVD.bat ListAllAVDs
rootAVD.bat EnvFixTask
rootAVD.bat InstallApps

rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-30\google_apis\x86\ramdisk.img
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-30\google_apis\x86\ramdisk.img DEBUG PATCHFSTAB GetUSBHPmodZ
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-30\google_apis\x86\ramdisk.img restore
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-30\google_apis\x86\ramdisk.img InstallKernelModules
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-30\google_apis\x86\ramdisk.img InstallPrebuiltKernelModules
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-30\google_apis\x86\ramdisk.img InstallPrebuiltKernelModules GetUSBHPmodZ PATCHFSTAB DEBUG

rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-27\google_apis\x86\ramdisk.img
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-27\google_apis\x86\ramdisk.img DEBUG PATCHFSTAB GetUSBHPmodZ
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-27\google_apis\x86\ramdisk.img restore
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-27\google_apis\x86\ramdisk.img InstallKernelModules
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-27\google_apis\x86\ramdisk.img InstallPrebuiltKernelModules
rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-27\google_apis\x86\ramdisk.img InstallPrebuiltKernelModules GetUSBHPmodZ PATCHFSTAB DEBUG
```

```
C:\Windows\System32\cmd.e

C:\Users\VedantWayal\Videos\rootAVD-master>rootAVD.bat %LOCALAPPDATA%\Android\Sdk\system-images\android-30\google_apis\x86\ramdisk.img
[*] Set Directorys
[-] Test if ADB SHELL is working
[-] ADB connectoin possible
[-] In any AVD via ADB, you can execute code without root in /data/data/com.android.shell
[*] looking for Magisk installer Zip
[*] Cleaning up the ADB working space
[*] Creating the ADB working space
[*] Push Magisk.zip into /data/data/com.android.shell/Magisk
[-] C:\Users\VedantWayal\Videos\rootAVD-master\Magisk.zip: 1 file pushed, 0 skipped. 105.5 MB/s (11278270 bytes in 0.102s)
[-] Backup exists already
[*] Push ramdisk.img into /data/data/com.android.shell/Magisk
[-] C:\Users\VedantWayal\AppData\Local\Android\Sdk\system-images\android-30\google_apis\x86\ramdisk.img: 1 file pushed, 0 skipped. 107.0 MB/s (1790530 bytes
 in 0.016s)
[-] Copy rootAVD Script into Magisk DIR
rootAVD.sh: 1 file pushed, 0 skipped. 139.0 MB/s (7855800 bytes in 0.054s)
[-] run the actually Boot/Ramdisk/Kernel Image Patch Script
[*] from Magisk by topjohnwu and modded by NewBit XDA
[!] We are in a ranchu emulator shell
[-] Api Level Arch Detect
[-] Device Platform is x86 only
[-] Device SDK API: 27
[-] First API Level:
[-] The AVD runs on Android 8.1.0
[-] Switch to the location of the script file
[*] Extracting busybox from script ...
[!] There is no busybox behind the script
[*] Trying to Download the Up-To-Date Script Version
[!] Temporarily installing Magisk
[!] Removing Temporarily installed Magisk
[-] Checking AVDs Internet connection...
[!] AVD is offline
[*] Extracting Magisk.zip ...
[*] Re-Run rootAVD in Magisk Busybox STANDALONE (D)ASH
[-] We are now in Magisk Busybox STANDALONE (D)ASH
[*] rootAVD with Magisk  Installer
[-] Get Flags
[-] Encrypted data, keep forceencrypt
```

8. Reboot the device and your device will be rooted along with Magisk installed.



9. Do additional settings if prompted.

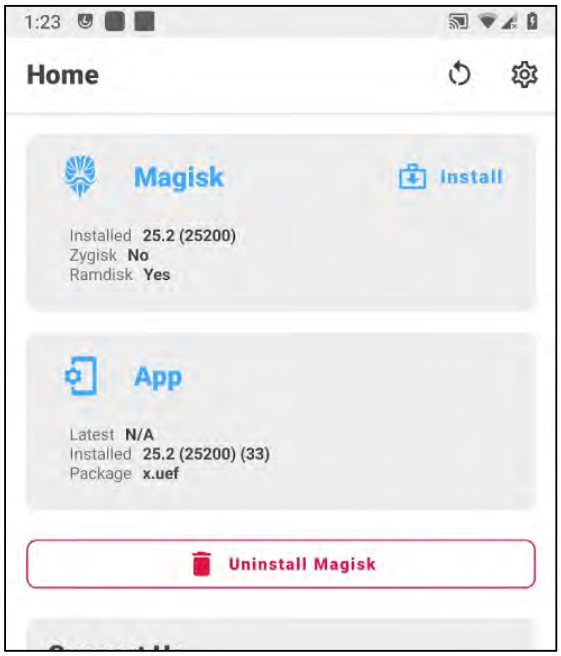


10. Also, note that all other devices which are running on API 27 will also get rooted.

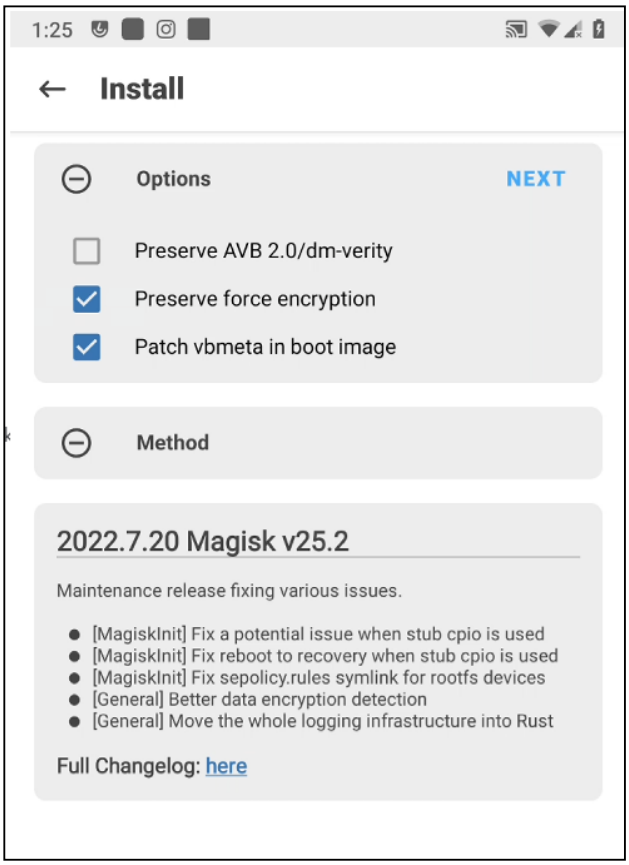
INSTALLING MAGISK ON PHYSICAL DEVICE

1. Download the latest Magisk APK from this link. Once the APK has been downloaded, you can install it from the device.

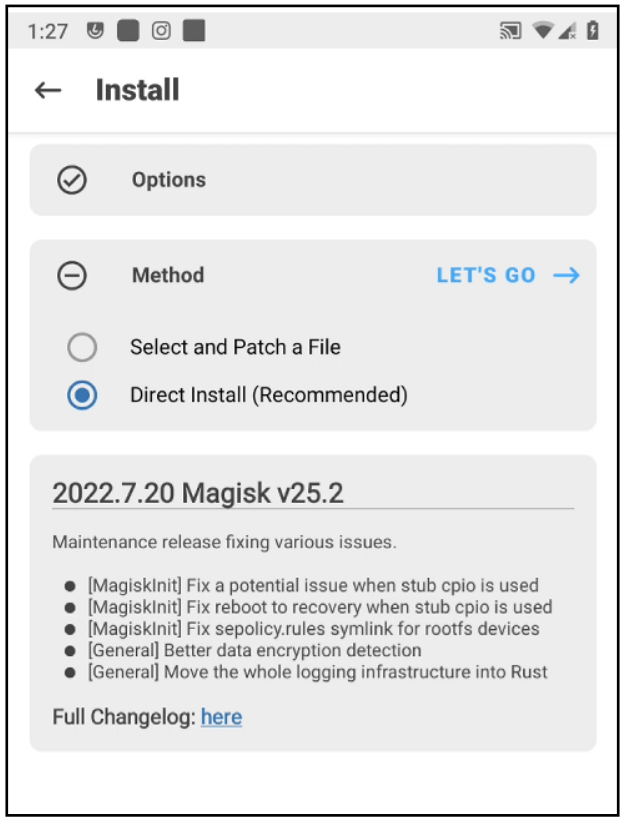
2. Open the Magisk application and you should see the following screen. Click on install.



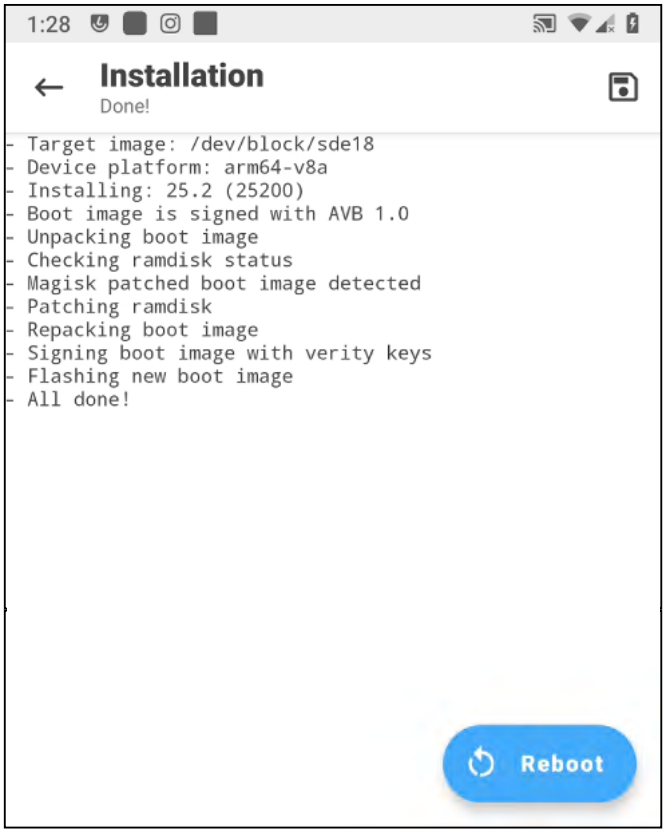
3. Clicking on install should bring you to this page. After this, click on Next.



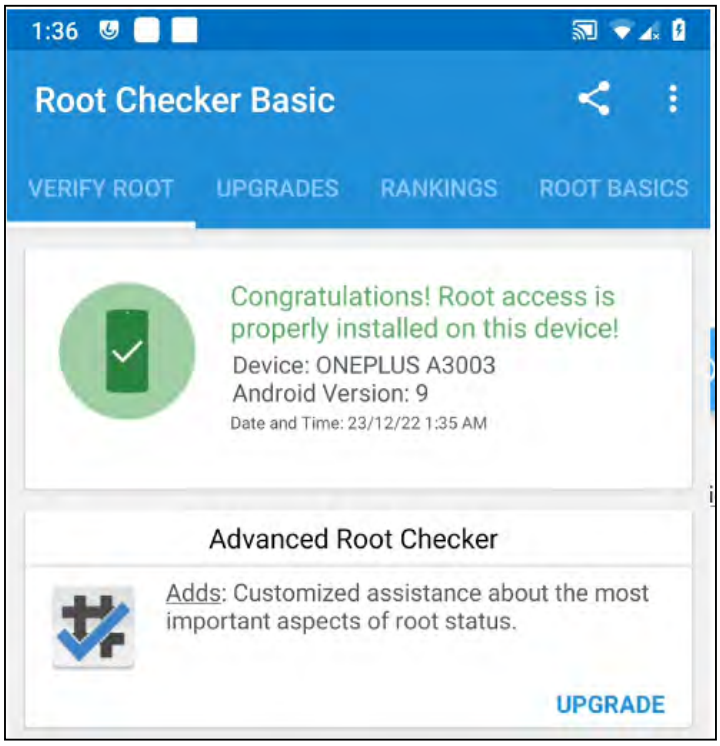
4. Select Direct Install and click on Let's Go.



5. Once the installation is complete, reboot the device. The device is now rooted.



6. You can check the root status of your device by downloading this application.



DROZER VIA MOBEXLER

Drozer installation is a challenging and time-consuming task. Mobexler comes with preinstalled drozer, and can be used instead of Drozer. We will be using the drozer from Mobexler VM.

Pre-requisites:

- Running Mobxler VM
- Genymotion device running

Installation:

1. Download *drozer-agent.apk* from here https://github.com/mwrlabs/drozer/releases/download/2.3.4/drozer-agent-2.3.4.apk
2. Start Mobexler VM
3. Connect your Genymotion device to Mobelxer

```
adb connect <your device ip>
```

4. Open the directory where *drozer-agent.apk* is downloaded and install *drozer-agent.apk* into the Genymotion device

```
adb install drozer-agent.apk
```

Usage:

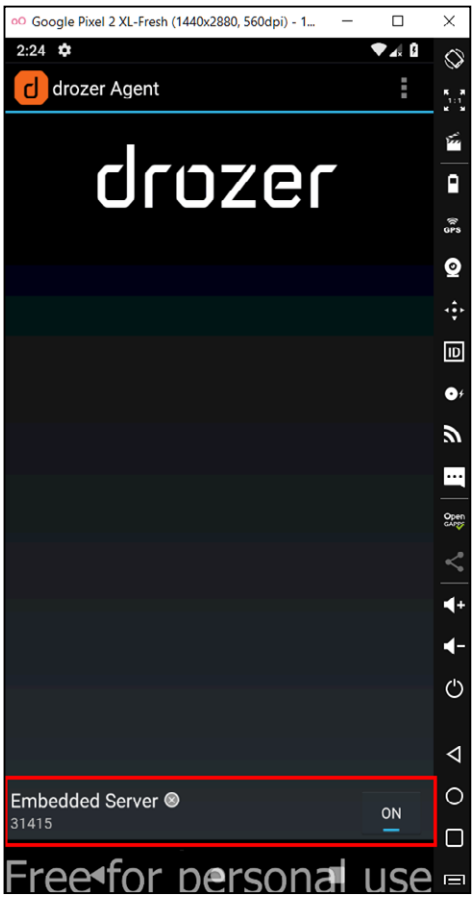
1. Start the server from Mobexler
    a. The agent is running on port *31415*, and we need to port forward to establish the communication between the Drozer Client and Agent. Run this command to do the

```
adb forward tcp:31415 tcp:31415
```

2. Connect your Genymotion device to Mobelxer

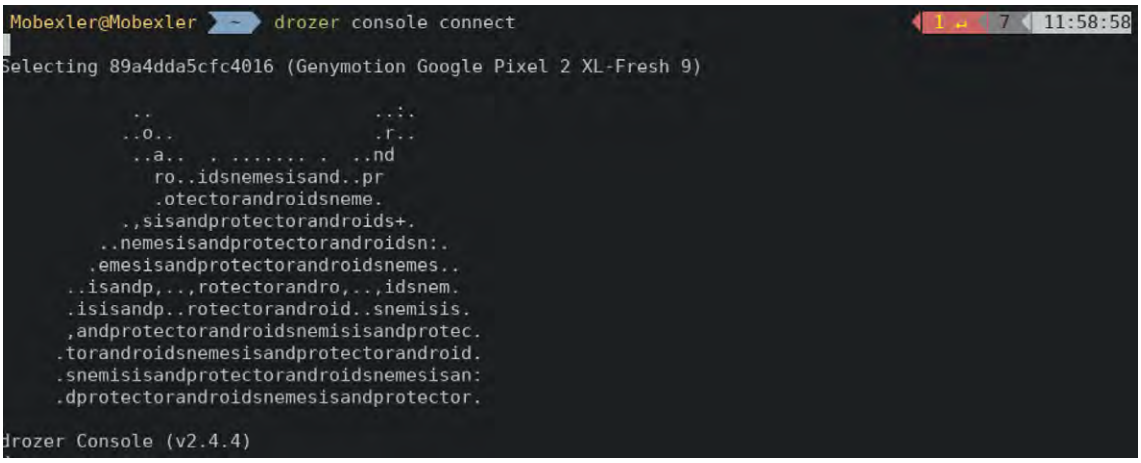
```
adb connect <your device ip>
```

3. Open *drozer-agent.apk* from the device and Start the server



4. Connect to the drozer server from Mobexler

```
drozer console connect
```



5. Type `env` command to check if the server is connected

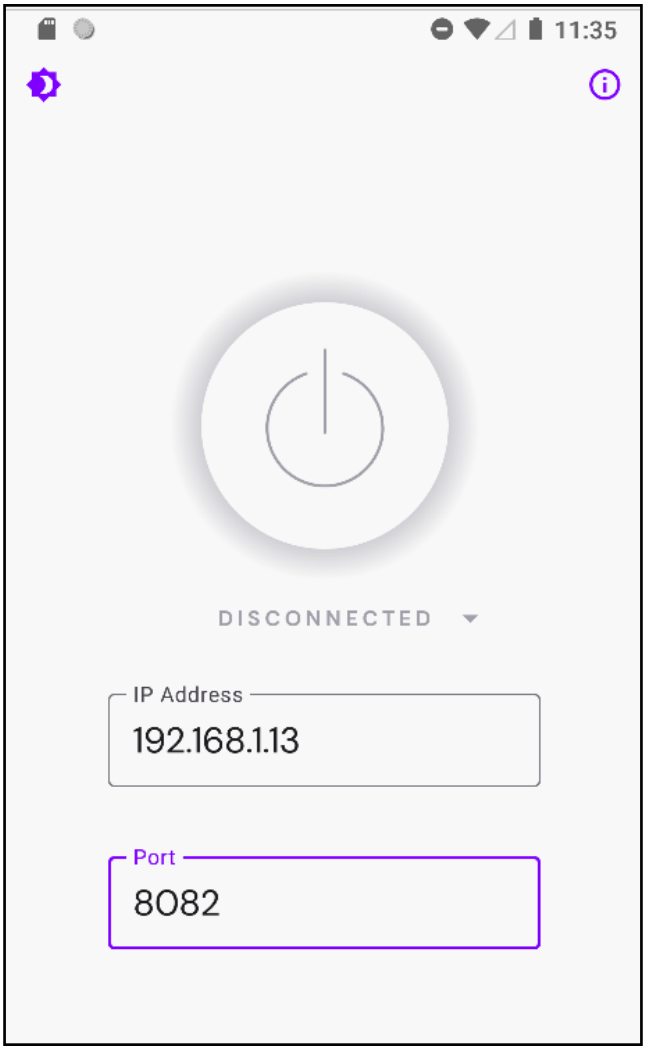
```
dz> env
PATH=/data/user/0/com.mwr.dz/bin:/sbin:/vendor/bin:/system/sbin:/syst
em/bin:/system/xbin
WD=/data/user/0/com.mwr.dz
```

ONE CLICK PROXY

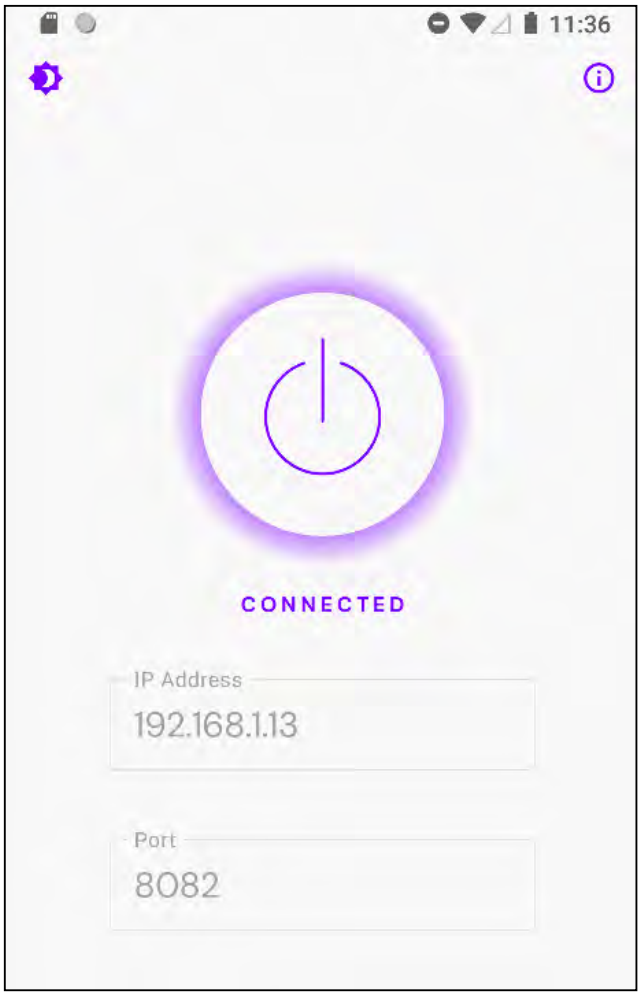
Proxy Toggle tool lets us quickly enable/disable proxy settings in our Android devices.

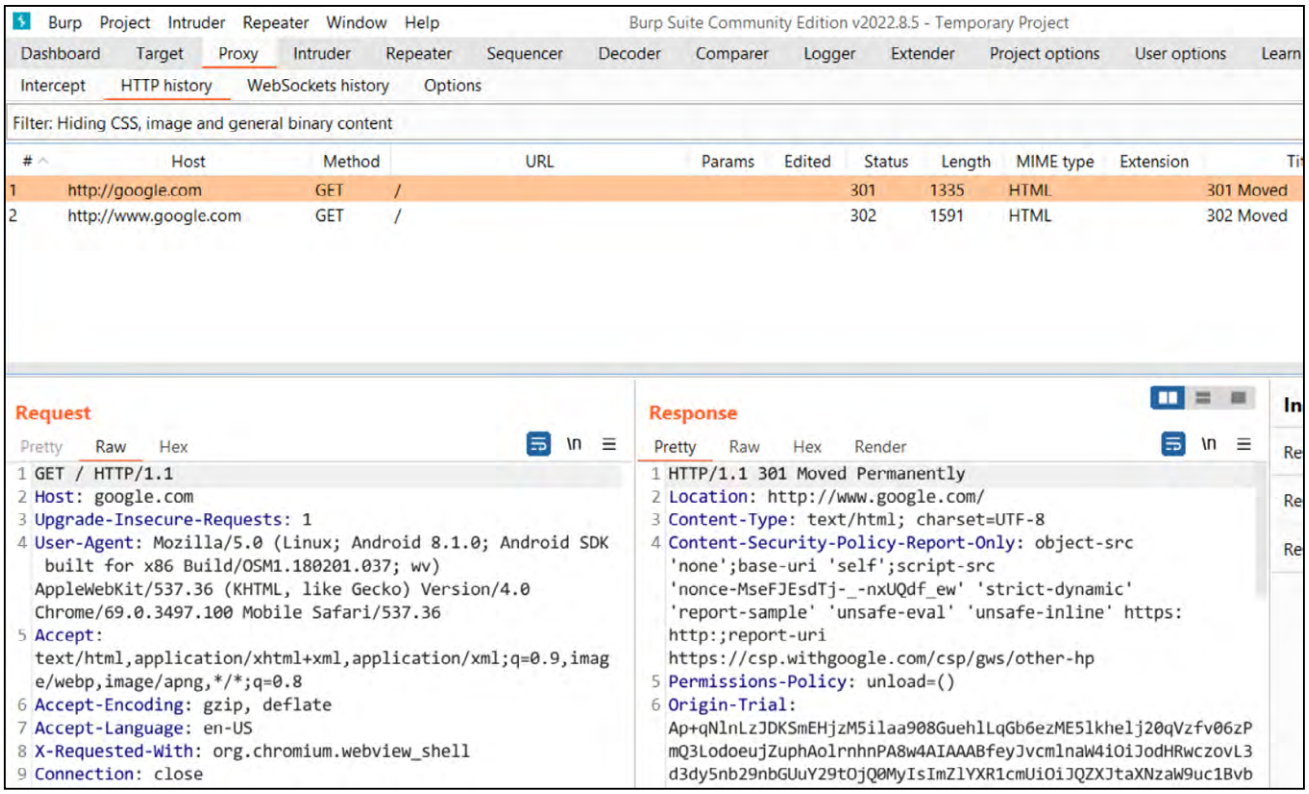
STEPS:

1. Download Proxy Toggle zip from following repository: https://github.com/theappbusiness/android-proxy-toggle
2. Extract the zip and install the `proxy-toggle.apk` into the Android device.
3. Now simply put the proxy IP address and port number and click on the Start button



4. The proxy will be started, and all traffic will be sent via the provided combination of IP address and Port.


11:36
CONNECTED
IP Address
192.168.1.13
Port
8082

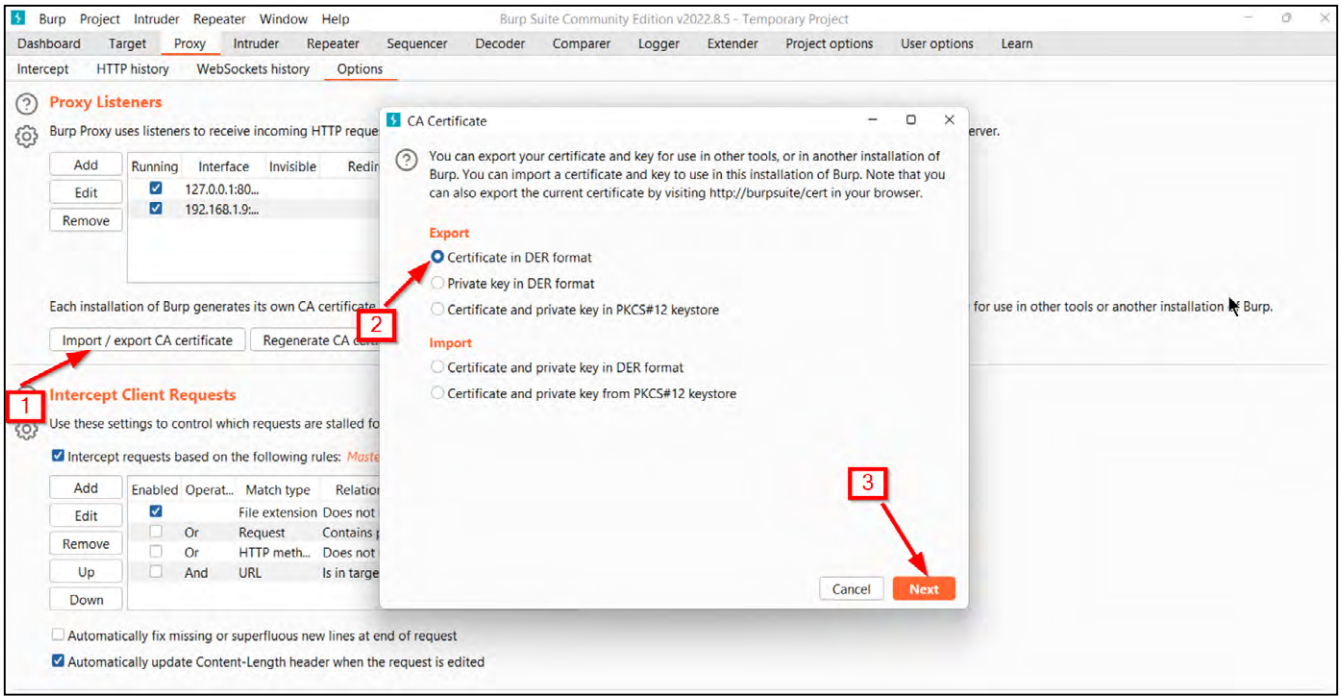

Burp Project Intruder Repeater Window Help
Burp Suite Community Edition v2022.8.5 - Temporary Project
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn
Intercept HTTP history WebSockets history Options
Filter: Hiding CSS, image and general binary content
Host Method URL Params Edited Status Length MIME type Extension
1 http://google.com GET / 301 1335 HTML 301 Moved
2 http://www.google.com GET / 302 1591 HTML 302 Moved
Request
Pretty Raw Hex
1 GET / HTTP/1.1
2 Host: google.com
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Linux; Android 8.1.0; Android SDK built for x86 Build/OSM1.180201.037; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/69.0.3497.100 Mobile Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
6 Accept-Encoding: gzip, deflate
7 Accept-Language: en-US
8 X-Requested-With: org.chromium.webview_shell
9 Connection: close
Response
Pretty Raw Hex Render
1 HTTP/1.1 301 Moved Permanently
2 Location: http://www.google.com/
3 Content-Type: text/html; charset=UTF-8
4 Content-Security-Policy-Report-Only: object-src 'none';base-uri 'self';script-src 'nonce-MseFJEsdTj-_-nxUQdf_ew' 'strict-dynamic' 'report-sample' 'unsafe-eval' 'unsafe-inline' https: http:;report-uri https://csp.withgoogle.com/csp/gws/other-hp
5 Permissions-Policy: unload=()
6 Origin-Trial: Ap+qNlnLzJDKSmEHjzM5ilaa908GuehlLqGb6ezME5lkhelj20qVzfv06zPmQ3LodoeujZuphAolrnhnPA8w4AIAAABfeyJvcmlnaW4iOiJodHRwczovL3d3dy5nb29nbGUuY29tOjQ0MyIsImZlYXR1cmUiOiJQZXJtaXNzaW9uc1Bvb

SETTING UP BURPSUITE WITH ANDROID DEVICE

METHOD 1

We can directly import the certificate and convert it to .crt format.

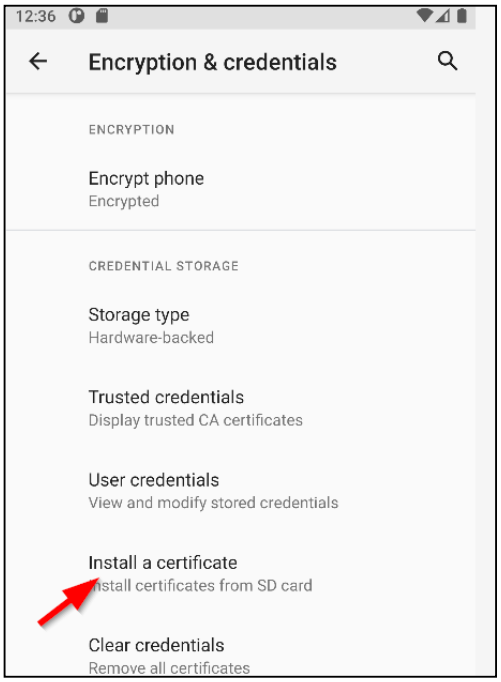
1. Import certificate in .der format and change extension as .crt



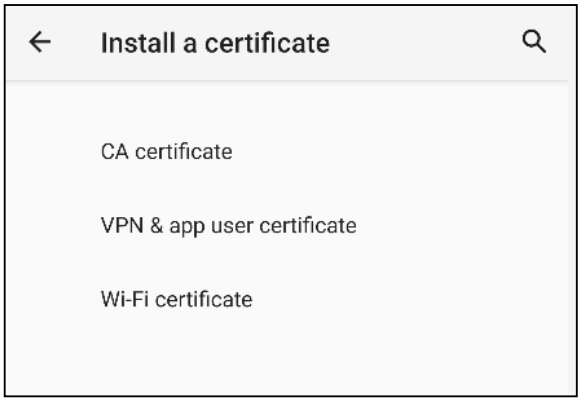
2. Push this certificate into /sdcard folder of the device with adb

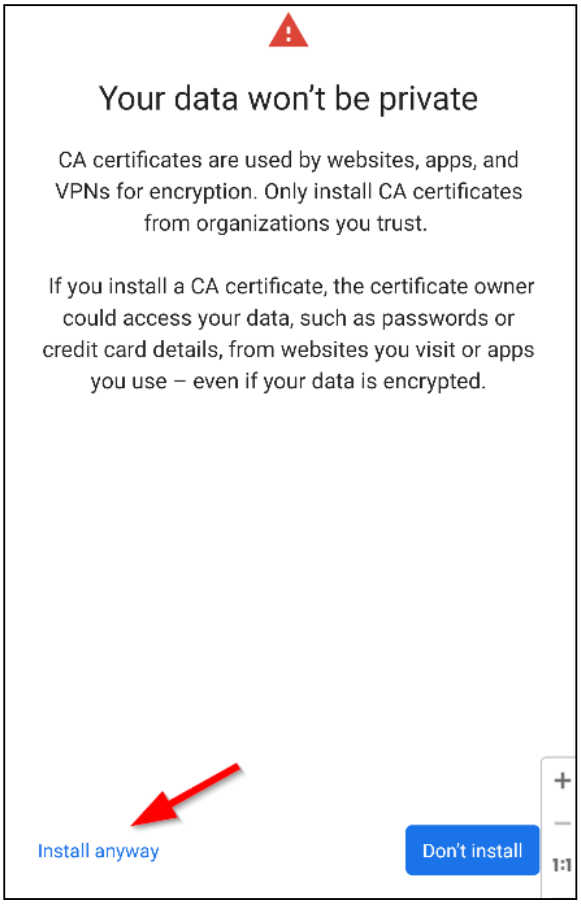
```
adb push burpcert.crt /sdcard
```

3. Now go to
   Settings>>Security>>Encryption & Credentials"
   and click on "Install a certificate"

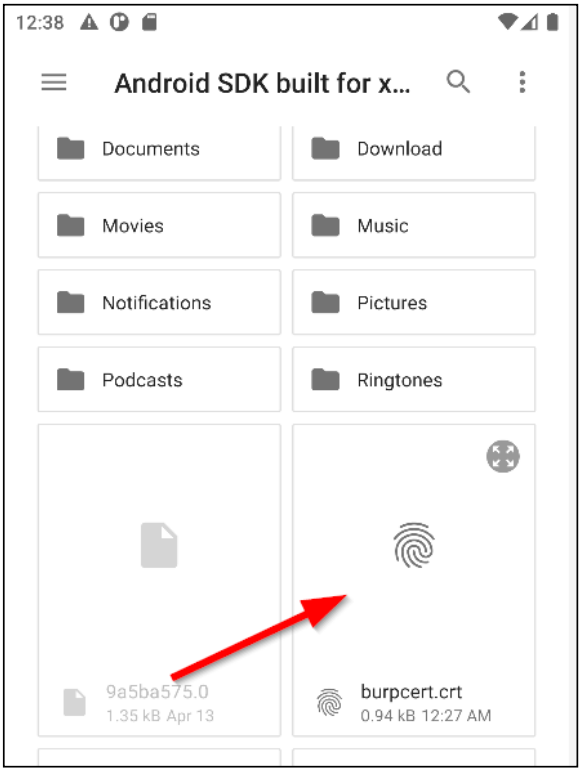


4. Select CA certificate>>Install Anyway

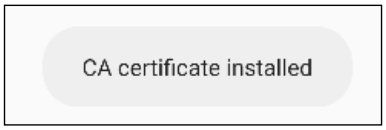




5. Select the Burp Certificate from the sd card



6. The CA certificate will be installed



7. However, we cannot intercept the SSL requests yet. From Android 7 and upwards, Android uses 2 different Trust Stores, the user trust store and the system trust store. We will have to inject our CA cert into the System Trust Store to solve this. The easiest way is to use Magisk and a module.

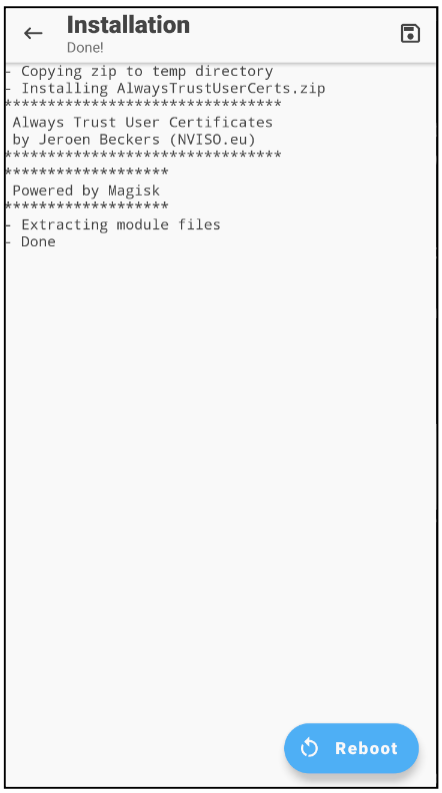
8. Root the Android device as mentioned in the other part of this document and install Magisk Manager.

9. Now download the AlwaysTrustUserCerts.zip file which we will install via the Magisk manager
https://github.com/NVISOsecurity/MagiskTrustUserCerts

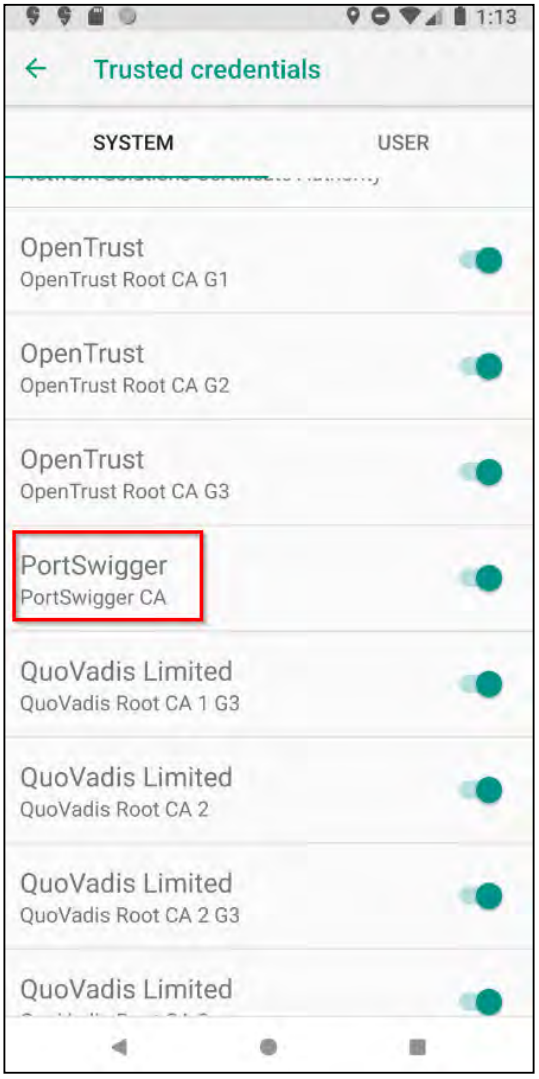
10. Now drag and drop this zip into the device.

11. Open Magisk Manager on a Rooted Android device and go to the Modules tab

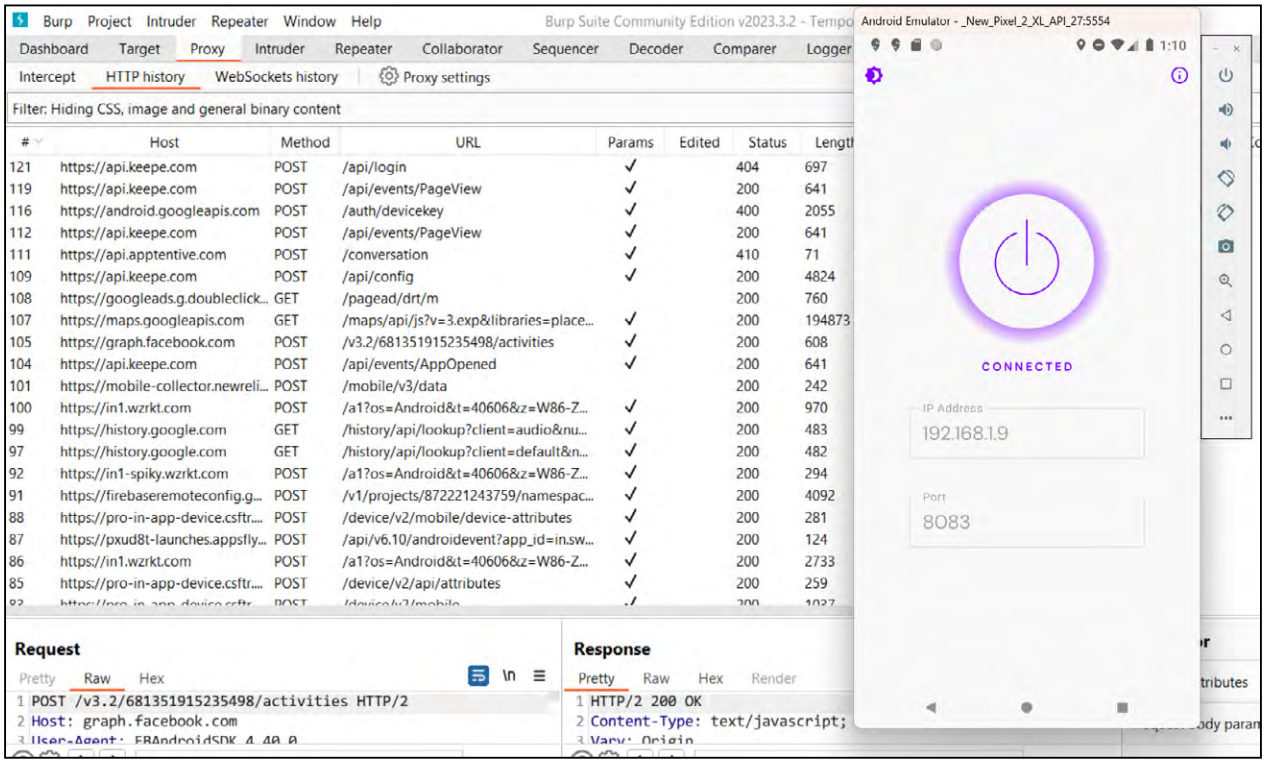
12. Now click on “Install from storage” and select “AlwaysTrustUserCerts.zip”



13. After rebooting, check trusted credentials. The custom CA cert we installed will be under the system.



14. You will be able to intercept the device traffic



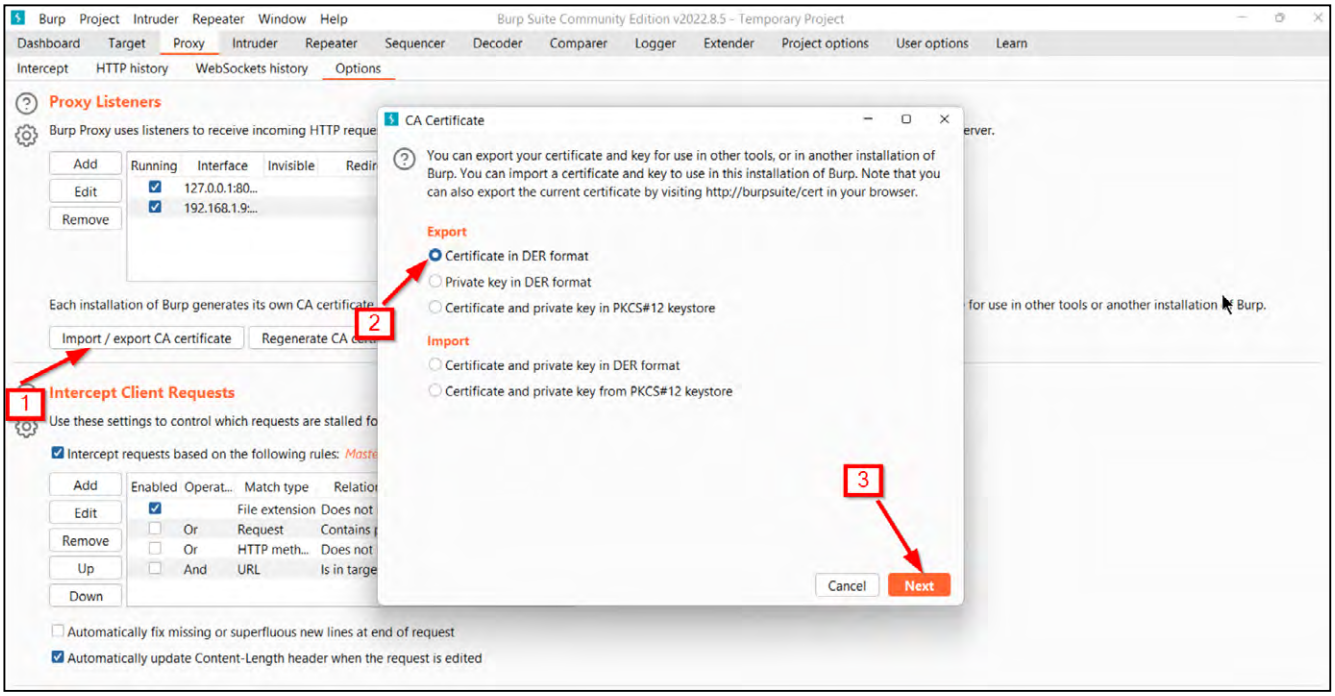
METHOD 2

Starting with Nougat, Android changed the default behavior regarding the trust of user-installed certificates. We can no longer simply install the Burp CA from the SD card to begin intercepting device traffic. Unless specified otherwise, apps will now *only* trust system-level certificate authorities (CAs).

We have to install a Burp CA certificate as a system-level CA on the device.

STEPS

1. Export Burp CA certificate from BurpSuite



2. Android wants certificate value in .pem format along with filename as subject_hash_old value appended with `.0`

3. We will use OpenSSL for this. Make sure you have installed OpenSSL.

4. Run the following commands:

```
openssl x509 -inform DER -in cacert.der -out cacert.pem
openssl x509 -inform PEM -subject_hash_old -in cacert.pemsdcard
```

```
root@payatu-PF3SDX2N:/mnt/c/Users/VedantWayal/Downloads/Burp# openssl x509 -inform DER -in burpcert.der -out burpcert.pem
root@payatu-PF3SDX2N:/mnt/c/Users/VedantWayal/Downloads/Burp# openssl x509 -inform PEM -subject_hash_old -in burpcert.pem
9a5ba575
-----BEGIN CERTIFICATE-----
MIIDqDCCApCgAwIBAgIFAOW0iOYwDQYJKoZIhvcNAQELBQAwgYoxFDASBgNVBAYT
C1BvcnRTd2lnZ2VyMRQwEgYDVQQIEwtQb3J0U3dpZ2dlcjEUMBIGA1UEBxMLUG9y
dFN3aWdnZXIxFDASBgNVBAoTC1BvcnRTd2lnZ2VyMRcwFQYDVQQLEw5Qb3J0U3dp
Z2dlciBDQTEXMBUGA1UEAxMOUG9ydFN3aWdnZXIgQ0EwHhcNMTQwOTI4MTgwNzAx
WhcNMzMwOTI4MTgwNzAxWjCBijEUMBIGA1UEBhMLUG9ydFN3aWdnZXIxFDASBgNV
BAgTC1BvcnRTd2lnZ2VyMRQwEgYDVQQHEwtQb3J0U3dpZ2dlcjEUMBIGA1UEChML
UG9ydFN3aWdnZXIxFzAVBgNVBAsTDlBvcnRTd2lnZ2VyIENBMRcwFQYDVQQDEw5Q
b3J0U3dpZ2dlciBDQTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANIx
```

5. Copy the hash value and rename burpcert.pem as .0

```
mv burpcert.pem <hash>.0
```

6. Restart the device as root and remount the partition

```
adb root
adb remount
adb push <cert>.0 /sdcard/
```

```
C:\Users\VedantWayal\Downloads\Burp>adb root

C:\Users\VedantWayal\Downloads\Burp>adb remount
remount succeeded

C:\Users\VedantWayal\Downloads\Burp>adb push 9a5ba575.0 /sdcard/
9a5ba575.0: 1 file pushed, 0 skipped. 0.1 MB/s (1330 bytes in 0.017s)
```

7. If you get any “Read-only” error then close the emulator and try to run the emulator with the -writable-system flag with the following method.

```
emulator -list-avds
emulator -writable-system -avd _New_Pixel_2_XL_API_27 -no-snapshot-load -qemu
```

(Make sure you have set the emulator path in the environment variable. For example `C:\Users\VedantWayal\AppData\Local\Android\Sdk\emulator`)

8. Take shell access of the device and move the certificate to /cacerts/ directory

```
adb shell
mv /sdcard/<cert>.0 /system/etc/security/cacerts/
```

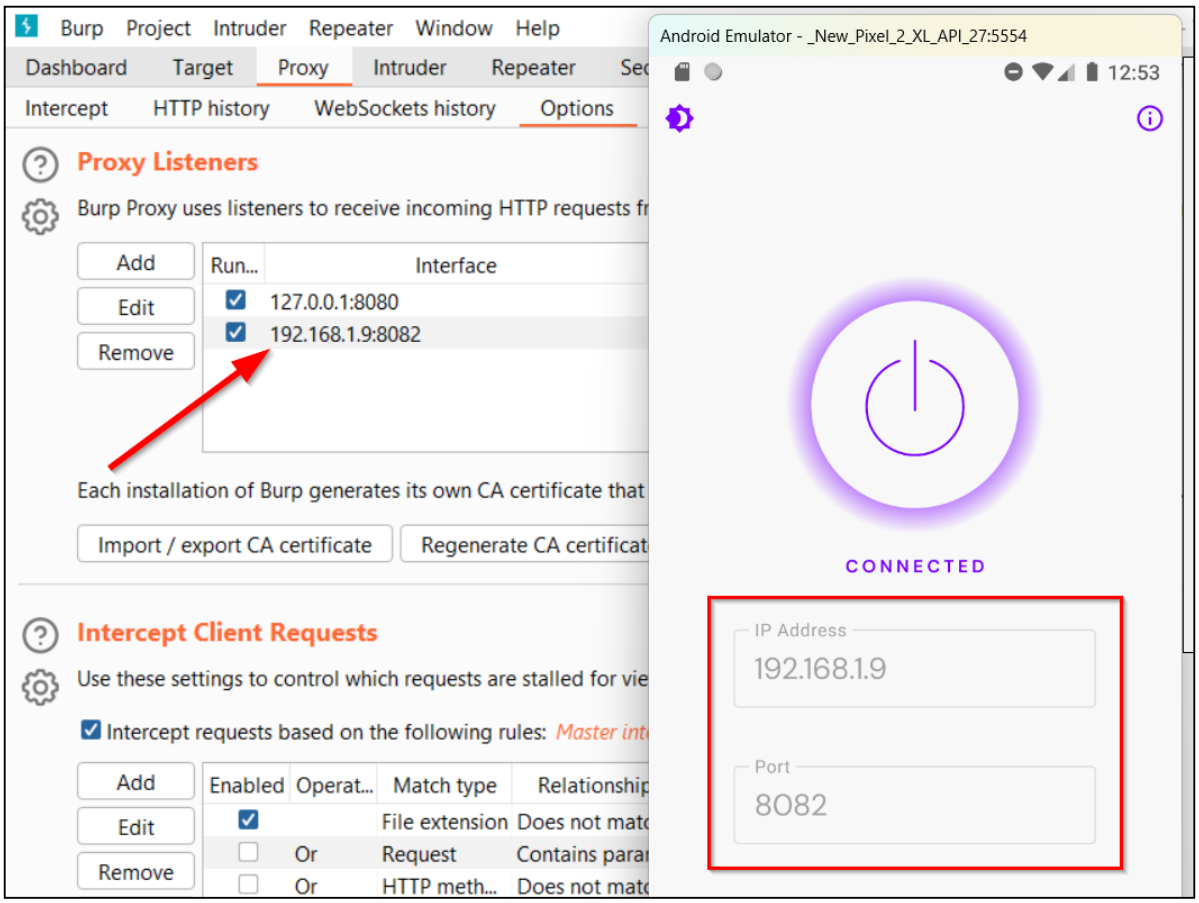
```
C:\Users\VedantWayal\Downloads\Burp>adb shell
generic_x86:/ # mv /sdcard/9a5ba575.0 /system/etc/security/cacerts/
generic_x86:/ # ls /system/etc/security/cacerts/
00673b5b.0 1df5a75f.0 2fa87019.0 3c9a4d3b.0 52b525c7.0 69105f4f.0 82223c44.0 9576d26b.0 a3896b44.0 b936d1c6.0 ccc52f49.0 d8317ada.0 facacbc6.0
02756ea4.0 1e1eab7c.0 31188b5e.0 3d441de8.0 559f7c71.0 6e8bf996.0 85cde254.0 95aff9e3.0 a7605362.0 bc3f2570.0 cf701eeb.0 dbc54cab.0 fb5fa911.0
04f60c28.0 1e8e7201.0 343eb6cb.0 3e7271e8.0 5a250ea7.0 6fcc125d.0 86212b19.0 9685a493.0 a7d2cf64.0 bdacca6f.0 d06393bb.0 dc99f41e.0 fd08c599.0
0d5a4e1c.0 1eb37bdf.0 35105088.0 40dc992e.0 5a3f0ff8.0 75680d2e.0 87753b0d.0 9772ca32.0 a81e292b.0 bf64f35b.0 d16a5865.0 dfc0fe80.0 fde84897.0
0d69c7e1.0 1f58a078.0 3929ec9f.0 418595b9.0 5cf9d536.0 76579174.0 882de061.0 9a5ba575.0 ab5346f4.0 c491639e.0 d18e9066.0 e442e424.0
10531352.0 21855f49.0 399e7759.0 455f1b52.0 5e4e69e7.0 7672ac4b.0 89c02a45.0 9c3323d4.0 aeb67534.0 c51c224c.0 d41b5e2a.0 e48193cf.0
111e6273.0 219d9499.0 3a3b02ce.0 48a195d8.0 5f47b495.0 7999be0d.0 8d6437c3.0 9d6523ce.0 b0ed035a.0 c7e2a638.0 d4c339cb.0 e775ed2d.0
12d55845.0 23f4c490.0 3ad48a91.0 4be590e0.0 60afe812.0 7a7c655d.0 91739615.0 9dbefe7b.0 b0f3e76e.0 c907e29b.0 d59297b8.0 e8651083.0
17b51fe6.0 27af790d.0 3c58f906.0 4e18c148.0 6187b673.0 7a819ef2.0 9282e51c.0 9f533518.0 b3fb433b.0 c90bc37d.0 d66b55d9.0 ea169617.0
1dac3003.0 2add47b6.0 3c6676aa.0 5046c355.0 63a2c897.0 7d453d8f.0 9339512a.0 a0bc6fbb.0 b7db1890.0 cb156124.0 d6e6eab9.0 ed39abd0.0
1dcd6f4c.0 2d9dafe4.0 3c860d51.0 524d9b43.0 67495436.0 81b9768f.0 9479c8c3.0 a2c66da8.0 b872f2b4.0 cb1c3204.0 d7746a63.0 ee7cd6fb.0
generic_x86:/ #
```

9. Change the permission of this certificate and reboot the device

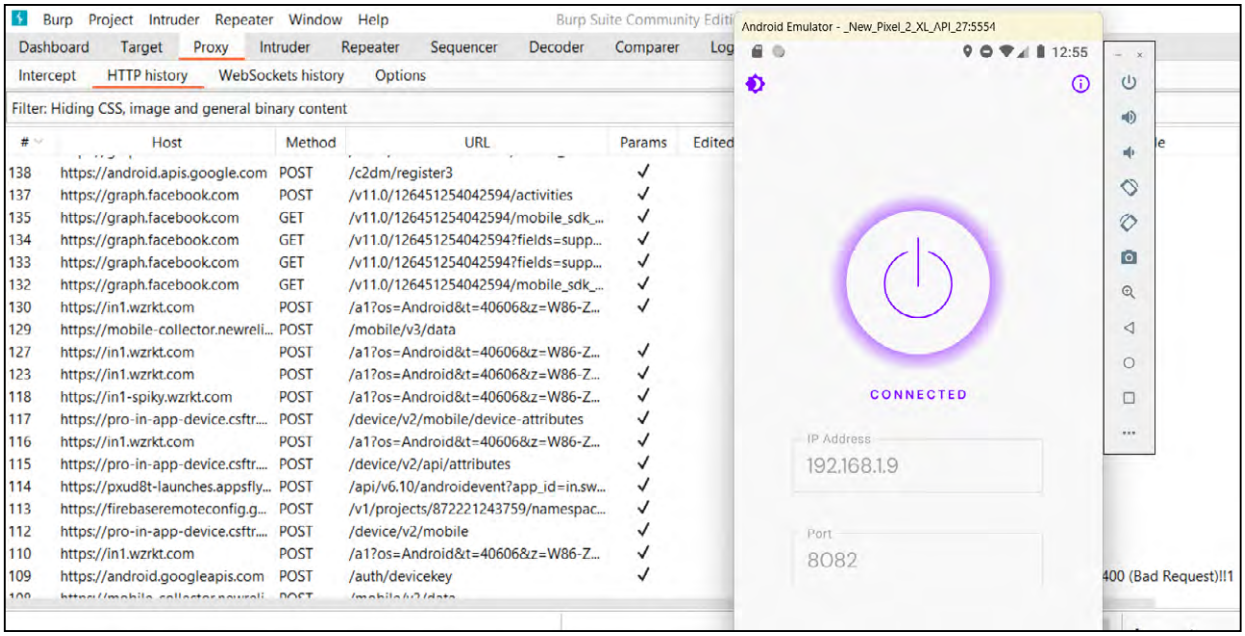
```
chmod 644 /system/etc/security/cacerts/9a5ba575.0
adb reboot
```

```
generic_x86:/ # chmod 644 /system/etc/security/cacerts/9a5ba575.0
generic_x86:/ # reboot
```

10. The certificate will be installed successfully. Now set up the proxy with Burp

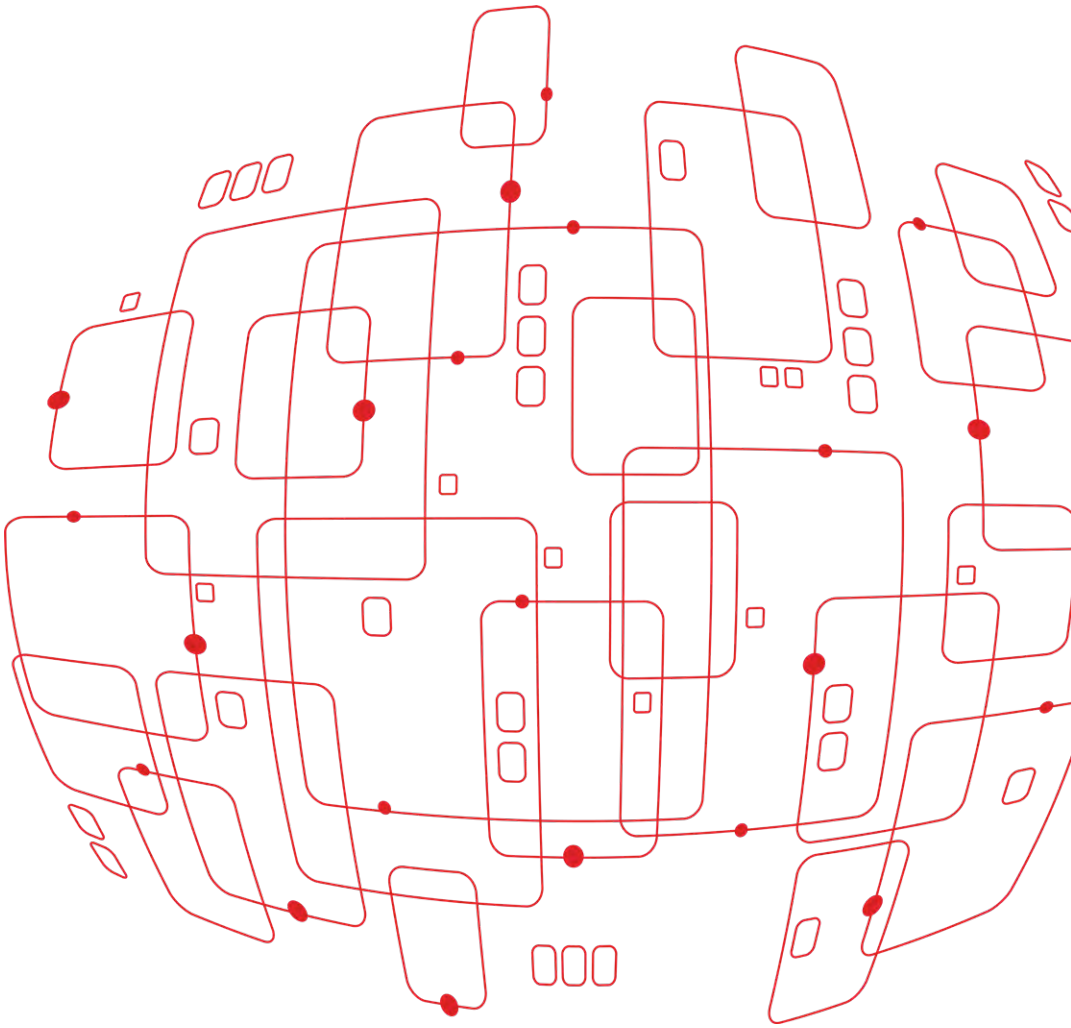


11. Now you can intercept the Android Device traffic into BurpSuite.



2.2 Useful Magisk Modules & Tools

1. **Always Trust User Certificates**
   - This module makes all installed user certificates part of the system certificate store, so that they will automatically be used when building the trust chain. This module makes it unnecessary to add the network_security_config property to an application's manifest.
   - https://github.com/NVISOsecurity/MagiskTrustUserCerts
2. **MagiskFrida**
   - MagiskFrida lets you run frida-server on boot with Magisk
   - https://github.com/ViRb3/magisk-frida
3. **ADB Root**
   - This module allows you to run the ADB daemon from the root user.
   - https://github.com/evdenis/adb_root
4. **De-bloater**
   - De-Bloater is an application using the power of Magisk to de-bloat unwanted applications systemless-ly.
   - https://github.com/sunilpaulmathew/De-Bloater
5. **zygisk-frida**
   - ZygiskFrida is a zygisk module allowing you to inject frida gadget in Android applications in a more stealthy way.
   - https://github.com/lico-n/ZygiskFrida
6. **MagiskTrustUserCerts**
   - This module makes all installed user certificates part of the system certificate store so that they will automatically be used when building the trust chain. This module makes it unnecessary to add the network_security_config property to an application's manifest.
   - https://github.com/NVISOsecurity/MagiskTrustUserCerts
7. **Android Proxy Toggle**
   - Wasting time turning proxy on and off from Wi-Fi Settings during a pentest? Here is Proxy Toggle for Android, which can help avoid that. You have to configure it once and then use a quick setting tile to toggle the proxy on and off.
   - https://github.com/theappbusiness/android-proxy-toggle



3. SETTING UP THE iOS DEVICE

Using Physical device

3.1 Initial Steps

The first step is ensuring that your iPhone can connect to your PC.

Verify the connectivity of the cable, phone, and computer.

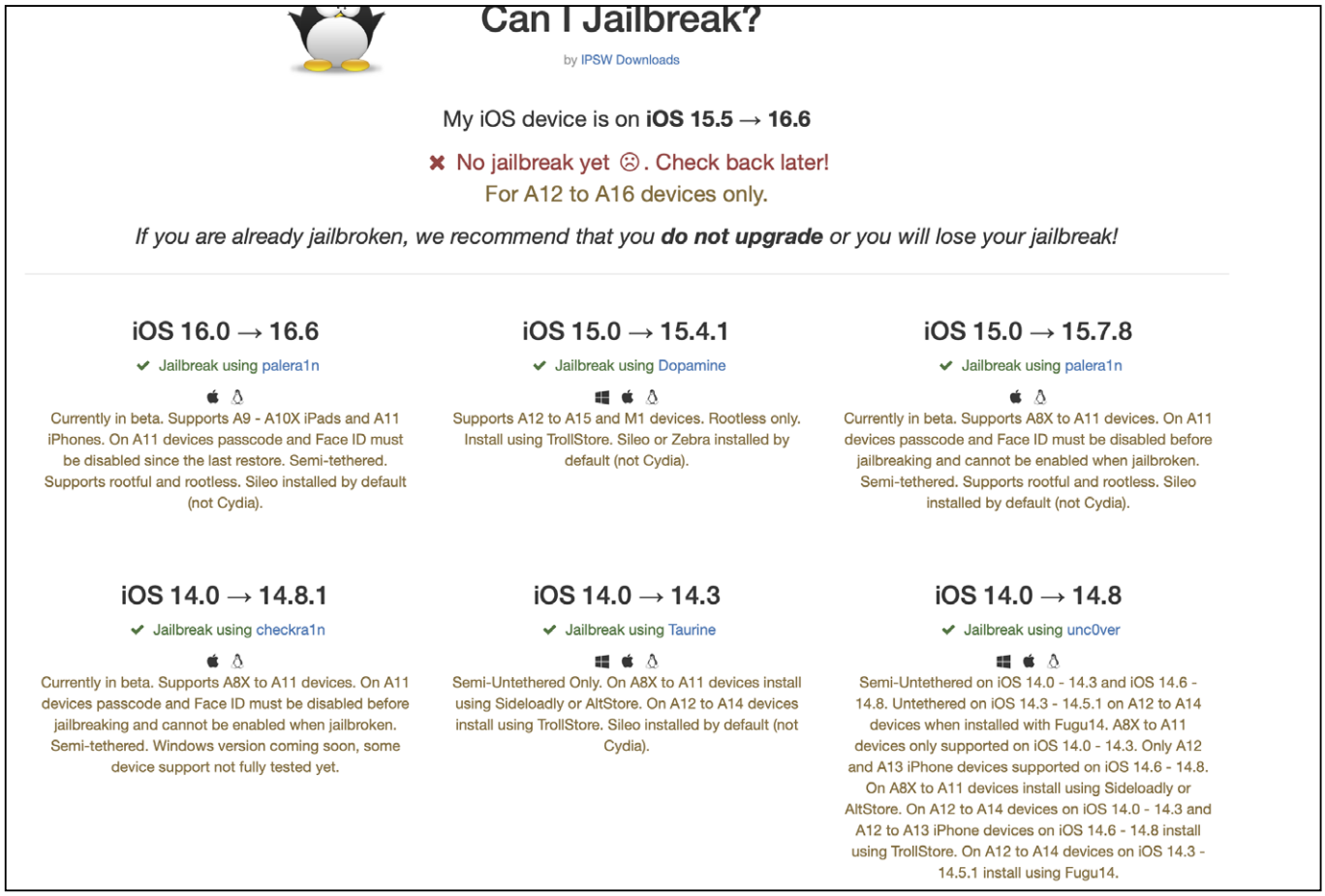
I will assume this is a test device, not a personal one.

Connect the phone to the computer, then check if a pop-up asking you to trust the device appears. If it does, open the Mac's Finder program. Access the phone using Finder, and click on the trust option.

On the phone, this should prompt the pop-up. If your trust is established and you can connect successfully between the computer and iPhone, proceed to the next step.

3.2 Jailbreak

Jailbreak is the process of removing user restrictions imposed by the manufacturer on your device. It's important to note that jailbreaking voids the warranty of your device. There are various jailbreaking methods for iOS, which vary depending on the version. Canijailbreak is a website that can provide suggestions for jailbreak tools compatible with your iOS version. Additionally, it's worth noting that jailbreaking can be done using Windows, Mac, or Linux. There are also different types of jailbreaking classifications.



For our testing, we will need root privileges on our phones. This can be achieved through jailbreaks, which are techniques to gain root access. The method you use will depend on the hardware of the iOS device and its version. I am using an iPhone X on iOS 14.4.1. In this case, I will be using CheckRa1n to jailbreak the device.

Checkra1n can be used on any iOS device as it is a hardware exploit. Download CheckRa1n onto your computer from its website (you may want to use a virtual machine, but you may experience issues if you do). Once you have it, run it (it only works on Mac and Linux), you should see a pop-up window with information on Checkra1n and an options button.

Plug in the iOS to the USB now, and you should see the device version and the iOS version appear in the Checkra1n box. Since 13.5.1 was outside the tested range at the time of writing, I needed to go into the options on Checkra1n and click on the top option to enable untested versions. After hitting back, I could hit start.

If you are still confused, check out this YouTube video and follow the steps.

Follow the on-screen instructions, and if all goes well, you should be jailbroken within a couple of minutes!

This is a semi-tethered jailbreak, which means it will last until you reboot the device. After re-booting, you will need to repeat this process.

TYPES OF JAILBREAK

Below are some types of jailbreaking techniques, and you can jailbreak your device as per your requirement.

UNTETHERED JAILBREAK: Permanent jailbreak; the device will be jailbroken even after a reboot.

TETHERED JAILBREAK: Temporary jailbreak; after a reboot, the device will return to its normal state.

SEMI-TETHERED JAILBREAK: A semi-tethered jailbreak is one where the device can begin independently, but it'll not have a patched kernel and thus won't be ready to run modified code.

SEMI-UNTETHERED JAILBREAK: A semi-untethered jailbreak is analogous to an untethered jailbreak wherein it allows the device to stay abreast of its own. The device's start-up sequence is unaltered on each boot, and it boots into its original, non-jailbroken state. However, instead of employing a computer to jailbreak, as during a tethered or semi-tethered case, the users can re-jailbreak their devices using an app that runs on their device.

WHICH JAILBREAK TECHNIQUE IS BETTER?

We recommend Checkra1n, but there are other techniques such as Uncover, Odyssey, etc.

For example, if you take Odyssey and unc0ver, there's not much difference between them. In fact, Odyssey uses a reverse-engineered version of the Unc0ver exploit, which was detailed by Project Zero.

Talking about *Unc0ver* vs *Checkra1n* vs. *Odyssey jailbreak* and which is better, it does not really matter if you use Checkra1n or Odyssey or Unc0ver as long as you don't pay attention to the tweaks you use.

If you are utilizing modifications from unidentified sources, pirated sources, or generally shady-looking sources other than the official ones, you may use something like tweak compatible, for instance, to determine whether your tweaks are compatible with your iOS version.

Neither the quality of the jailbreak nor the iOS version you are using are relevant. It won't improve since the changes are from questionable, pirated, or unreliable sources, or they aren't updated or updated incorrectly.

They are changed, which naturally has the potential to cause problems, and they occasionally even include malware.

Find out here how to save your Cydia customizations and subsequently restore them.

The quality of the jailbreak is, therefore, irrelevant. Suitable adjustments might assist you once more with this issue.

Fun Fact: you can get and install tweak compatible for free from Cydia.

You may choose an iOS version and determine whether a given change is compatible with it or not.

In conclusion, you shouldn't select the jailbreak for a company based just on the design of its logo or who made it (Unc0ver vs Checkra1n vs. Odyssey). However, you should give them a try to see how they perform on your specific device and, of course, how they respond to the modifications you apply.

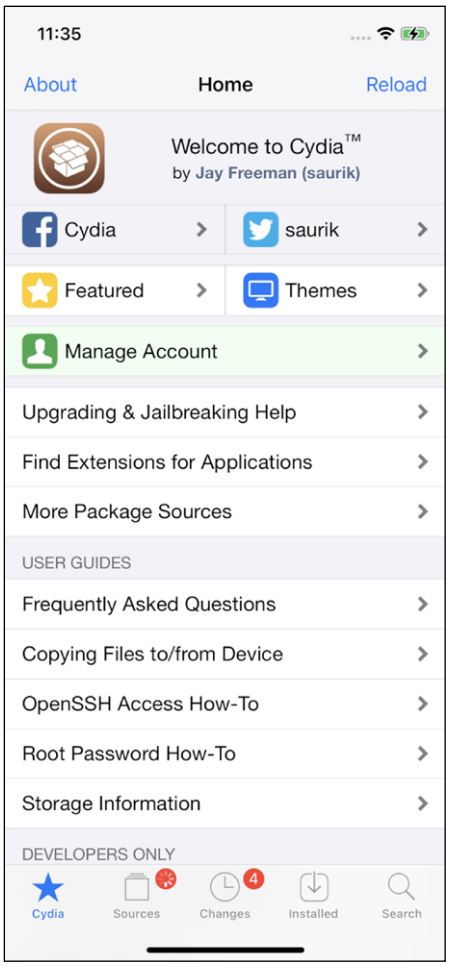
Still, the essential thing to have your jailbreak work perfectly fine is to keep it updated to the latest version. And, of course, to use tweaks from official sources. It is also worth mentioning that you should never use too many tweaks. If you don't need it, do not install it. Only use the tweaks you need.

3.3 Installing Tools on iOS

Now that you are jailbroken, you should see the Cydia application on your phone. This is a package manager for the device and apps you can't get without root. Go through these steps.

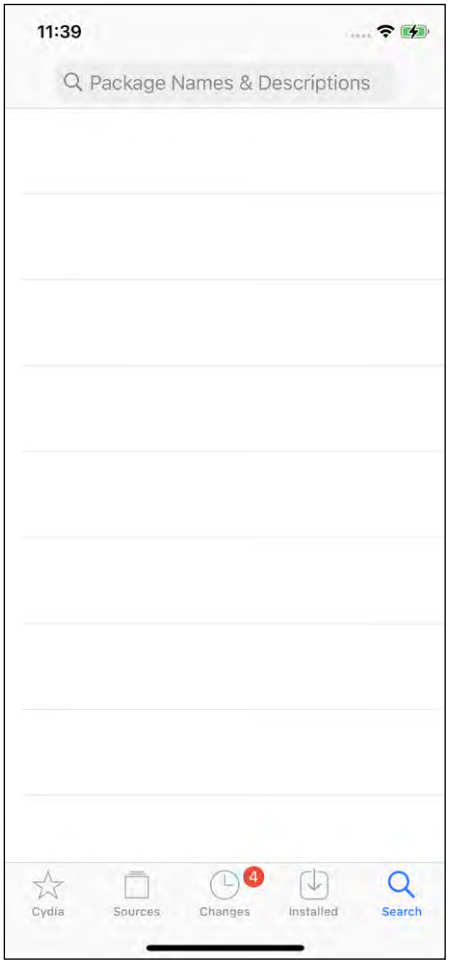
SSH

1. Open Cydia



2. If it says you are missing packages, hit on Upgrade all

3. Click on Search at the bottom

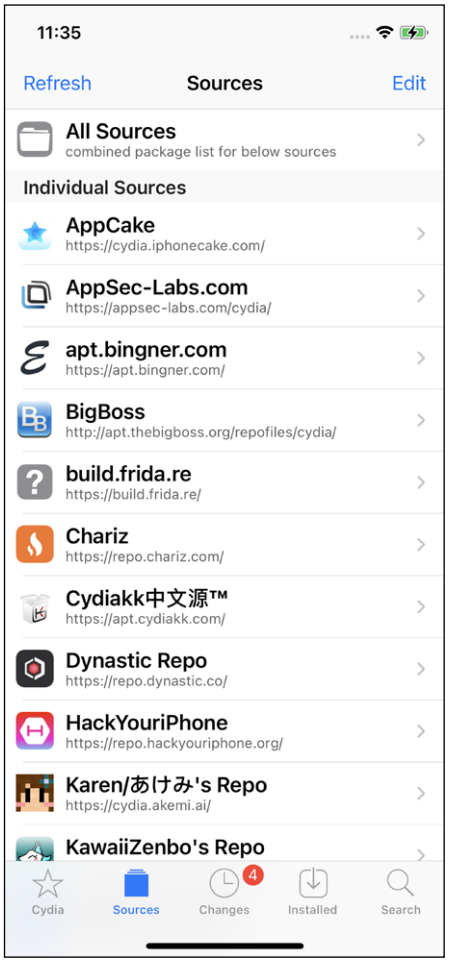


4. Search for and install any tool which you want

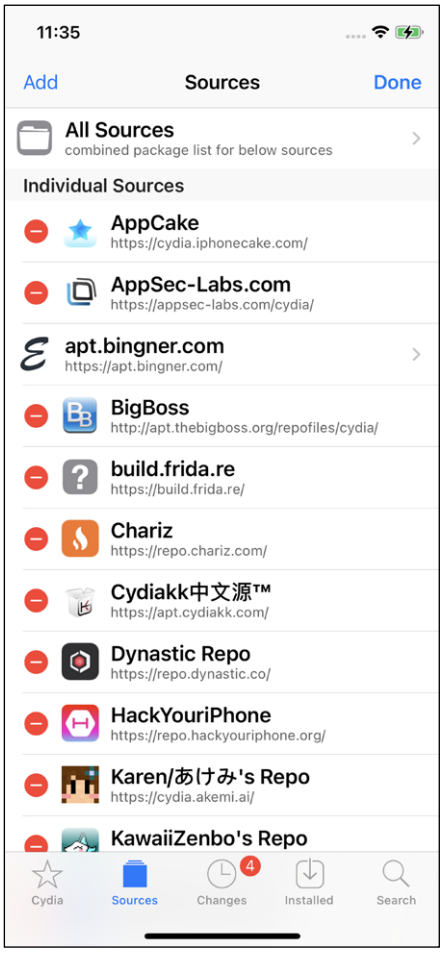
   Install SSH and try to check if it's working or not. To check if this worked or not, open the iOS Settings > General > WiFi > Click on your connected Wifi network > Note the IP address of the device terminal and take SSH shell: ssh root@iphoneIP with the default password “alpine” and you should be able to connect. (We recommend changing the password after taking the shell of the device).

ADD SOURCES IN CYDIA AS PER THE BELOW STEPS.

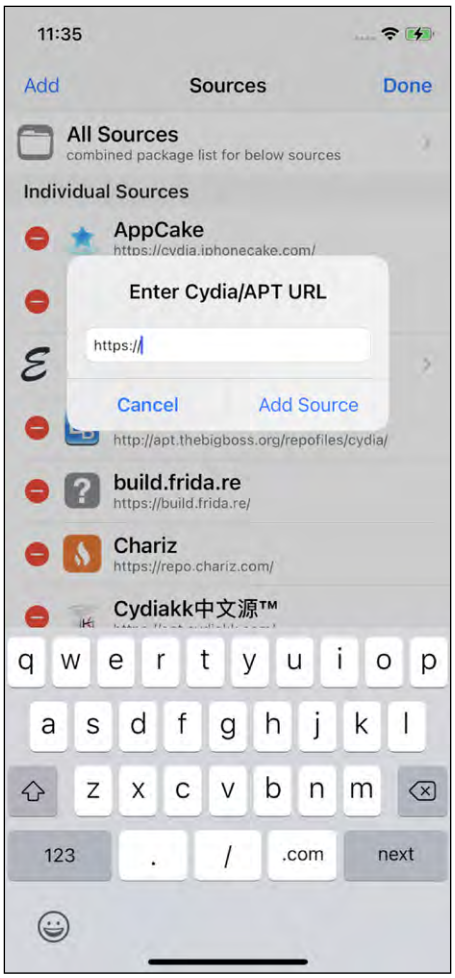
1. Click on the Sources button at the bottom



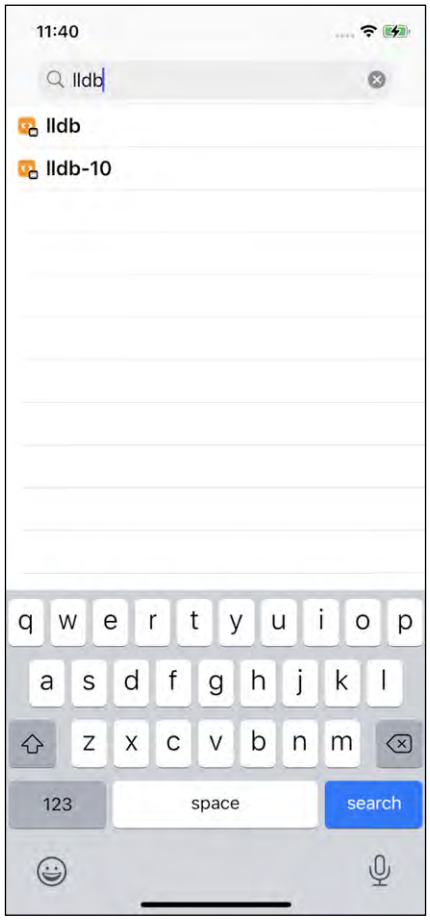
2. Click on edit at the top right.



3. Click on add in the top left.
4. Add `https://XYZ.COM/`



5. Tap Return to Cydia.
6. Click on Search.
7. Search for any tool and install it on your device.
8. You can use the installed tool now.



9. Add all those below resources in your Cydia and restart the springboard.

```
http://cydia.iphonecake.com
http://apt.saurik.com/
http://repo.nesolabs.de/
https://build.frida.re/
http://appsec-labs.com/cydia/
http://cydia.zodttd.com/repo/cydia/
http://mobiletools.mwrinfosecurity.com/cydia/
http://repo666.ultrasn0w.com/
http://apt.thebigboss.org/repofiles/cydia/
http://cydia.radare.org/
http://apt.modmyi.com/
http://coolstar.org/publicrepo/
http://getdelta.co/
http://julioverne.github.io/
```

```
http://apt.bingner.com/
http://repo.dynastic.co/
http://mcapollo.github.io/Public/
http://apt.hackcn.net/
http://repo.chariz.io/
http://cydia.ichitaso.com/
https://level3tjg.github.io
http://ryleyangus.com/repo
*https://havoc.app/**
```

(How to Restart the spring boot)

- Take the shell of the device using its SSH and IP address
- After getting the shell of the iOS device, run the below command
- Run `killall SpringBoard`
- Retake a shell and use that tool
- Run tool name

3.4 Installing Tools on Mac

XCODE

Utilize the Apple Store to install Xcode. This is the accepted method to offer the ability to simulate iOS, build iOS applications, and some useful tools to make it easier for us to load programs onto the iPhone. For me, downloading this took a long time.

FRIDA

Here, we need some prerequisites for installing the Frida. Make sure you have installed Python3 on your device. Linux and Mac should have this pre-installed. You may need to do so if you haven't installed it yet.
Make sure you have pip3 installed on your device. If you don’t, then you can install it through Python3.
Installation steps for Frida using pip3.

```
pip3 install frida-tools
```

You should now be able to run the following in the terminal if installation is successful.
(connect your iOS device with a USB cable and system)

```
frida-ps -U
```

You should see some application names that relate to things on the iOS device.

GHIDRA

Ghidra is not required; however, it can be helpful to reverse-engineer some applications. We advise utilizing any method you find most comfortable. We’ll utilize this because I’ve been utilizing Ghidra a lot recently.

You must visit the Ghidra website and download the ZIP file that is available there. Since it is only a self-contained Java file, installation is not necessary. The next step is to get the most recent JDK by searching for one on Google. Installing this is required.

Once you have the JDK installed, you should be able to go to the Ghidra directory in the terminal and use `ghidrarun` to run Ghidra. For Ghidra basics, you can check out my Minesweeper hacking post.

To load an application, you will need to follow these steps:

1. Find the .ipa file you want to analyze.
2. Rename it to a .zip file.
3. Unzip the contents.
4. Have a look inside the contents and go into the Payload folder.
5. You should be able to see a .app file. This is the application and will be named the same as the application.
6. Load the .app file into the reversing tool (for Ghidra, you will need to make a new project and then import the file).

OBJECTION

The objection will help inject into apps and be a useful addition to Frida. It can be installed in the same way with pip.

```
pip3 install objection
```

Using the command `objection` should show the help menu.

KEYCHAIN DUMPER

Dumping data from the keychain will be important. This can be done with a dedicated tool. However, I will do it through other tools (Passionfruit seems to work well for keychain access).

PASSIONFRUIT

Passionfruit is a GUI application that allows us to conveniently view data about the application. To install this, we will need to get a node.

```
brew install node
```

Then, we will need to use the node package manager (npm) to download Passionfruit.

```
npm install -g passionfruit
```

Now, we should just be able to run it!

```
passionfruit
```

This should show a URL (localhost:31337 by default). Browse this URL in your browser, and you should see the Passionfruit home page. Connect your device over USB, and you should see your device there as well!
Note: This will require Frida and often crash, but it is easy to reboot.

SQLITE BROWSER

Googling the SQLite browser and downloading should work well. Installation is the same as a regular Mac app.

REALM BROWSER

This can be retrieved from the Apple Store. Search for the Realm Browser and install it, and you should be good to go.

3.5 Installing Tools on Windows

1. **iFunBox:**
   - **Use Case:** iFunBox allows for file management on iOS devices, making it useful for extracting app data, accessing file systems, and performing forensic analysis.
   - **Reference:** iFunBox Official Website

2. **iMazing:**
   - **Use Case:** iMazing is a comprehensive iOS device management tool that supports tasks like backup, file transfer, app management, and data extraction. It's valuable for data backup and recovery, as well as app analysis.
   - **Reference:** iMazing Official Website

3. **iTunes:**
   - **Use Case:** iTunes, though primarily a media player, is used in iOS penetration testing for tasks such as backups, restores, and firmware updates. It's vital for creating and restoring device backups.
   - **Reference:** iTunes Download Page

4. **Frida:**
   - **Use Case:** Frida is a dynamic instrumentation toolkit that allows for hooking and modification of function calls in running processes. It's employed for runtime manipulation and analysis of iOS applications.
   - **Reference:** Frida Official GitHub Repository

5. **Objection:**
   - **Use Case:** Objection is a runtime mobile exploration toolkit that facilitates dynamic analysis of iOS apps. It provides functionalities for tasks like SSL unpinning, exploring UI components, and more.
   - **Reference:** Objection Official GitHub Repository

6. **Ghidra:**
   - **Use Case:** Ghidra is an open-source software reverse engineering (SRE) framework. It's used for analyzing binaries, disassembling code, and reverse engineering iOS applications for security assessment.
   - **Reference:** Ghidra Official Website

3.6 Useful Tweaks & Tools for iOS Device

1. **Class Dump:**
   - **Description:** Class Dump is a command-line tool used for examining the Objective-C runtime information stored in Mach-O files. It extracts class and method information from the compiled binary.
   - **Usage:** Pentesters use Class Dump to retrieve class names, methods, and their signatures from iOS applications. This information aids in understanding the app's functionality and identifying potential vulnerabilities.
   - **Reference Link:** Class-Dump GitHub Repository
2. **Substrate:**
   - **Description:** Substrate is a platform that simplifies the development of third-party add-ons (tweaks) for iOS. It provides a framework for hooking into and modifying the behavior of iOS apps.
   - **Usage:** Pentesters leverage Substrate to develop custom modifications to iOS applications during security assessments. This may involve bypassing security checks, altering functionality, or injecting additional code for testing purposes.
   - **Reference Link:** Substrate GitHub Repository
3. **cycript:**
   - **Description:** Cycript is a JavaScript-to-JavaScript compiler and immediate mode console environment. It can be injected into running processes, allowing dynamic interaction and modification of the app's behavior.
   - **Usage:** Pentesters use cycript for runtime analysis and manipulation of applications. It enables inspection of live objects, dynamic patching of code, and automation of interactions for security testing.
   - **Reference Link:** Cycript GitHub Repository
4. **AppList:**
   - **Description:** AppList allows developers to query the list of installed apps on an iOS device. It provides a structured view of installed applications, including their bundle identifiers.
   - **Usage:** Pentesters use AppList to gather information about installed apps on a target device. This information is valuable for identifying potential attack surfaces, analyzing dependencies, and targeting specific apps for security assessments.

- **Reference Link:** AppList GitHub Repository

5. **PreferenceLoader:**
   - **Description:** PreferenceLoader is a MobileSubstrate-based utility that allows developers to add entries to the Settings application. This enables the creation of custom preference panes, similar to those used by AppStore apps.
   - **Usage:** Pentesters utilize PreferenceLoader to create custom settings panels for tweaks or modifications made using Substrate. This allows for user-configurable options for customizations during security assessments.
   - **Reference Link:** PreferenceLoader GitHub Repository
6. **AppSync Unified:**
   - **Description:** AppSync Unified enables the installation of unsigned iOS applications, allowing users to sync and install applications that haven't been signed by Apple's App Store.
   - **Usage:** While primarily used for development purposes, pentesters can utilize AppSync Unified to install and test unsigned applications that may not be available through official channels.
   - **Reference Link:** AppSync Unified GitHub Repository
7. **LLDB:**
   - **Description:** LLDB is a powerful debugger included in Xcode, used for reversing and debugging applications.
   - **Usage:** Pentesters use LLDB to analyze and debug iOS applications during security assessments. It allows them to step through code, set breakpoints, inspect memory, and understand the inner workings of the application for vulnerability discovery.
   - **Reference Link:** LLDB on LLVM GitHub Repository
8. **otool:**
   - **Description:** The otool command displays specified parts of object files or libraries, providing insights into the structure and content of Mach-O files.
   - **Usage:** Pentesters use otool to extract information about the binary, such as linked libraries, load commands, and other attributes. This information is valuable for understanding dependencies and potential vulnerabilities.
   - **Reference Link:** otool Documentation
9. **Clutch:**
   - **Description:** Clutch is a tool for decrypting iOS applications and dumping them into binary or .ipa files. It allows security researchers to access the unencrypted application code for analysis.

- **Usage:** Pentesters use Clutch to obtain the decrypted version of iOS applications. This allows for deeper analysis, such as static code analysis and identifying potential vulnerabilities that might not be visible in the encrypted version.
- **Reference Link:** Clutch GitHub Repository

10. Dumpdecrypted:

- **Description:** Dumpdecrypted is a tool used to extract decrypted Mach-O files from encrypted iPhone applications in memory. It's crucial for security researchers to analyze the code under encryption.
- **Usage:** Pentesters employ Dumpdecrypted to retrieve the decrypted binary of an iOS application, which can then be analyzed using various security testing techniques.
- **Reference Link:** Dumpdecrypted GitHub Repository

11. Passionfruit:

- **Description:** Passionfruit is a simple iOS app blackbox assessment tool with a fully web-based GUI. It's powered by frida.re and vuejs, enabling dynamic analysis of iOS apps.
- **Usage:** Pentesters can use Passionfruit to perform black-box assessments of iOS applications. It provides a user-friendly interface for interacting with the app, monitoring function calls, and conducting runtime analysis to uncover security vulnerabilities.
- **Reference Link:** Passionfruit GitHub Repository

12. Filza:

- **Description:** Filza is a file manager app for iOS devices. It allows users to explore and manage the filesystem, including system files.
- **Usage:** Pentesters may use Filza to navigate through the filesystem of a jailbroken device, enabling them to inspect system files and configurations for security assessments.

13. Choicy:

- **Description:** Choicy is a tweak for jailbroken iOS devices that allows users to disable substrate (Cydia Substrate) in specific apps.
- **Usage:** Pentesters can use Choicy to selectively disable tweaks in certain apps. This is useful for isolating the effects of specific modifications during security testing.

14. HideJB:

- **Description:** HideJB is a tweak for jailbroken iOS devices that helps in detecting if a device is jailbroken and allows users to hide the fact that it's jailbroken from certain apps.
- **Usage:** Pentesters may use HideJB to assess how an app behaves on jailbroken devices. It helps in identifying apps that have anti-jailbreak mechanisms.

15. KernBypass:

- **Description:** KernBypass is a tweak for jailbroken iOS devices that bypasses kernel-level jailbreak detection.
- **Usage:** Pentesters can utilize KernBypass to bypass advanced jailbreak detection techniques implemented by apps, enabling them to assess the security of such apps on jailbroken devices.

16. Liberty Lite:

- **Description:** Liberty Lite is a tweak for jailbroken iOS devices that bypasses jailbreak detection in certain apps.
- **Usage:** Pentesters can use Liberty Lite to bypass jailbreak detection mechanisms in specific apps, allowing them to assess the security of those apps on jailbroken devices.

17. OpenSSH:

- **Description:** OpenSSH is a suite of secure networking utilities for remote administration of Unix-like systems.
- **Usage:** Pentesters can install OpenSSH on a jailbroken iOS device to gain remote access, enabling them to perform security assessments and tests from a remote location.

18. SafeShutdown:

- **Description:** SafeShutdown is a tweak for jailbroken iOS devices that enhances the shutdown process by ensuring that all running processes are terminated properly before the device powers off.
- **Usage:** Pentesters may use SafeShutdown to ensure that no processes are left running in the background before conducting security assessments on the device.

19. Shadow:

- **Description:** Shadow is a tweak for jailbroken iOS devices that helps in preventing screenshot detection by apps.
- **Usage:** Pentesters can use Shadow to assess how an app handles screenshots on jailbroken devices. It helps in identifying apps that implement screenshot detection mechanisms.

References

- https://docs.genymotion.com/desktop/03_Virtual_devices/
- https://www.genymotion.com/blog/release-note/genymotion-desktop-3-3-0/
- https://developer.android.com/docs/
- https://stackoverflow.com/questions/68955016/connect-to-android-emulator-from-android-studio-dev-env-in-a-virtualbox-vm
- https://support.genymotion.com/hc/en-us/articles/8957952431389-How-to-install-Magisk-on-Genymotion-
- https://github.com/theappbusiness/android-proxy-toggle
- https://infosecwriteups.com/hail-frida-the-universal-ssl-pinning-bypass-for-android-e9e1d733d29
- https://blog.ropnop.com/configuring-burp-suite-with-android-nougat/
- OWASP Mobile Security Testing Guide - iOS Testing
- Frida - iOS Setup
- Mobile Security Testing Guide - iOS Setup
- HackerSploit - Setting Up an iOS Pentesting Environment
- Medium - iOS Pentesting Setup
- Infosec Resources - iOS App Pentesting Setup Guide
- PentesterLand - Setting Up an iOS Pentesting Environment
- Null Byte - How to Set Up a New MacOS for Hacking
- Medium - iOS Application Security Testing Guide
- Security Trails - The Ultimate Guide to iOS Penetration Testing

Payatu

About Payatu

Payatu is a Research-powered cybersecurity services and training company specialized in IoT, Embedded Web, Mobile, Cloud, & Infrastructure security assessments with a proven track record of securing software, hardware and infrastructure for customers across 20+ countries.

Mobile Security Testing

Detect complex vulnerabilities & security loopholes. Guard your mobile application and user's data against cyberattacks, by having Payatu test the security of your mobile application.

IoT Security Testing

IoT product security assessment is a complete security audit of embedded systems, network services, applications and firmware. Payatu uses its expertise in this domain to detect complex vulnerabilities & security loopholes to guard your IoT products against cyberattacks.

Cloud Security Assessment

As long as cloud servers live on, the need to protect them will not diminish. Both cloud providers and users have a shared. As long as cloud servers live on, the need to protect them will not diminish.

Both cloud providers and users have a shared responsibility to secure the information stored in their cloud Payatu's expertise in cloud protection helps you with the same. Its layered security review enables you to mitigate this by building scalable and secure applications & identifying potential vulnerabilities in your cloud environment.

Payatu

Web Security Testing

Internet attackers are everywhere. Sometimes they are evident. Many times, they are undetectable. Their motive is to attack web applications every day, stealing personal information and user data. With Payatu, you can spot complex vulnerabilities that are easy to miss and guard your website and user's data against cyberattacks.

DevSecOps Consulting

DevSecOps is DevOps done the right way. With security compromises and data breaches happening left, right & center, making security an integral part of the development workflow is more important than ever. With Payatu, you get an insight to security measures that can be taken in integration with the CI/CD pipeline to increase the visibility of security threats.

Code Review

Payatu's Secure Code Review includes inspecting, scanning and evaluating source code for defects and weaknesses. It includes the best secure coding practices that apply security consideration and defend the software from attacks.

Red Team Assessment

Red Team Assessment is a goal-directed, multidimensional & malicious threat emulation. Payatu uses offensive tactics, techniques, and procedures to access an organization's crown jewels and test its readiness to detect and withstand a targeted attack.

Product Security

Save time while still delivering a secure end-product with Payatu. Make sure that each component maintains a uniform level of security so that all the components “fit” together in your mega-product.

Critical Infrastructure Assessment

There are various security threats focusing on Critical Infrastructures like Oil and Gas, Chemical Plants, Pharmaceuticals, Electrical Grids, Manufacturing Plants, Transportation systems etc. and can significantly impact your production operations. With Payatu's OT security expertise you can get a thorough ICS Maturity, Risk and Compliance Assessment done to protect your critical infrastructure.

CTI

The area of expertise in the wide arena of cybersecurity that is focused on collecting and analyzing the existing and potential threats is known as Cyber Threat Intelligence or CTI. Clients can benefit from Payatu's CTI by getting – social media monitoring, repository monitoring, darkweb monitoring, mobile app monitoring, domain monitoring, and document sharing platform monitoring done for their brand.

More Services Offered

- AI/ML Security Audit
- Trainings

More Products Offered

- EXPLIoT
- CloudFuzz

Payatu

Payatu Security Consulting Pvt. Ltd.

www.payatu.com

info@payatu.io

+91 20 41207726