

Android Userland Fuzzing and Exploitation

Lab Setup Guide

Lab Setup Outline

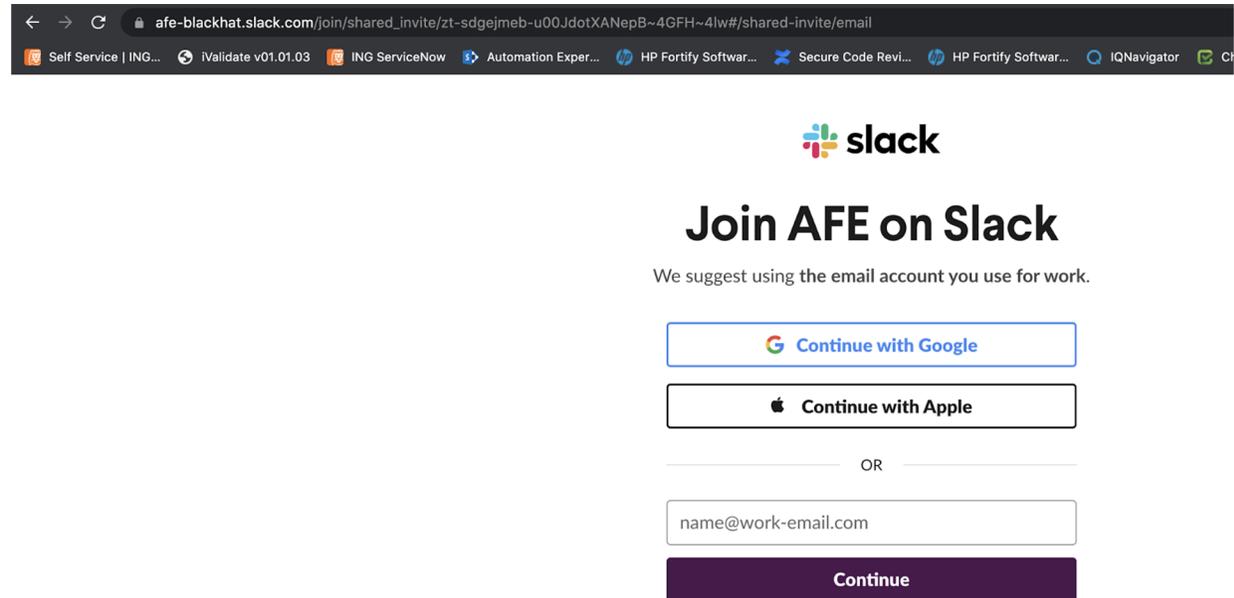
- Lesson 1: Slack Channel Setup
- Lesson 2: Install Vbox and Vbox Addition Package
- Lesson 3: Import Ubuntu/Fuzzing VM
- Lesson 4: Installing Android Studio
- Lesson 5: Installing NDK
- Lesson 6: Import "Vulnerable Project" in Android Studio
- Lesson 7: Download and run ARM Android VM
- Lesson 8: Connect to phone from Ubuntu VM
- Lesson 9: Corellium Lab Setup

Lesson 1: Slack Channel Setup

- Use the Invite link to join the AFE slack channel
 - https://join.slack.com/t/afe-blackhat/shared_invite/zt-sdgejmeh-u00JdotXANepB~4GFH~4lw

Lesson 1: Slack Channel Setup

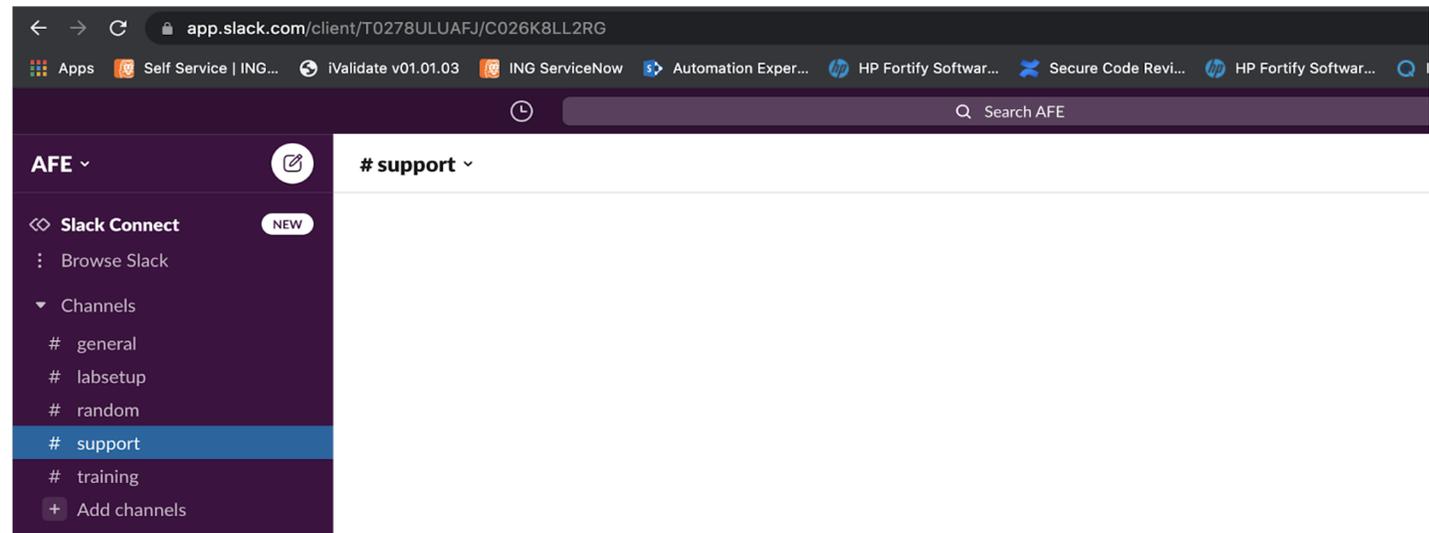
- Join the AFE channel either with your Gmail, Apple, or another email account



The screenshot shows a web browser window with the URL `afe-blackhat.slack.com/join/shared_invite/zt-sdgejmeh-u00JdotXANepB~4GFH~4lw#/shared-invite/email`. The page features the Slack logo at the top, followed by the heading "Join AFE on Slack". Below the heading, a message states "We suggest using the email account you use for work." There are two buttons for social login: "Continue with Google" and "Continue with Apple". Below these is the text "OR" and a text input field containing the placeholder email `name@work-email.com`. At the bottom is a dark purple "Continue" button.

Lesson 1: Slack Channel Setup

- Once you have joined the slack channel group, you can find multiple channels on the left panel.



- Use separate channels for communication
- If you have any queries on Lab setup, connect us through the Lab setup channel

Lesson 2: Install Vbox and Vbox Addition Package

- Install VirtualBox
- Install VirtualBox Addition Packages

Lesson 2: Install VirtualBox

- Download and Install VirtualBox for your OS
- <https://www.virtualbox.org/wiki/Downloads>

VirtualBox 6.1.22 platform packages

- [Windows hosts](#)
- [OS X hosts](#)
- [Linux distributions](#)
- [Solaris hosts](#)
- [Solaris 11 IPS hosts](#)

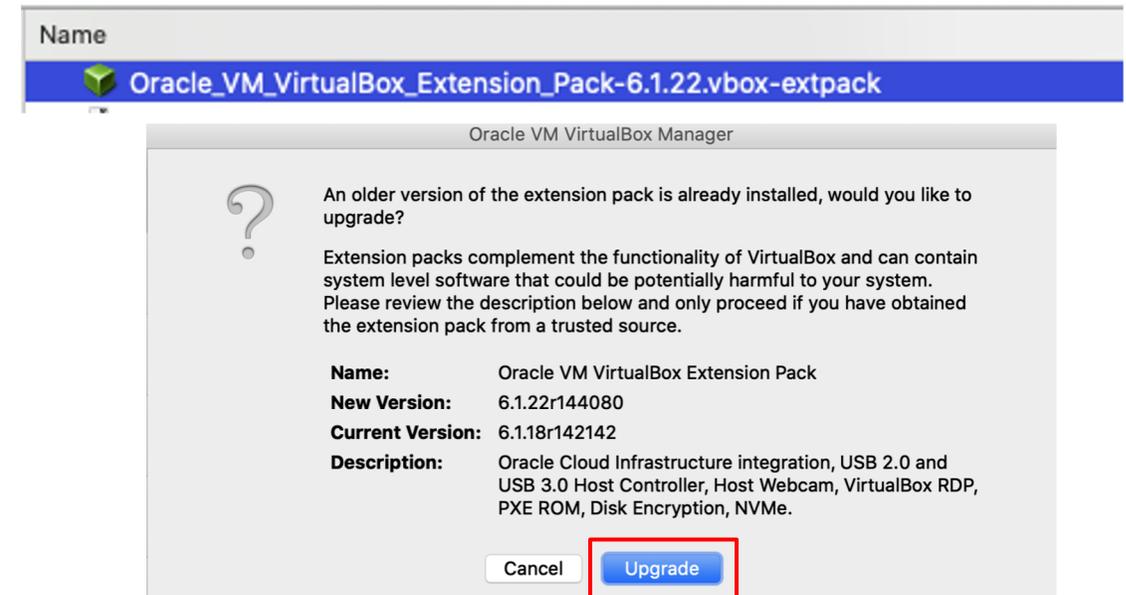
Lesson 2: Installing VirtualBox Extension Pack

- Download and install the VirtualBox Extension Pack
- <https://www.virtualbox.org/wiki/Downloads>
- Download the "All support platform" file
- Double click this file to import
- NOTE: The version number is very important, use the newest version

VirtualBox 6.1.22 Oracle VM VirtualBox Extension Pack

- [All supported platforms](#)

Support for USB 2.0 and USB 3.0 devices, VirtualBox RDP, disk encryption binaries are released under the [VirtualBox Personal Use and Evaluation](#)



Lesson 2: Wrap-up

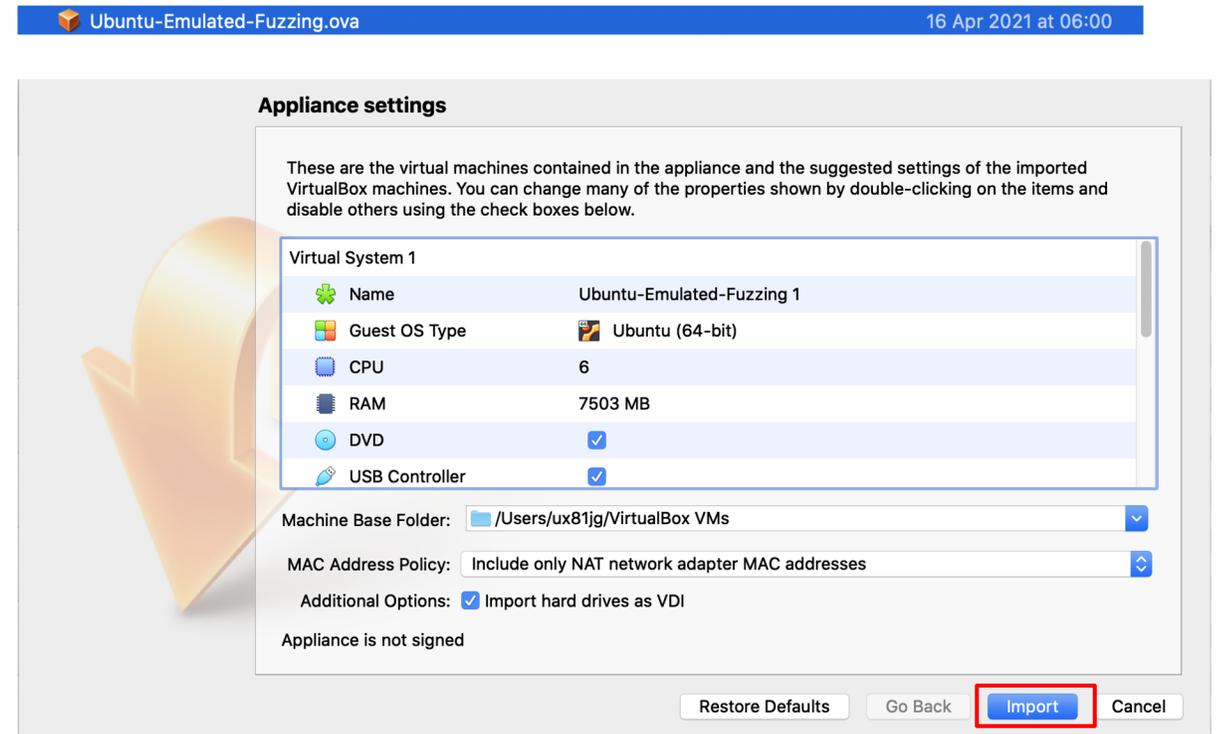
- Installed VirtualBox
- Installed VirtualBox Extension Pack

Lesson 3: Import Ubuntu/Fuzzing VM

- Import Ubuntu/Fuzzing VM

Lesson 3: Import Ubuntu/Fuzzing VM

- Download the training VM
- <https://drive.google.com/file/d/1AqBUVohKMxI22fMnNe4I1Fq7j4DQLzb7/view?usp=sharing>
- Import the file into VirtualBox by double clicking on the file
- Press the “Import” button to import



Lesson 3: Import Ubuntu/Fuzzing VM

- Downloaded the VM image
- Imported the VM image into VirtualBox



debian



Saved



Ubuntu-Emulated-Fuzzing (Installed Ghidra and finished qemu mode)



Saved



Lesson 4: Installing Android Studio

- Download Android Studio
- Install Android Studio

Lesson 4: Installing Android Studio

- Download Android Studio for your Operating System
- <https://developer.android.com/studio>
- Install according to your Operating System



Android Studio provides the fastest tools for building apps on every type of Android device.

DOWNLOAD ANDROID STUDIO

4.2 for Mac (937 MiB)

DOWNLOAD OPTIONS

RELEASE NOTES

Lesson 4: Wrap-up

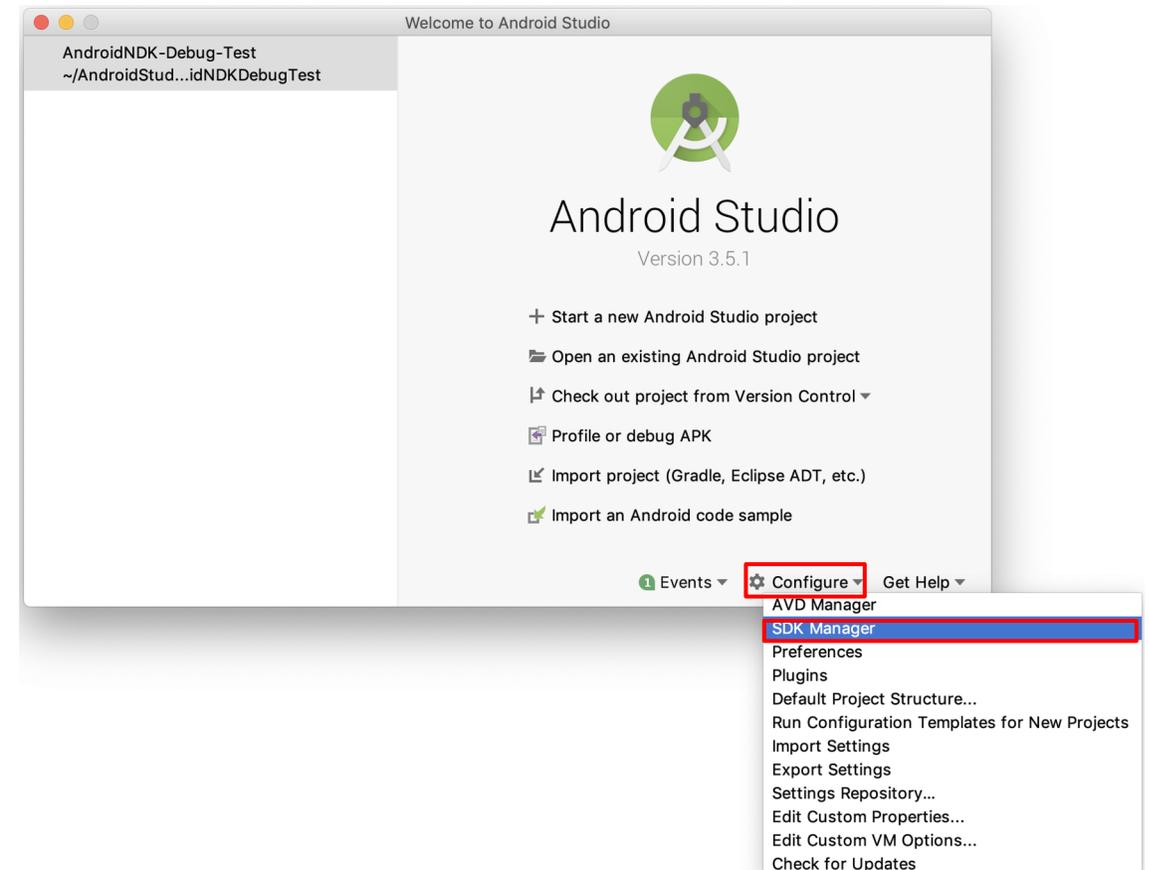
- Download Android Studio for your Operating System
- Install Android Studio

Lesson 5: Installing NDK

- Downloading and installing NDK

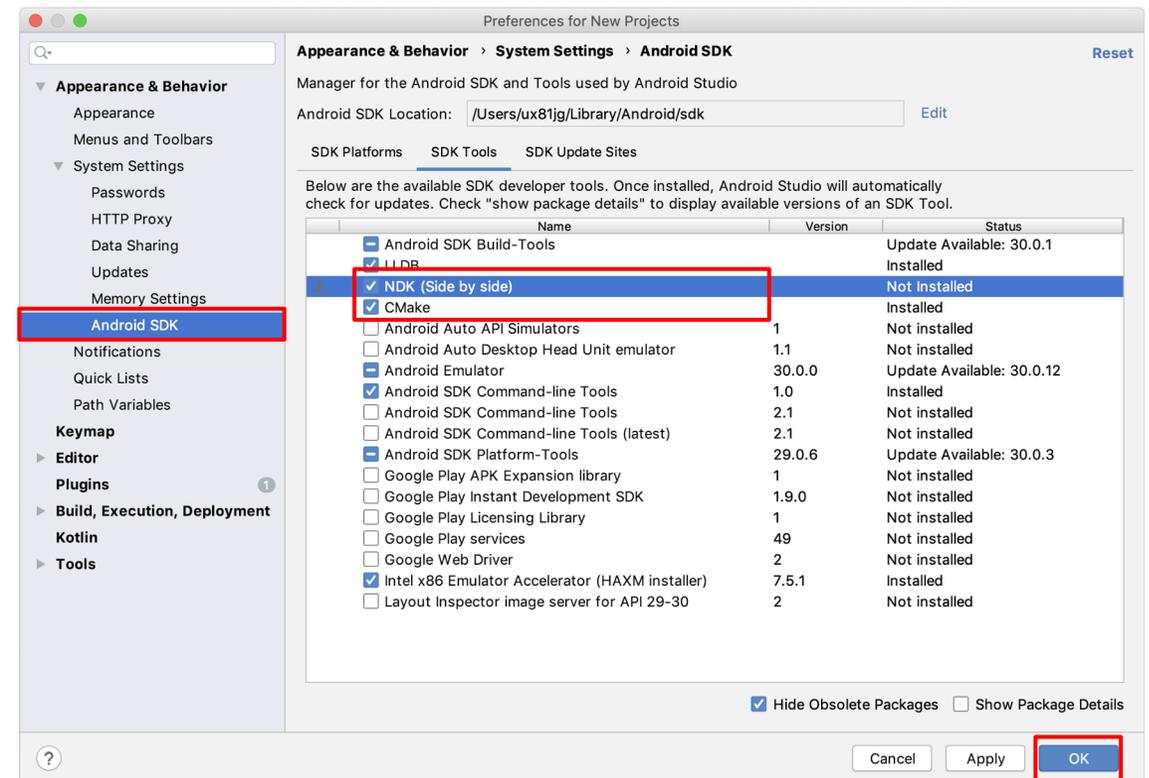
Lesson 5: Installing NDK

- Open Android Studio
- Click the “Configure” option
- Select SDK Manager from the options



Lesson 5: Installing NDK

- Click on “SDK Tool” tab
- Check the “NDK” check box
- Check the “CMake” checkbox
- Press the “Ok” button to install



Lesson 5: Wrap-up

- Download Android Studio for your Operating System
- Install Android Studio

Lesson 6: Import "Vulnerable Project" in Android Studio

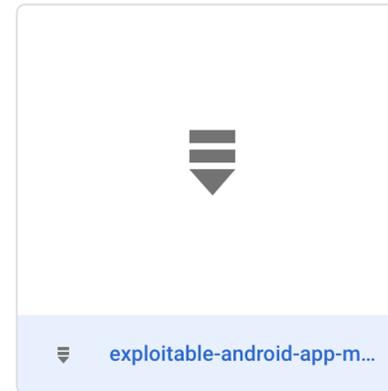
- Download "vulnerable project"
- <https://drive.google.com/file/d/1V7-JAMxH2u5c6nfHxtcYfgcCuI2jCsJK/view?usp=sharing>
- Import project

Lesson 6: Download “Vulnerable Project”

- Download the “vulnerable project”
- <https://drive.google.com/file/d/1V7-JAMxH2u5c6nfHxtcYfgcCul2jCsJK/view?usp=sharing>
- Unzip the project

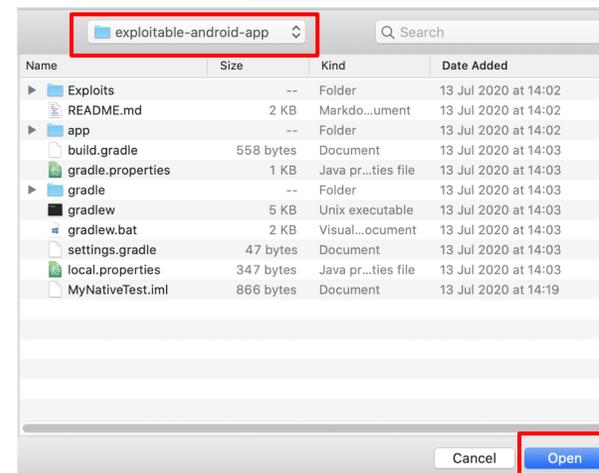
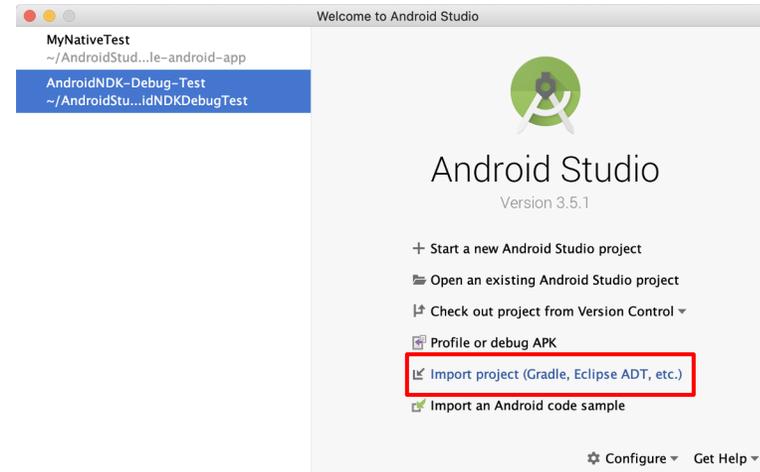


Files



Lesson 6: Import "Vulnerable Project" in Android Studio

- Open Android Studio
- Click "Import Project"



Lesson 6: Wrap-up

- Download Android Studio for your Operating System
- Install Android Studio

Lesson 7: Download and run ARM Android VM

- Download ARM Android VM
- Running the VM

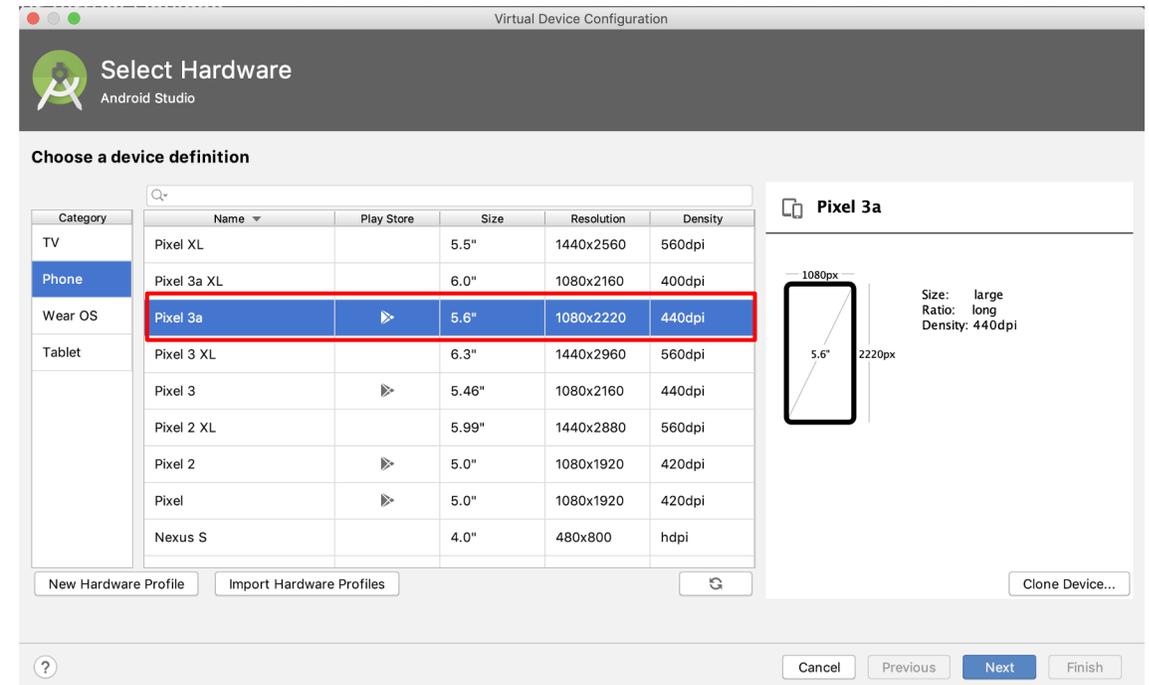
Lesson 7: Download and run ARM Android VM

- Open Android Studio
- Select the newly imported project and open
- On the top choose the “No devices” and select “Open AVD Manager”



Lesson 7: Download and run ARM Android VM

- Select the Hardware we would like to run
- Choose Pixel 3a as shown in picture
- Click next



Lesson 7: Download and run ARM Android VM

- Select the correct image
 - Nougat
 - API Level 25
 - Armabi-v7a
 - Android 7.1.1
- Click “Download”
- Click next after download

Virtual Device Configuration

System Image
Android Studio

Select a system image

Recommended x86 Images **Other Images**

Release Name	API Level	ABI	Target
Nougat Download	25	arm64-v8a	Android 7.1.1 (Google APIs)
Nougat Download	25	armeabi-v7a	Android 7.1.1 (Google APIs)
Nougat Download	24	arm64-v8a	Android 7.0 (Google APIs)
Nougat Download	24	arm64-v8a	Android 7.0
Nougat Download	24	armeabi-v7a	Android 7.0
Marshmallow Download	23	armeabi-v7a	Android 6.0 (Google APIs)
Marshmallow Download	23	armeabi-v7a	Android 6.0
Lollipop Download	22	armeabi-v7a	Android 5.1 (Google APIs)
Lollipop Download	22	armeabi-v7a	Android 5.1
Lollipop Download	21	armeabi-v7a	Android 5.0 (Google APIs)
Lollipop Download	21	armeabi-v7a	Android 5.0
KitKat Download	19	armeabi	Android 4.4 (Google APIs)

Nougat

 API Level
25
Android
7.1.1
Google Inc.
System Image
armeabi-v7a

Recommendation
Consider using an x86 system image on an x86 host for better emulation performance.

Questions on API level?
[See the API level distribution chart](#)

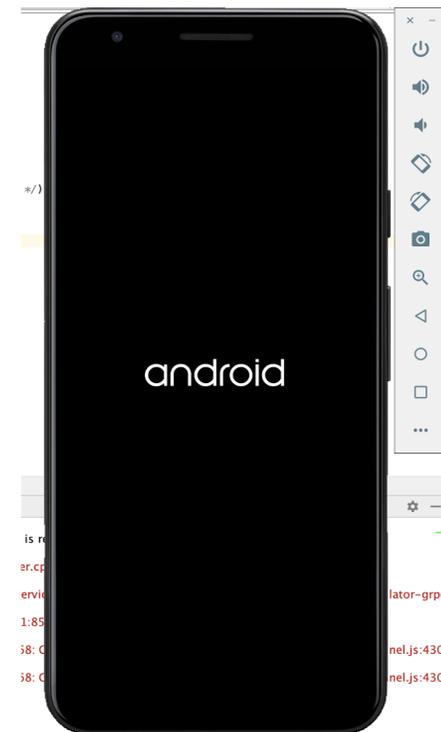
A system image must be selected to continue.

Cancel Previous Next Finish

Lesson 7: Download and run ARM Android VM

- Running the VM is easy, click the play button in the ribbon
- This should launch the VM

NOTE: It will take some time for the VM to boot up and install the app

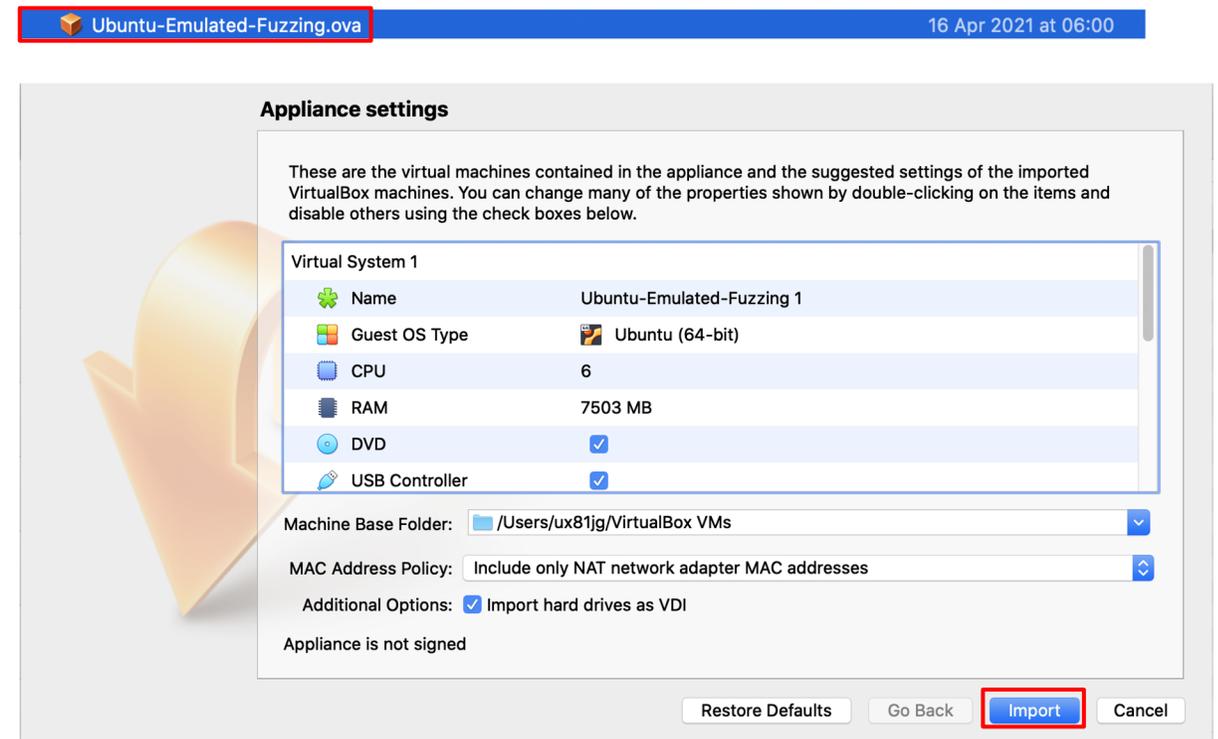


Lesson 7: Wrap-up

- Download Android Studio for your Operating System
- Install Android Studio

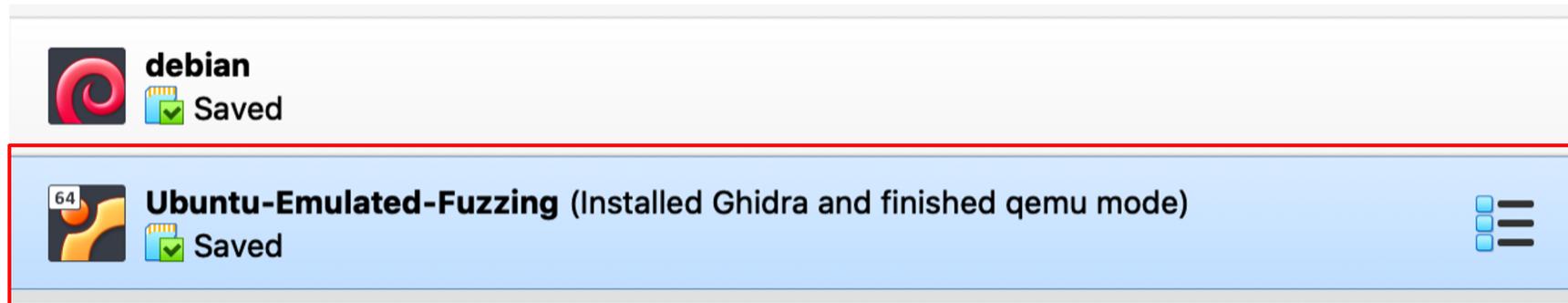
Lesson 8: Connect to phone from Ubuntu VM

- Download the training VM
- <https://drive.google.com/file/d/1AqBUVohKMxI22fMnNe4I1Fq7j4DQLzb7/view?usp=sharing>
- Import the file into VirtualBox by double clicking on the file
- Press the “Import” button to import



Lesson 8: Connect to phone from Ubuntu VM

- Open VirtualBox
- Double click on the imported machines to run it

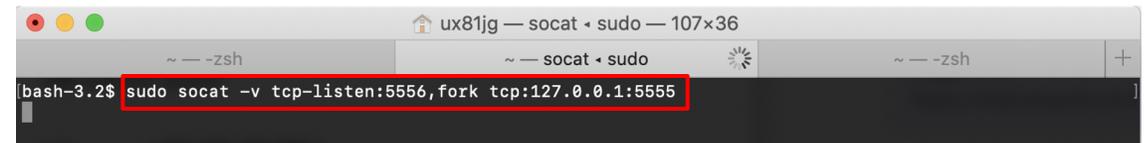


Lesson 8: Connect to phone from Ubuntu VM

- ADB connection forward
- Android App VM Connection forward
- Android Device GDB Server forward

Lesson 8: ADB connection forward

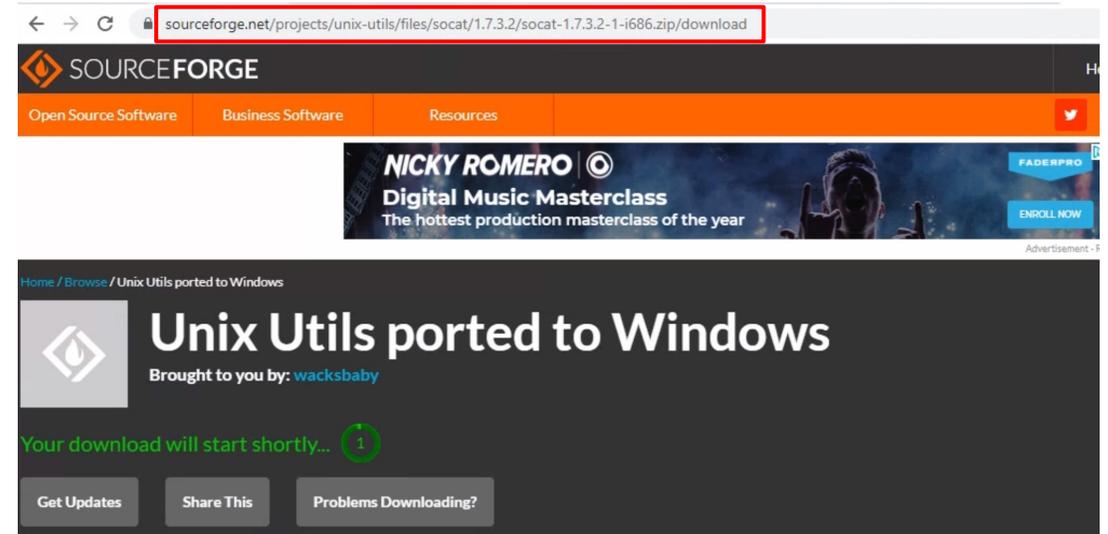
- Download SOCAT
 - for Linux(Ubuntu)
 - apt install socat
 - for Mac:
 - brew install socat
 - For Windows
 - see next slides
- Open terminal and type:
 - sudo socat -v tcp-listen:5556,fork tcp:127.0.0.1:5555
- This will forward all ADB connection on localhost to all interfaces on port 5556



```
ux81jg — socat • sudo — 107x36
~ — -zsh
~ — socat • sudo
~ — -zsh
[bash-3.2$ sudo socat -v tcp-listen:5556,fork tcp:127.0.0.1:5555
```

Lesson 8: ADB connection forward for Windows

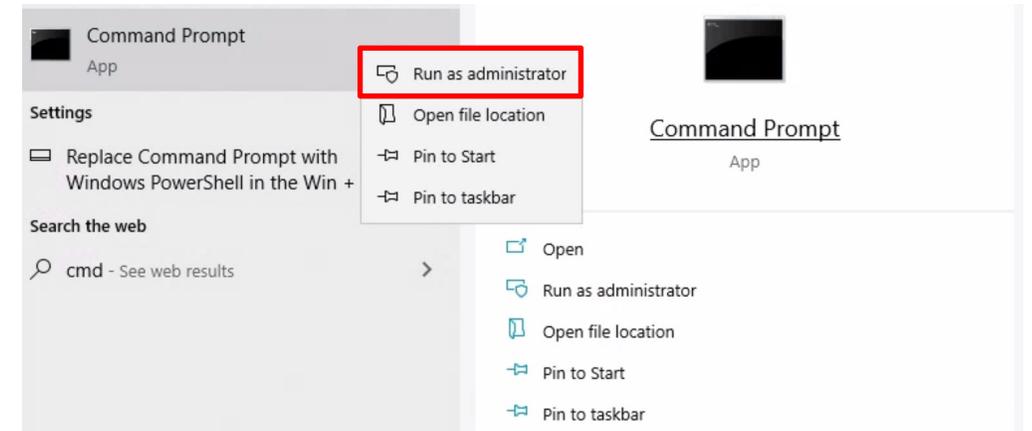
- Download the SOCAT using below link
 - <https://sourceforge.net/projects/unix-utils/files/socat/1.7.3.2/socat-1.7.3.2-1-i686.zip/download>
- Extract the downloaded zip file



NOTE: This step is only required if you use Windows as your base machine!

Lesson 8: ADB connection forward for Windows

- Open command prompt run as administrator
- Go to unzipped location and type:
• `socat.exe -v tcp-listen:5556,fork tcp:127.0.0.1:5555`
- This will forward all ADB connection on localhost to all interfaces on port 5556



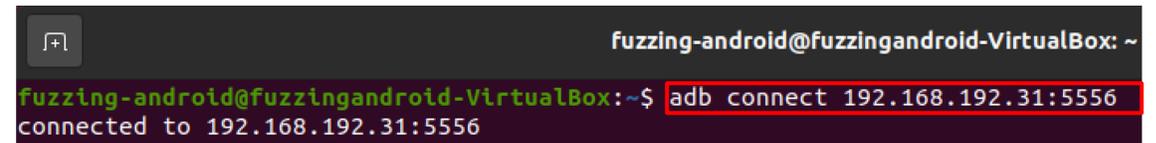
```
Administrator: Command Prompt - socat.exe -v tcp-listen:5556,fork tcp:127.0.0.1:5555
Microsoft Windows [Version 10.0.19043.1052]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Users\gopi\Downloads\socat-1.7.3.2-1-i686
C:\Users\gopi\Downloads\socat-1.7.3.2-1-i686>socat.exe -v tcp-listen:5556,fork tcp:127.0.0.1:5555
```

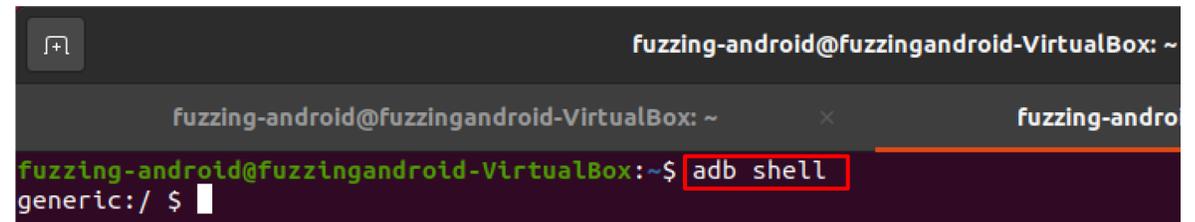
NOTE: This step is only required if you use Windows as your base machine!

Lesson 8: ADB connection forward

- Open you VirtualBox VM
- In the Ubuntu VM open a terminal
- Connect to you host adb from the VM by typing the following command
- **“adb connect 192.168.192.31:5556”**
- **NOTE: Your ip address is your HOST machine IP**
- Now type “adb shell” and you should be dropped into a shell of the Android VM



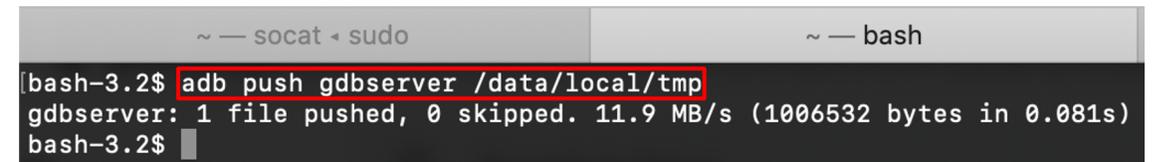
```
fuzzing-android@fuzzingandroid-VirtualBox: ~  
fuzzing-android@fuzzingandroid-VirtualBox:~$ adb connect 192.168.192.31:5556  
connected to 192.168.192.31:5556
```



```
fuzzing-android@fuzzingandroid-VirtualBox: ~  
fuzzing-android@fuzzingandroid-VirtualBox: ~  
fuzzing-android@fuzzingandroid-VirtualBox:~$ adb shell  
generic:/ $
```

Lesson 8: Download and Upload GDBServer to device

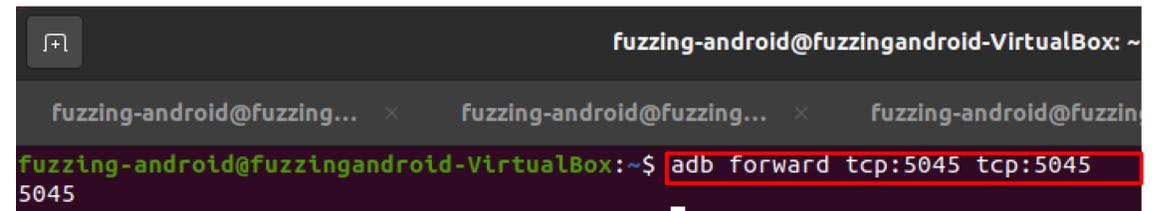
- Download the precompiled version of GDBServer
- https://drive.google.com/file/d/19QLHCRnMh16AnJcmM_iYQW87xlgh9bPa/view?usp=sharing
- Push the file to the device with the following command
- “adb push gdbserver /data/local/tmp”



```
~ — socat ◀ sudo          ~ — bash
[bash-3.2$ adb push gdbserver /data/local/tmp
gdbserver: 1 file pushed, 0 skipped. 11.9 MB/s (1006532 bytes in 0.081s)
bash-3.2$ █
```

Lesson 8: Download and Upload GDBServer to device

- From the connect adb VM now lets forward the gdbserver port to our Ubuntu VM
- Type the following command
- “Adb forward tcp:5045 tcp:5045”



```
fuzzing-android@fuzzingandroid-VirtualBox: ~  
fuzzing-android@fuzzing... x fuzzing-android@fuzzing... x fuzzing-android@fuzzin  
fuzzing-android@fuzzingandroid-VirtualBox:~$ adb forward tcp:5045 tcp:5045  
5045
```

Lesson 8: Download and Upload GDBServer to device

- First start the app then “adb shell” into the Android VM Device
- Type “su” to become root
- Start our app with the following command
- “am start com.example.mynativetest/.MainActivity”
- Go to /data/local/tmp
- There we have our gdbserver
- Make it executable with chmod
- “chmod +x gdbserver”
- Type “ps | grep mynative” to find your process pid
- Then run gdbserver and attach to our PID
- ./gdbserver. :5045 --attach PID

```
fuzzing-android@fuzzingandroid-VirtualBox:~$ adb shell
generic:/ $ su
generic:/ # am start com.example.mynativetest/.MainActivity
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.ca
generic:/ #
```

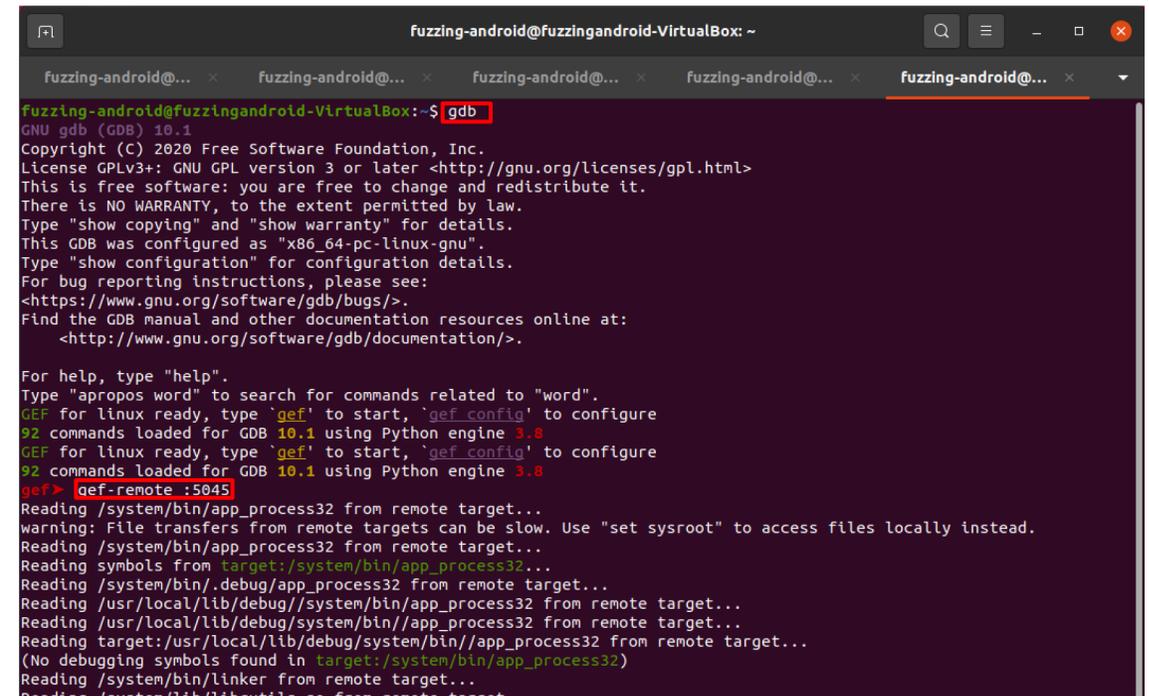
```
generic:/ # cd /data/local/tmp/
generic:/data/local/tmp # ls
gdbserver
generic:/data/local/tmp # chmod +x gdbserver
generic:/data/local/tmp #
```

```
fuzzing-android@fuzzingandroid-VirtualBox: ~
fuzzing-android@fuzzing... x fuzzing-android@fuzzing... x fuzzing-android@fuzzing... x
127|generic:/data/local/tmp # ps | grep mynative
u0_a84 7287 5692 1727368 147968 Sys_epoll_aaec350 S com.example.mynativetest
```

```
fuzzing-android@fuzzingandroid-VirtualBox: ~
fuzzing-android@fuzzing... x fuzzing-android@fuzzing... x fuzzing-android@fuzzing... x
generic:/data/local/tmp # ./gdbserver :5045 --attach 7287
Attached; pid = 7287
Listening on port 5045
```

Lesson 8: Download and Upload GDBServer to device

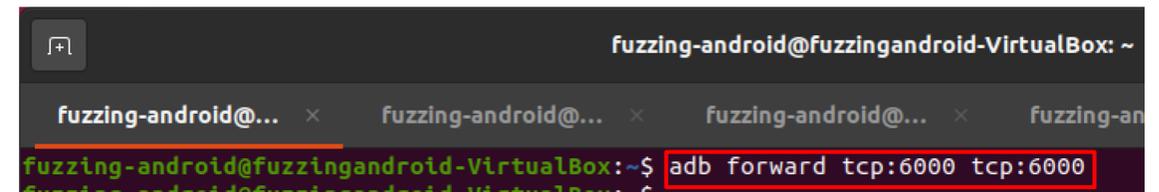
- Now open another terminal and run gdb
- In gdb run “gef-remote :5045”
- You should see the same as the image on the right



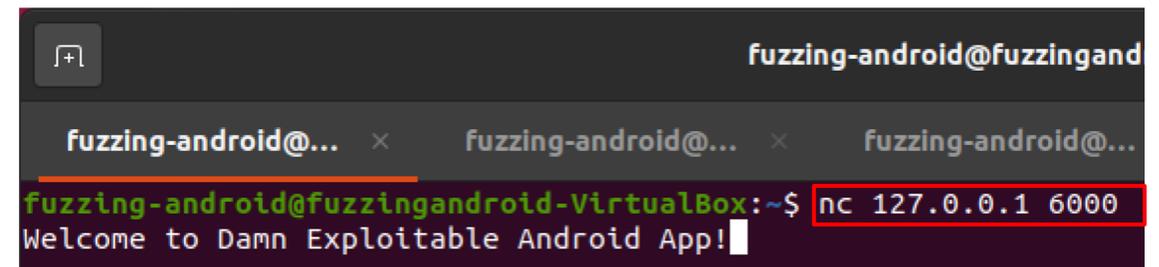
```
fuzzing-android@fuzzingandroid-VirtualBox: ~  
fuzzing-android@... x fuzzing-android@... x fuzzing-android@... x fuzzing-android@... x fuzzing-android@... x  
fuzzing-android@fuzzingandroid-VirtualBox:~$ gdb  
GNU gdb (GDB) 10.1  
Copyright (C) 2020 Free Software Foundation, Inc.  
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law.  
Type "show copying" and "show warranty" for details.  
This GDB was configured as "x86_64-pc-linux-gnu".  
Type "show configuration" for configuration details.  
For bug reporting instructions, please see:  
<https://www.gnu.org/software/gdb/bugs/>.  
Find the GDB manual and other documentation resources online at:  
  <http://www.gnu.org/software/gdb/documentation/>.  
  
For help, type "help".  
Type "apropos word" to search for commands related to "word".  
GEF for linux ready, type `gef' to start, `gef config' to configure  
92 commands loaded for GDB 10.1 using Python engine 3.8  
GEF for linux ready, type `gef' to start, `gef config' to configure  
92 commands loaded for GDB 10.1 using Python engine 3.8  
gef> gef-remote :5045  
Reading /system/bin/app_process32 from remote target...  
warning: File transfers from remote targets can be slow. Use "set sysroot" to access files locally instead.  
Reading /system/bin/app_process32 from remote target...  
Reading symbols from target:/system/bin/app_process32...  
Reading /system/bin/.debug/app_process32 from remote target...  
Reading /usr/local/lib/debug/system/bin/app_process32 from remote target...  
Reading /usr/local/lib/debug/system/bin//app_process32 from remote target...  
Reading target:/usr/local/lib/debug/system/bin//app_process32 from remote target...  
(No debugging symbols found in target:/system/bin/app_process32)  
Reading /system/bin/linker from remote target...  
Reading /system/lib/libutils.so from remote target...
```

Lesson 8: Forward Vulnerable App port to Ubuntu VM

- Forward app listening port to Ubuntu VM local loopback
- Now open another terminal and type
 - “adb forward tcp:6000 tcp:6000”
- Now connect from the Ubuntu VM to test
- “nc 127.0.0.1 6000”



```
fuzzing-android@fuzzingandroid-VirtualBox: ~  
fuzzing-android@... x fuzzing-android@... x fuzzing-android@... x fuzzing-an  
fuzzing-android@fuzzingandroid-VirtualBox:~$ adb forward tcp:6000 tcp:6000
```



```
fuzzing-android@fuzzingand  
fuzzing-android@... x fuzzing-android@... x fuzzing-android@...  
fuzzing-android@fuzzingandroid-VirtualBox:~$ nc 127.0.0.1 6000  
Welcome to Damn Exploitable Android App!
```

Lesson 8: Wrap Up

- Forwarded adb from host to Ubuntu VM
- Downloaded precompiled GDBServer
- Forward GDBServer port to Ubuntu VM
- Pushed GDBServer to Android VM
- Attached GDBServer to the running Application
- Forward Vulnerable App port to Ubuntu VM
- Test connection to vulnerable application

Lesson 9: Corellium Lab Setup

- Login into Corellium portals using the below link:

<https://afe.enterprise.corellium.com/login>

- You will be provided access through our Slack Group

Lesson 9: Corellium Lab Setup

- Once you logged into the Corellium portal, turn on the device

The screenshot shows the Corellium portal interface. At the top left is the Corellium logo and a search bar labeled 'Find Device'. On the top right is a 'DEVICES' tab. The main heading is 'Devices' with a 'CREATE DEVICE' button. Below this is a search bar for 'ALL PROJECTS & DEVICES'. There are two project cards:

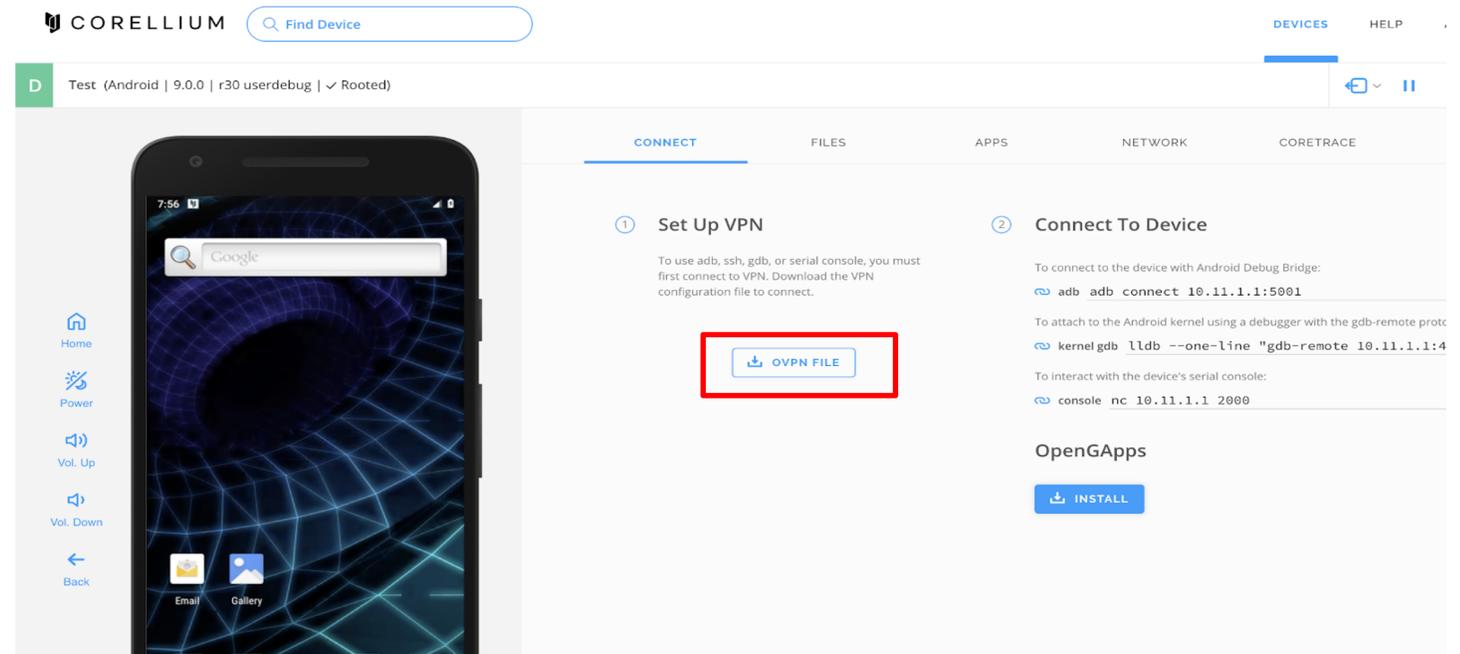
- AFE**: 0 unused CPU cores, 0 unused stored device slots. Message: "No devices created yet. Ask your organization to assign some CPU cores to this project."
- Default Project**: 6 unused CPU cores, 14 unused stored device slots. Contains a table:

	MODEL	VERSION	CREATED ON	
Test	Android	9.0.0	Jun 24, 2021	...

The dropdown menu for the 'Test' device shows two options: 'Turn On' (highlighted with a red box) and 'Delete'.

Lesson 9: Corellium Lab Setup

- Download ovpn file from your fuzzing VM



The screenshot shows the Corellium web interface. At the top, there is a search bar labeled 'Find Device' and a 'CORELLIUM' logo. Below the search bar, there is a header for the current device: 'Test (Android | 9.0.0 | r30 userdebug | ✓ Rooted)'. The main content area is divided into several tabs: 'CONNECT', 'FILES', 'APPS', 'NETWORK', and 'CORETRACE'. The 'CONNECT' tab is active and contains two main sections: '1 Set Up VPN' and '2 Connect To Device'. The 'Set Up VPN' section includes instructions: 'To use adb, ssh, gdb, or serial console, you must first connect to VPN. Download the VPN configuration file to connect.' Below this text is a red-bordered button labeled 'OVPN FILE'. The 'Connect To Device' section includes instructions: 'To connect to the device with Android Debug Bridge:' followed by the command 'adb adb connect 10.11.1.1:5001', 'To attach to the Android kernel using a debugger with the gdb-remote protocol:' followed by the command 'kernel gdb lldb --one-line "gdb-remote 10.11.1.1:4', and 'To interact with the device's serial console:' followed by the command 'console nc 10.11.1.1 2000'. Below these instructions is a section for 'OpenGApps' with an 'INSTALL' button. On the left side of the interface, there is a vertical navigation menu with icons for Home, Power, Vol. Up, Vol. Down, and Back. In the center, there is a large image of a smartphone displaying a Google search bar and icons for Email and Gallery.

```
fuzzing-android@fuzzingandroid-VirtualBox:~/Downloads$ ls  
corellium.ovpn
```

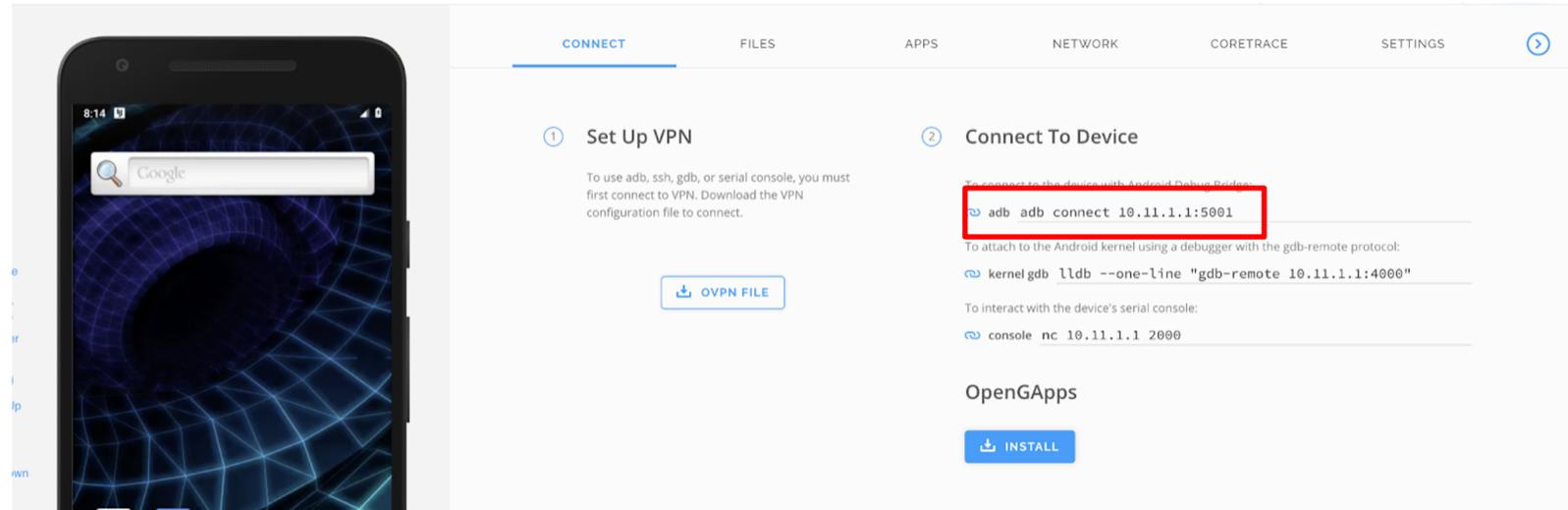
Lesson 9: Corellium Lab Setup

- Use openvpn to connect to the Corellium lab device
 - `sudo openvpn --config corellium.ovpn`

```
fuzzing-android@fuzzingandroid-VirtualBox:~/Downloads$ sudo openvpn --config corellium.ovpn
[sudo] password for fuzzing-android:
Sat Jul 3 13:43:28 2021 OpenVPN 2.4.7 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on Apr 27 2021
Sat Jul 3 13:43:28 2021 library versions: OpenSSL 1.1.1f 31 Mar 2020, LZO 2.10
Sat Jul 3 13:43:28 2021 TCP/UDP: Preserving recently used remote address: [AF_INET]18.218.250.49:16021
Sat Jul 3 13:43:28 2021 UDP link local: (not bound)
Sat Jul 3 13:43:28 2021 UDP link remote: [AF_INET]18.218.250.49:16021
Sat Jul 3 13:43:29 2021 [openvpn-ee351377-19a5-44af-b01f-cbcfa04cca7f] Peer Connection Initiated with [AF_INET]18.218.250.49:16021
Sat Jul 3 13:43:30 2021 TUN/TAP device tap0 opened
Sat Jul 3 13:43:30 2021 /sbin/ip link set dev tap0 up mtu 1500
Sat Jul 3 13:43:30 2021 /sbin/ip addr add dev tap0 10.11.3.3/22 broadcast 10.11.3.255
Sat Jul 3 13:43:30 2021 WARNING: this configuration may cache passwords in memory -- use the auth-nocache option to prevent this
Sat Jul 3 13:43:30 2021 Initialization Sequence Completed
```

Lesson 9: Corellium Lab Setup

- To connect with the device, copy the command from the Corellium interface



```
fuzzing-android@fuzzingandroid-VirtualBox:~$ adb connect 10.11.1.1:5001
connected to 10.11.1.1:5001
fuzzing-android@fuzzingandroid-VirtualBox:~$ adb shell
shell:/$
```

Summary of Lab Setup

- Joined the training Slack Group
- Installed Virtual Box and Guest Additions
- Imported Ubuntu/Fuzzing VM
- Installed Android Studio
- Installed NDK
- Imported "Vulnerable Project" in Android Studio
- Downloaded and tested ARM Android VM
- Connected the phone from Ubuntu VM
- Lesson 9: Corellium Lab Setup

Summary of Lab Setup

- For any other questions please join the Slack group!