MOBILE PENTESTING

BY JAMES ROBERSON

PROFESSIONAL | CYBERSECURITY | MARCH 17, 2024

WHAT HAPPENED?

In the task of the day, we are pentesting an Android device! By using ADB (Android Debug Bridge) to gain access to the mobile device and collect information, I then install Damn Insecure and Vulnerable App (DIVA) APK and complete two challenges contained in the app. ADB is a versatile tool that allows pentesting physically on a device or virtually connected to another OS.

- Install the Android VM.
- Discover IP with network mapper and begin establishing which services are on which port.
- Complete two Diva challeneges

PROOF:

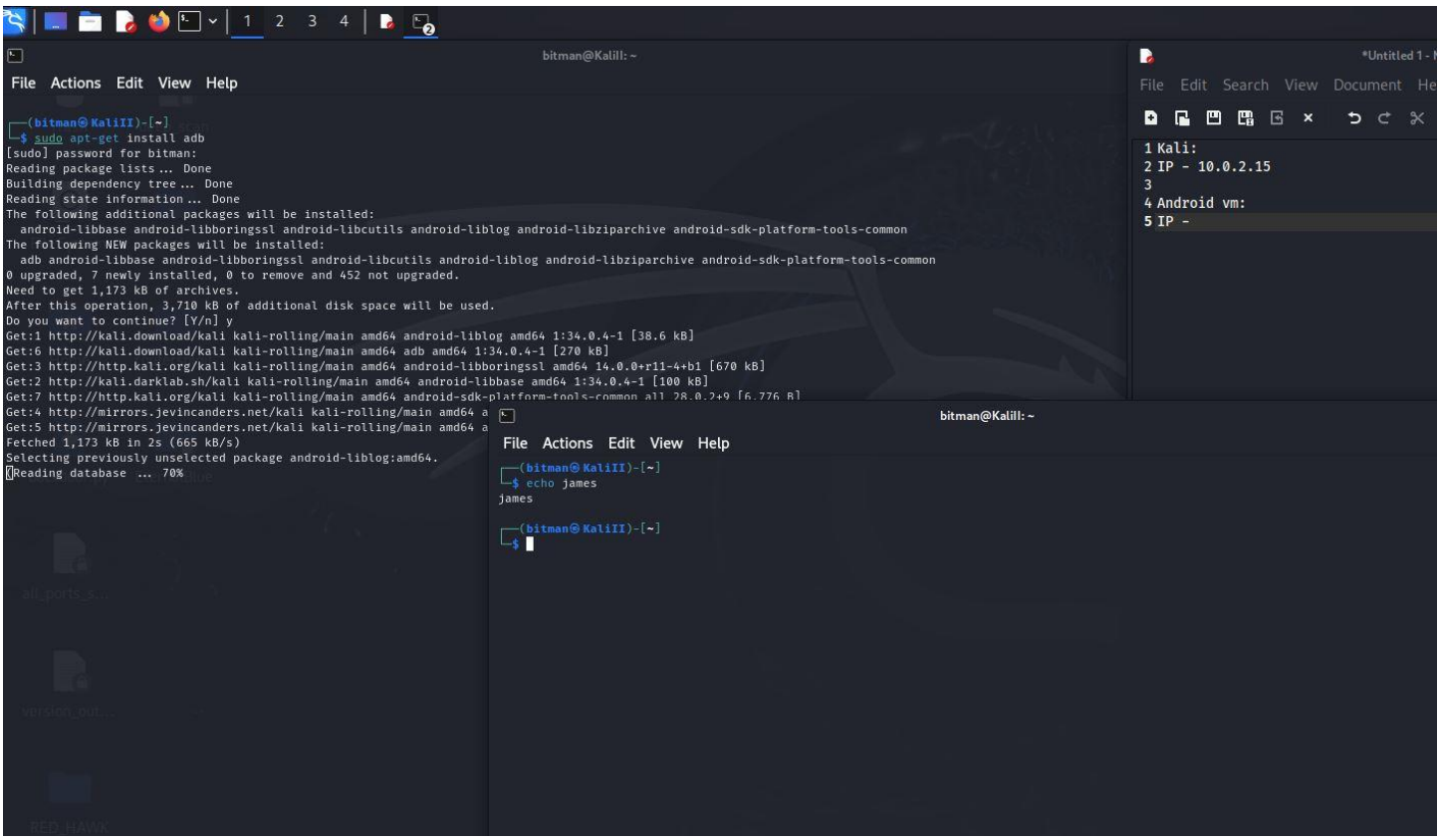

Figure 1. First step, host discovery. But first I had to install ADB.

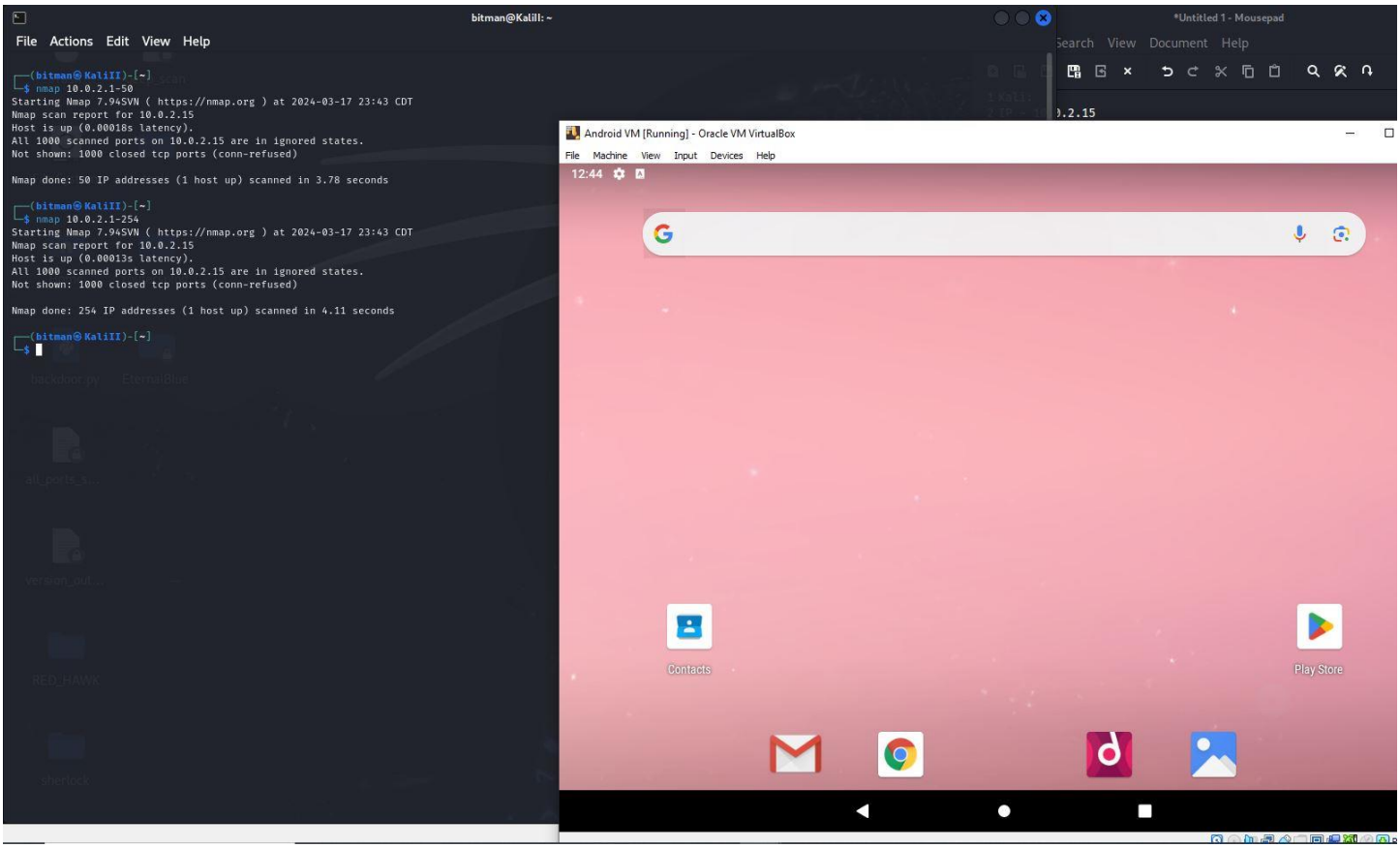

Figure 3. Both are on the "NAT" configuration in the network but let me try our NAT network.

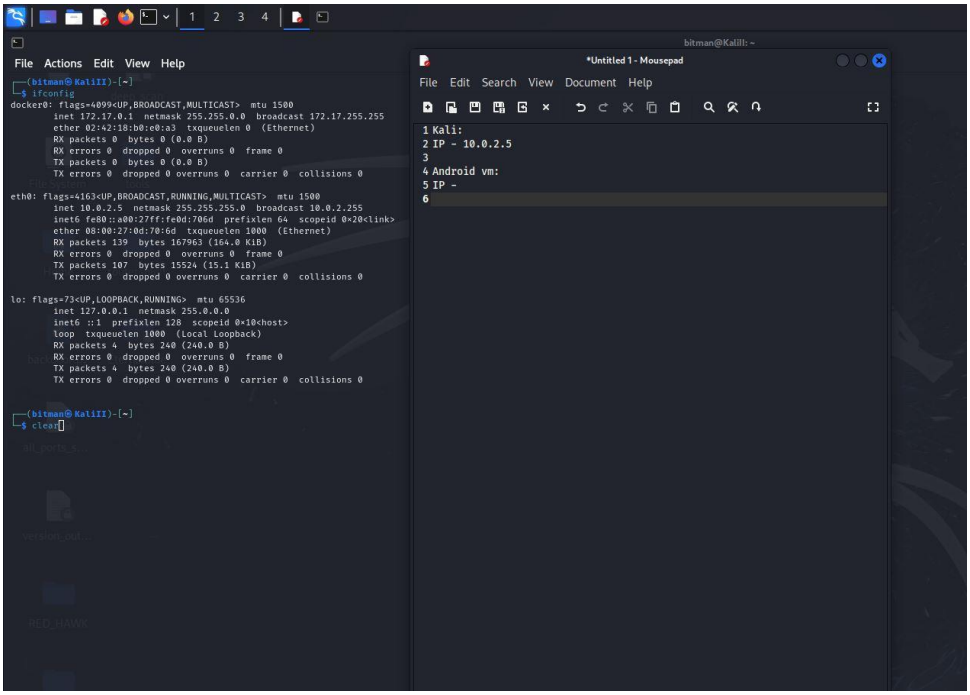

Figure 4. Once I set both VMs to be on the NAT network, I rebooted the machines and discovered the IP for both.

```
Nmap done: 254 IP addresses (3 hosts up) scanned in 6.11 seconds

┌──(bitman㉿KaliII)-[~]
└─$ git clone https://github.com/0xArab/diva-apk-file.git
Cloning into 'diva-apk-file'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 14 (delta 5), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (14/14), 1.17 MiB | 4.83 MiB/s, done.
Resolving deltas: 100% (5/5), done.

┌──(bitman㉿KaliII)-[~]
└─$ ls
Desktop  diva-apk-file  Documents  Downloads  Music  Pictures  Public  Templates  Videos

┌──(bitman㉿KaliII)-[~]
└─$ mv diva-apk-file /home/bitman/Documents/CodingDojo

┌──(bitman㉿KaliII)-[~]
└─$ ls
```

Figure 5. We have to use the DIVA apk file imported from github. We're using this because DIVA is a pentesting site for newly instated cybersecurity professionals. In this particular endeavor, we want to exploit the android device firstly with a msfvenom payload (10-minute tactical) and then use the DIVA application to complete a challenge.

```
  extracting: DIVA/resources.arsc
   inflating: DIVA/classes.dex
   inflating: DIVA/lib/mips/libdivajni.so
   inflating: DIVA/lib/armeabi-v7a/libdivajni.so
   inflating: DIVA/lib/x86_64/libdivajni.so
   inflating: DIVA/lib/x86/libdivajni.so
   inflating: DIVA/lib/arm64-v8a/libdivajni.so
   inflating: DIVA/lib/mips64/libdivajni.so
   inflating: DIVA/lib/armeabi/libdivajni.so
   inflating: DIVA/META-INF/MANIFEST.MF
   inflating: DIVA/META-INF/CERT.SF
   inflating: DIVA/META-INF/CERT.RSA

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file]
└─$ ls
DIVA  DivaApplication.apk  LICENSE  README.md

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file]
└─$ cd DIVA

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file/DIVA]
└─$ ls
AndroidManifest.xml  classes.dex  lib  META-INF  res  resources.arsc

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file/DIVA]
└─$
```

Figure 6. Next we examine the APKtool. That is done by installing the APKtool into our Host machine, Figure 7. And running the APK file against the APKtool, outputting the results to a file with the same name as the APK file, Figure 8.

```
  inflating: DIVA/lib/arm64-v8a/libdivajni.so
  inflating: DIVA/lib/mips64/libdivajni.so
  inflating: DIVA/lib/armeabi/libdivajni.so
  inflating: DIVA/META-INF/MANIFEST.MF
  inflating: DIVA/META-INF/CERT.SF
  inflating: DIVA/META-INF/CERT.RSA

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file]
└─$ ls
DIVA  DivaApplication.apk  LICENSE  README.md

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file]
└─$ cd DIVA

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file/DIVA]
└─$ ls
AndroidManifest.xml  classes.dex  lib  META-INF  res  resources.arsc

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file/DIVA]
└─$ cd ..

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file]
└─$ ls
DIVA  DivaApplication.apk  LICENSE  README.md

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file]
└─$ apktool
Command 'apktool' not found, but can be installed with:
sudo apt install apktool
Do you want to install it? (N/y)y
sudo apt install apktool
[sudo] password for bitman:
Sorry, try again.
[sudo] password for bitman:
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following additional packages will be installed:
  aapt android-framework-res android-libaapt android-libandroidfw android-libbacktrace android-libutils junit libantlr-java libantlr3-runtime-java libaopalliance-j
  libcommons-io-java libcommons-lang3-java libcommons-text-java liberror-prone-java libgeronimo-annotation-1.3-spec-java libgeronimo-interceptor-3.0-spec-java libg
  libmaven-file-management-java libmaven-jar-plugin-java libmaven-parent-java libmaven-resolver-java libmaven-shared-io-java libmaven-shared-utils-java libmaven3-c
  libplexus-component-annotations-java libplexus-interpolation-java libplexus-io-java libplexus-sec-dispatcher-java libplexus-utils2-java libsisu-inject-java libsi
  libwagon-provider-api-java libxmlunit-java libxpp3-java libxz-java libyaml-snake-java
Suggested packages:
  junit-doc libatinject-jsr330-api-java-doc libel-api-java libasm-java libcommons-io-java-doc libcglib-java libjcommander-java-doc libjsr305-java-doc libmaven-file-
  testng liblog4j1.2-java
The following NEW packages will be installed:
  aapt android-framework-res android-libaapt android-libandroidfw android-libbacktrace android-libutils apktool junit libantlr-java libantlr3-runtime-java libaopal
  libcommons-io-java libcommons-lang3-java libcommons-text-java liberror-prone-java libgeronimo-annotation-1.3-spec-java libgeronimo-interceptor-3.0-spec-java libg
  libmaven-file-management-java libmaven-jar-plugin-java libmaven-parent-java libmaven-resolver-java libmaven-shared-io-java libmaven-shared-utils-java libmaven3-c
  libplexus-component-annotations-java libplexus-interpolation-java libplexus-io-java libplexus-sec-dispatcher-java libplexus-utils2-java libsisu-inject-java libsi
  libwagon-provider-api-java libxmlunit-java libxpp3-java libxz-java libyaml-snake-java
0 upgraded, 54 newly installed, 0 to remove and 452 not upgraded.
Need to get 33.6 MB of archives.
After this operation, 81.3 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Figure 7…

```
File  Actions  Edit  View  Help

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file]
└─$ ls
DIVA  DivaApplication.apk  LICENSE  README.md

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file]
└─$ apktool d DivaApplication.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.7.0-dirty on DivaApplication.apk
I: Loading resource table ...
I: Decoding AndroidManifest.xml with resources ...
I: Loading resource table from file: /home/bitman/.local/share/apktool/framework/1.apk
I: Regular manifest package ...
I: Decoding file-resources ...
I: Decoding values */* XMLs ...
I: Baksmaling classes.dex ...
I: Copying assets and libs ...
I: Copying unknown files ...
I: Copying original files ...

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file]
└─$ ls
DIVA  DivaApplication  DivaApplication.apk  LICENSE  README.md

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file]
└─$
```

Figure 8…now we have the /DivaApplication/ directory.

```
DIVA  DivaApplication  DivaApplication.apk  LICENSE  README.md

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file]
└─$ cd DIVA

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file/DIVA]
└─$ ls
AndroidManifest.xml  classes.dex  lib  META-INF  res  resources.arsc

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file/DIVA]
└─$ d2j-dex2jar
Command 'd2j-dex2jar' not found, but can be installed with:
sudo apt install dex2jar
Do you want to install it? (N/y)y
sudo apt install dex2jar
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following NEW packages will be installed:
  dex2jar
0 upgraded, 1 newly installed, 0 to remove and 452 not upgraded.
Need to get 4,985 kB of archives.
After this operation, 6,083 kB of additional disk space will be used.
Get:1 http://http.kali.org/kali kali-rolling/main amd64 dex2jar all 2.1~nightly-28-0kali2 [4,985 kB]
Fetched 4,985 kB in 1s (6,786 kB/s)
Selecting previously unselected package dex2jar.
(Reading database ... 421839 files and directories currently installed.)
Preparing to unpack .../dex2jar_2.1~nightly-28-0kali2_all.deb ...
Unpacking dex2jar (2.1~nightly-28-0kali2) ...
Setting up dex2jar (2.1~nightly-28-0kali2) ...
Processing triggers for kali-menu (2023.4.7) ...
Scanning processes ...
Scanning linux images ...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/diva-apk-file/DIVA]
└─$
```

Figure 9.

When we CD into the /DIVA/ we run the command d2j-dex2jar; and again if the tool isn't already installed then just do a quick Enter, type 'y', and enter again. When we install the tool and run it against the classes.dex file in the /DIVA/ folder, then we get an output file of JAR. This file we can view in jd-gui.

A tool we will more than likely utilize later. We still have to install the DIVA packages onto the android device, which is accomplished with the help of ADB, Android Debug Bridge.

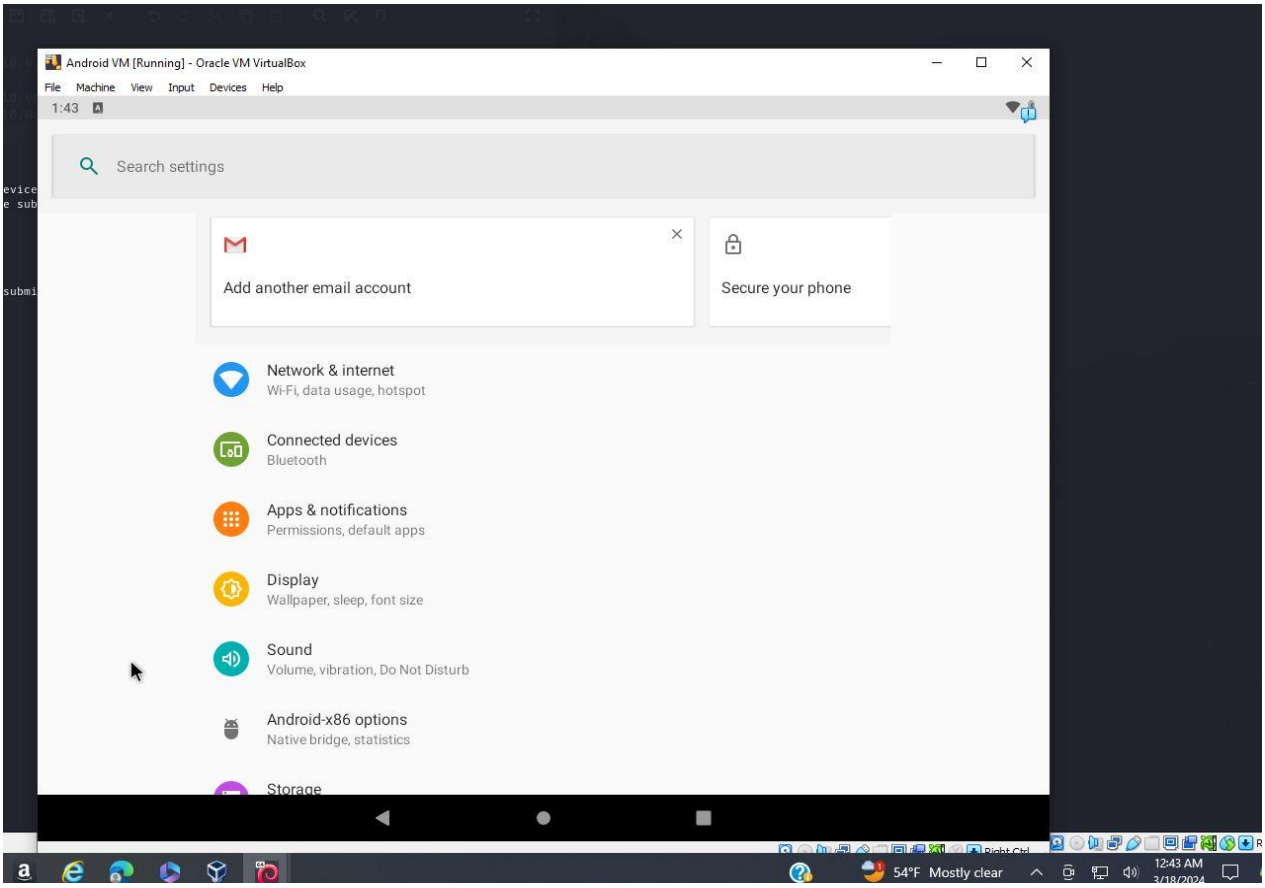

Figure 10. To set up the android device I began with enabling USB debugging on the android device. >> settings>About Device>Tap on build repeatedly to activate developer mode.

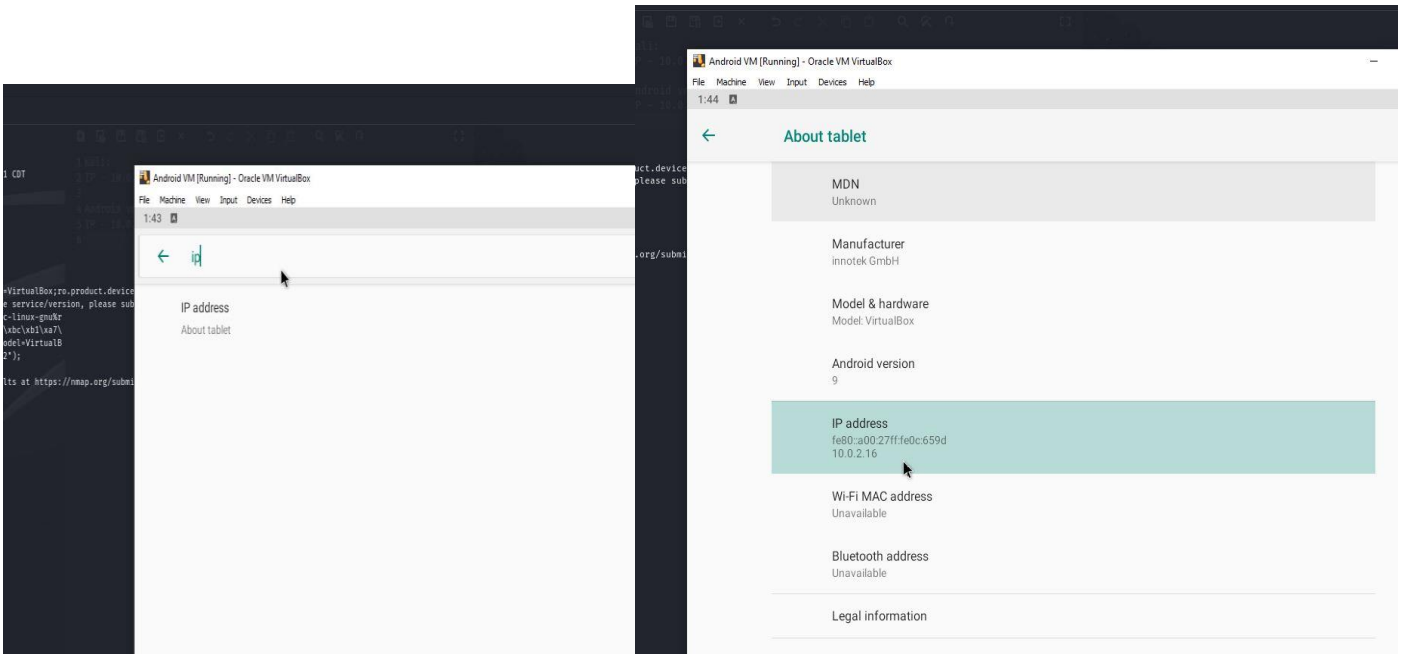

Figure 11. Before activating the developer mode I wanted to check out the IP. This is to confirm that the Android VM's IP is in fact 10.0.2.16.

```
File  Actions  Edit  View  Help
MAIN(); ×   ADB_(); ×

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/mobilePenTest]
└─$ nmap -A -T5 -oN information_schema.txt 10.0.2.16
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-18 00:41 CDT
Nmap scan report for 10.0.2.16
Host is up (0.0095s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT     STATE SERVICE   VERSION
5555/tcp open  freeciv?
| fingerprint-strings:
|   adbConnect:
|     CNXN
|_    device::ro.product.name=android_x86_64;ro.product.model=VirtualBox;ro.product.device=x86_64;
1 service unrecognized despite returning data. If you know the service/version, please submit the
SF-Port5555-TCP:V=7.94SVN%I=7%D=3/18%Time=65F7D40A%P=x86_64-pc-linux-gnu%r
SF:(adbConnect,91,"CNXN\x01\0\0\x01\0\x10\0\0y\0\0\0\xe0,\0\0\xbc\xb1\xa7\
SF:xb1device::ro\.product\.name=android_x86_64;ro\.product\.model=VirtualB
SF:ox;ro\.product\.device=x86_64;features=cmd,stat_v2,shell_v2");

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 91.31 seconds

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/mobilePenTest]
└─$
```

```
bitman@KaliII: ~
File  Actions  Edit  View  Help
┌──(bitman㉿KaliII)-[~]
└─$ echo james
james

┌──(bitman㉿KaliII)-[~]
└─$
```

Figure 12. Next I did a discovery scan to identify the services running. Here is where we see a lead to using ADB as our connecting tool. Let's see just how we can do all of this.

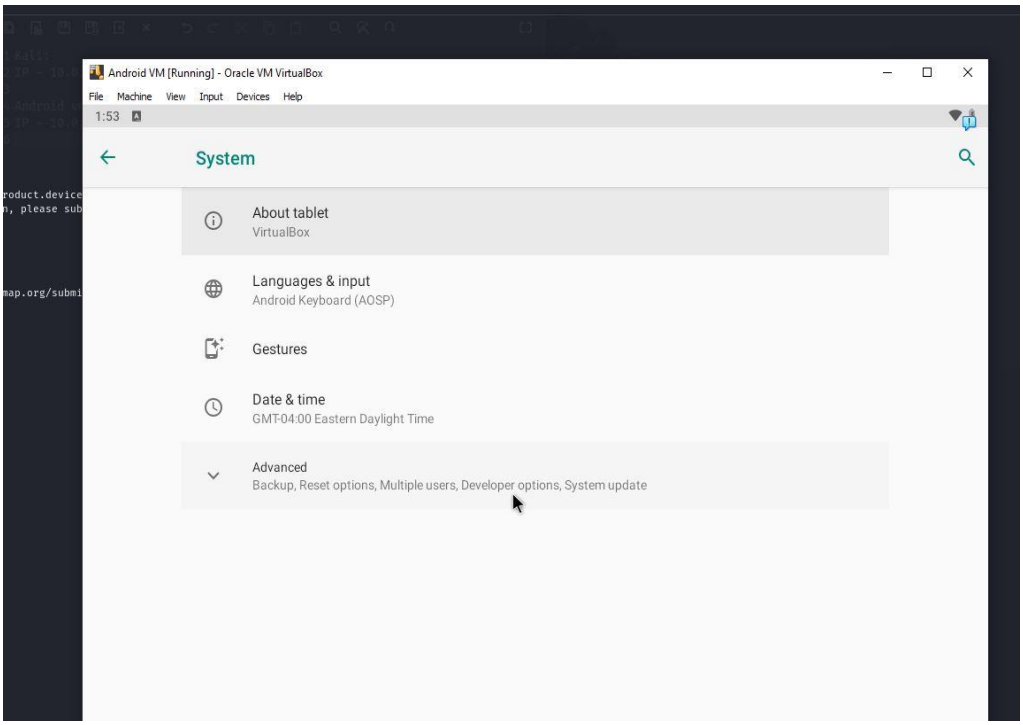

Figure 13…

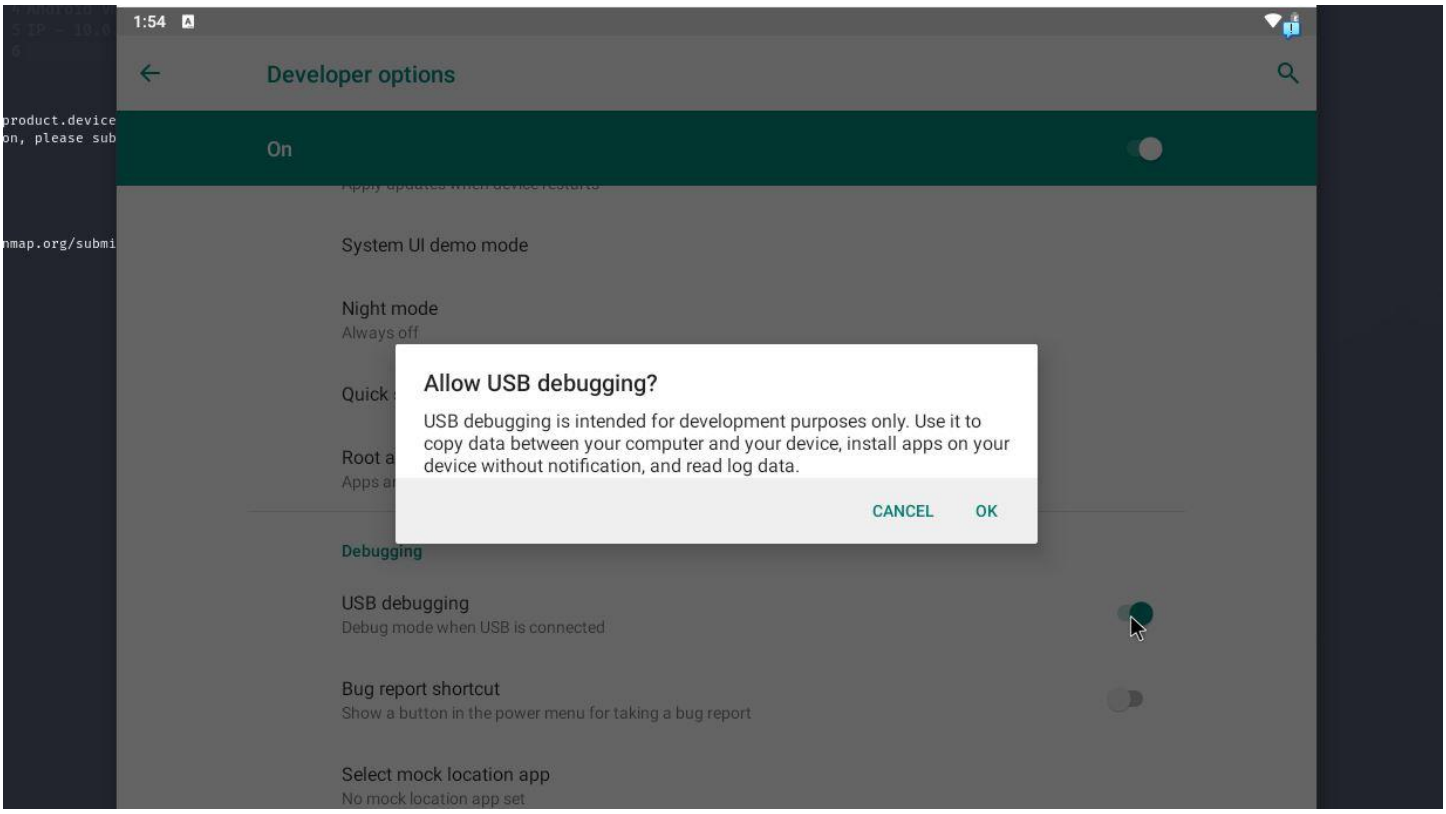

Figure 14. Figure 10 was the first few steps in setting up our Android so that we can connect and exploit it. Figures 13and 14 are showing the end result to Allowing USB debugging.

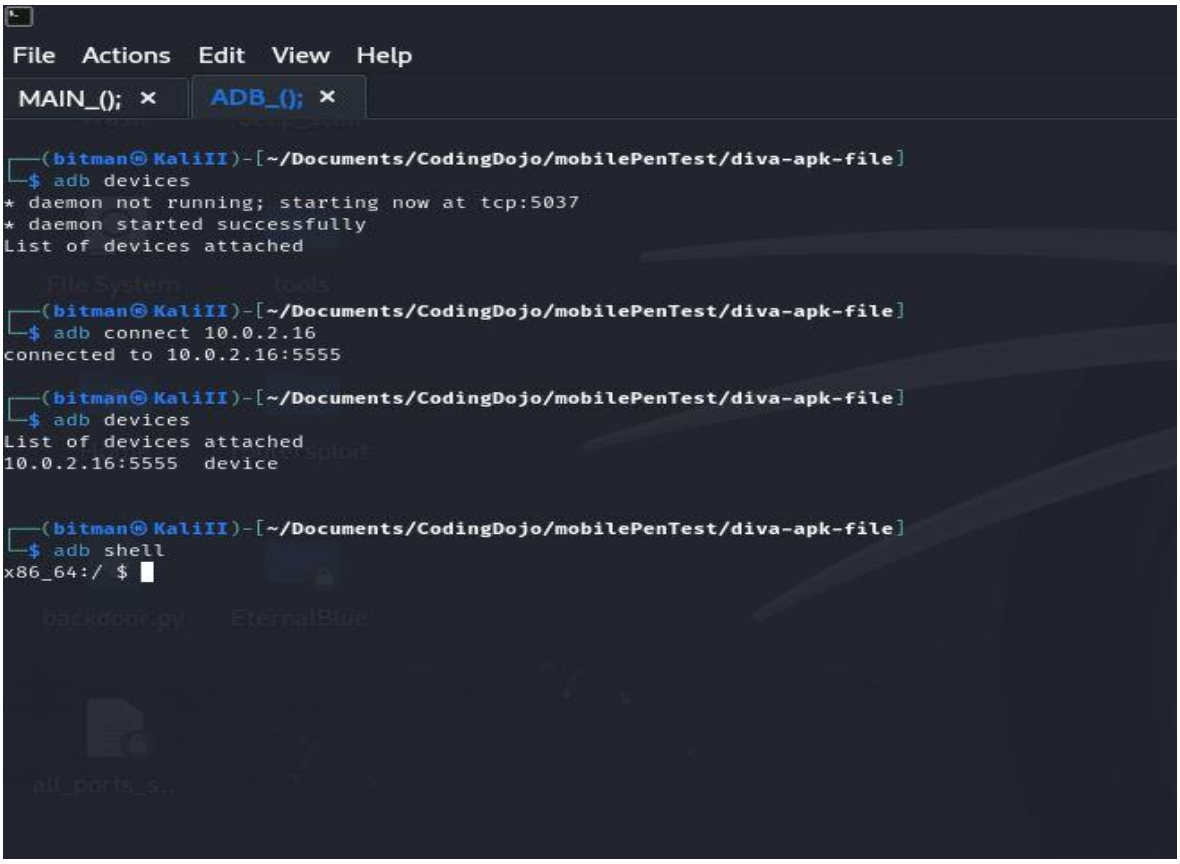

Figure 15. After installing all packages, directories, devices, and tools on to step 3; here, we have to utilize ADB commands to printout certain information Shown in Figure 16.

1. Devie Brand: Android_x86_64
2. Device Model: VirtualBox
3. Device Name: 10.0.2.16:5555 / x86_64
4. List of all installed packages on devices

Installed Packages (4):

Refer to Figure 17. (I just thought to copy and paste here but for the sake of time let's rely on the screenshot)

```
┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/mobilePenTest/diva-apk-file]
└─$ adb shell devices -l
/system/bin/sh: devices: not found

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/mobilePenTest/diva-apk-file]
└─$ adb devices -l
List of devices attached
10.0.2.16:5555         device product:android_x86_64 model:VirtualBox device:x86_64 transport_id:1


┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/mobilePenTest/diva-apk-file]
└─$
```

Figure 16…

```
File  Actions  Edit  View  Help
MAIN_(); ×   ADB_(); ×
┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/mobilePenTest/diva-apk-file]
└─$ adb shell pm list packages
package:com.android.cts.priv.ctsshim
package:com.android.internal.display.cutout.emulation.corner
package:com.google.android.ext.services
package:com.example.android.rssreader
package:com.android.internal.display.cutout.emulation.double
package:com.android.providers.telephony
package:org.android_x86.analytics
package:com.google.android.googlequicksearchbox
package:com.android.providers.calendar
package:com.android.providers.media
package:com.google.android.onetimeinitializer
package:com.google.android.ext.shared
package:com.android.wallpapercropper
package:org.zeroxlab.util.tscal
package:com.android.documentsui
package:com.android.externalstorage
package:com.android.htmlviewer
package:com.android.companiondevicemanager
package:com.android.mms.service
package:com.android.providers.downloads
package:com.android.defcontainer
package:com.android.providers.downloads.ui
package:com.android.vending
package:com.android.pacprocessor
package:com.android.simappdialog
package:com.android.internal.display.cutout.emulation.tall
package:com.android.certinstaller
package:com.android.carrierconfig
package:android
package:com.android.contacts
package:com.android.camera2
package:com.android.egg
package:com.android.mtp
package:com.android.launcher3
package:com.android.backupconfirm
package:com.android.statementservice
package:com.google.android.gm
package:com.android.settings.intelligence
package:com.android.calendar
package:com.android.systemui.theme.dark
package:com.google.android.setupwizard
package:com.android.providers.settings
package:com.android.sharedstoragebackup
package:com.android.printspooler
package:com.android.dreams.basic
package:com.android.inputdevices
package:com.android.bips
package:com.android.cellbroadcastreceiver
package:com.google.android.webview
package:com.google.android.syncadapters.contacts
package:com.example.android.notepad
package:com.android.keychain
package:com.android.chrome
package:com.android.dialer
```

Figure 17…

```
File  Actions  Edit  View  Help
MAIN_(); ×   ADB_(); ×
┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/mobilePenTest/diva-apk-file]
└─$ adb shell pm list packages | grep diva
package:jakhar.aseem.diva

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/mobilePenTest/diva-apk-file]
└─$
```

```
*Untitled 1 - Mousepad
File  Edit  Search  View  Document  Help
1 Kali:
2 IP - 10.0.2.5
3
4 Android vm:
5 IP - 10.0.2.16
6 diva application android name: jakhar.aseem.diva
```

Figure 18.

Above is the name of the DIVA package as it is on the Android device. The reason for this difference is because when installing into Kali the names differ. So, the name on my Kali machine isn't the same as the APK on the Android machine. That is what the command above does; list all packages specifying the keyword to look for on the Android machine.

```
File  Actions  Edit  View  Help
MAIN_(); ×   ADB_(); ×

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/mobilePenTest/diva-apk-file]
└─$ adb connect 10.0.2.16
connected to 10.0.2.16:5555

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/mobilePenTest/diva-apk-file]
└─$ ls
DIVA  DivaApplication  DivaApplication.apk  LICENSE  README.md

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/mobilePenTest/diva-apk-file]
└─$ adb install DivaApplication.apk
Performing Streamed Install
Success

┌──(bitman㉿KaliII)-[~/Documents/CodingDojo/mobilePenTest/diva-apk-file]
└─$
```

Figure 19. Next, let's install the APK application onto the Android device using Android Debug Bridge.

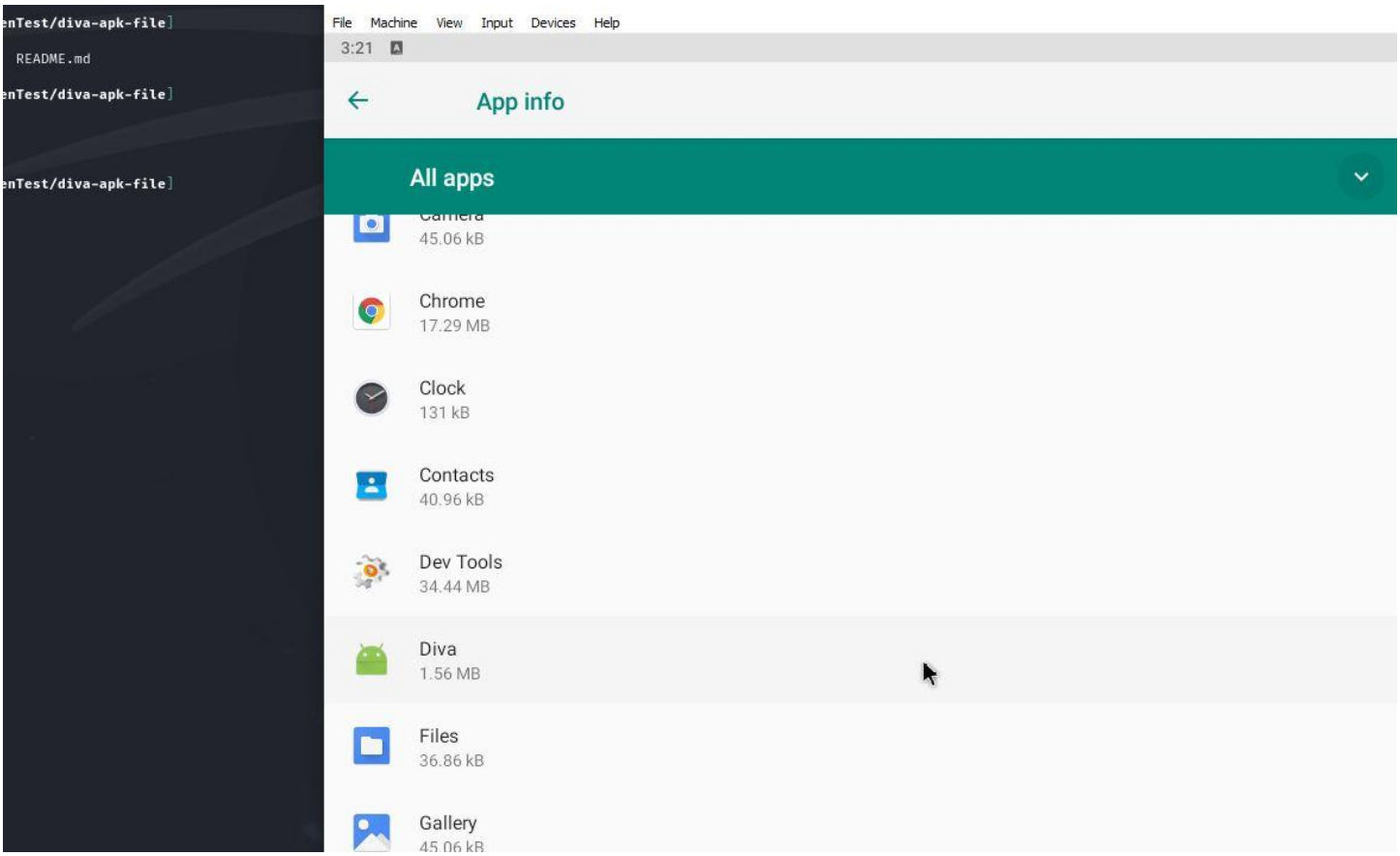

Figure 19. Now, onto step 6, which requires me to launch the DIVA application on the Android device and complete the Input Validation Issues – Part 1, as shown in Figure 20.

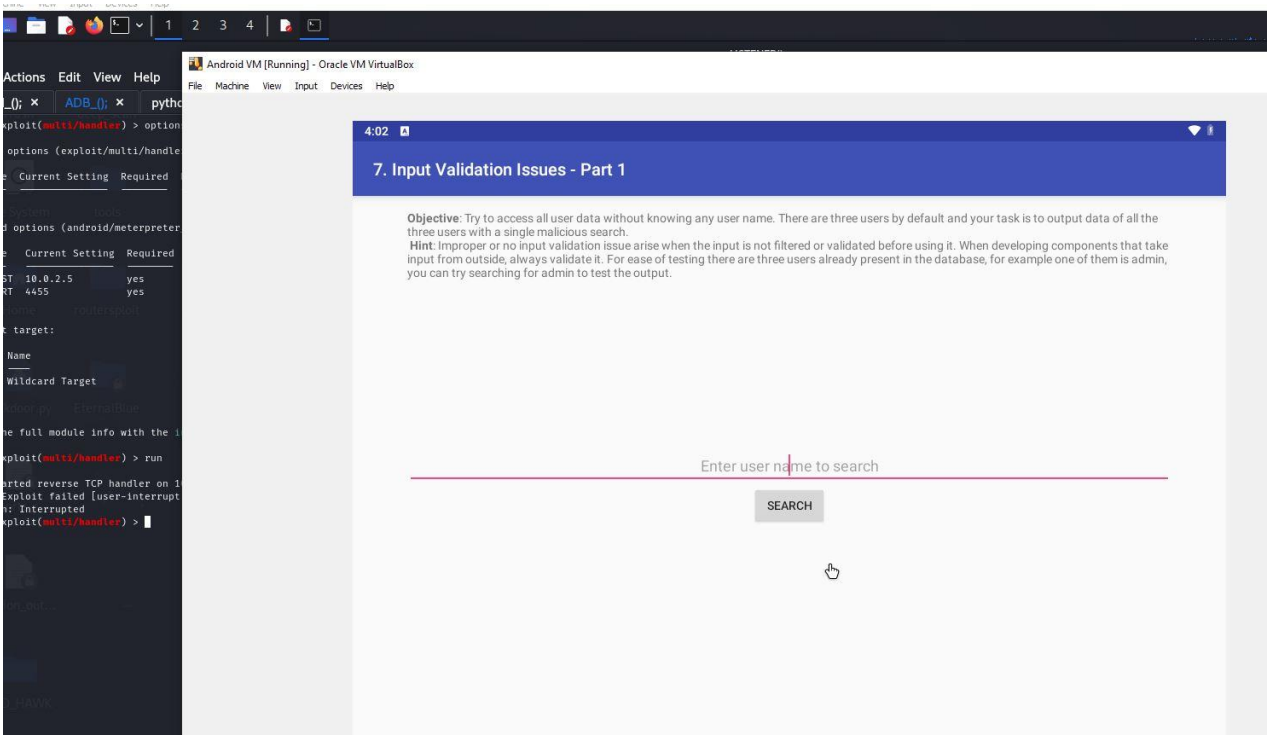

Figure 20…

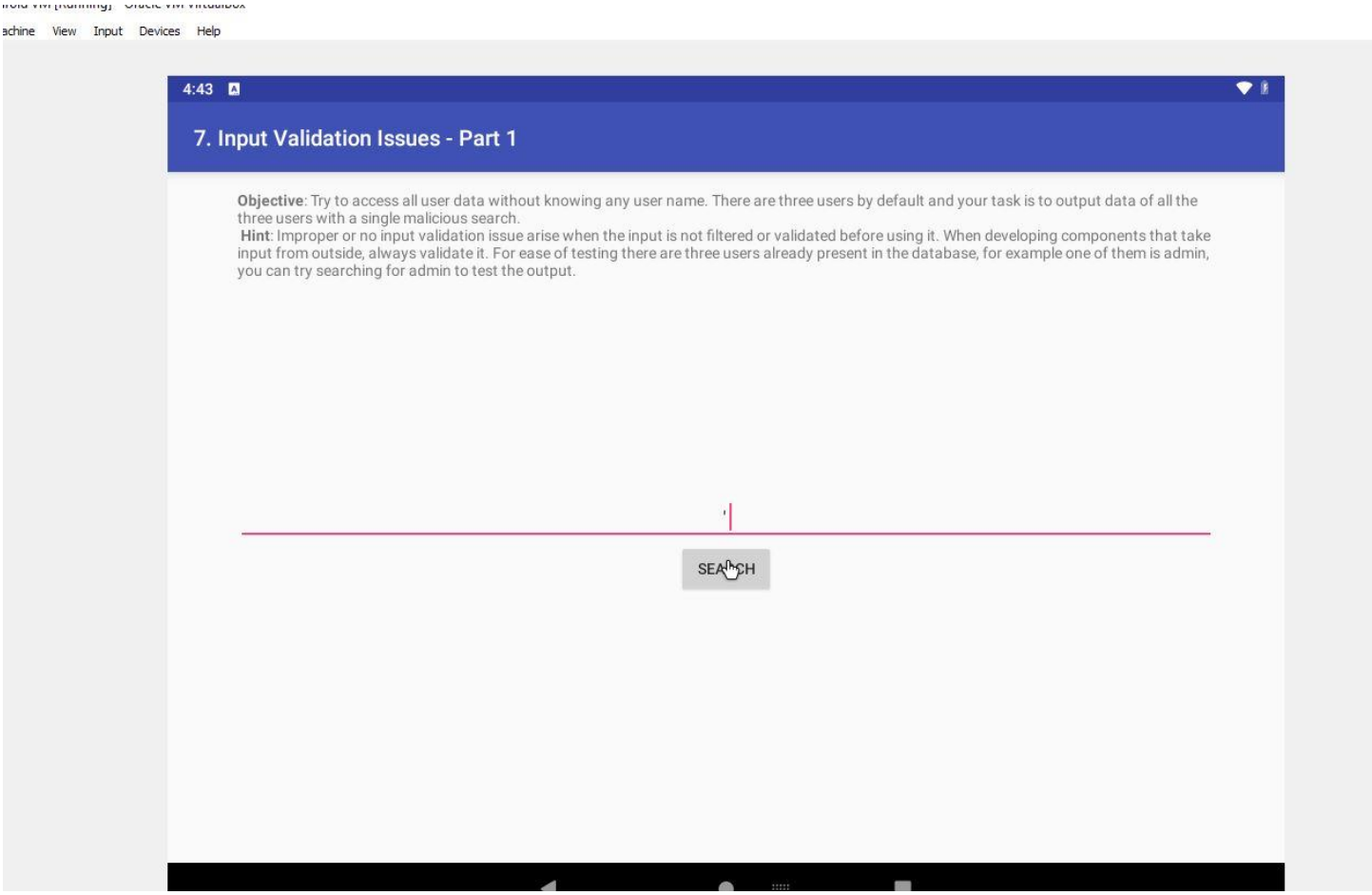

Figure 21. Step one of any injection, find a valid input and then try to break it. For me this one accomplished with a comma.

Figure 22. At first I was a bit confused because the statement above I thought should've worked; however, upon further investigation I noticed a discrepancy in the syntax. I had a single double quote instead of one single quote.

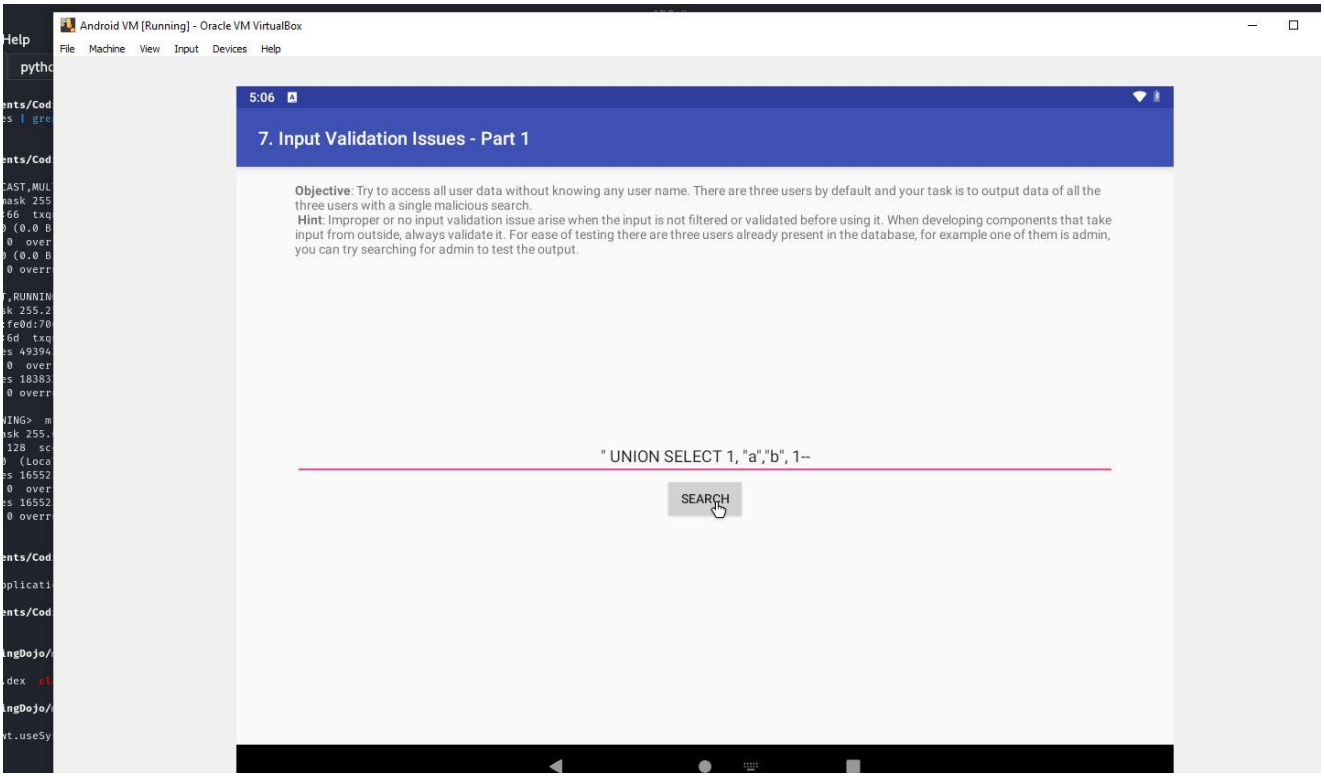

Figure 23. If that wasn't enough I decided to incorporate a UNION SELECT which did not bear fruit whatsoever. Yet again the single double quote popping up like a State Farm agent.

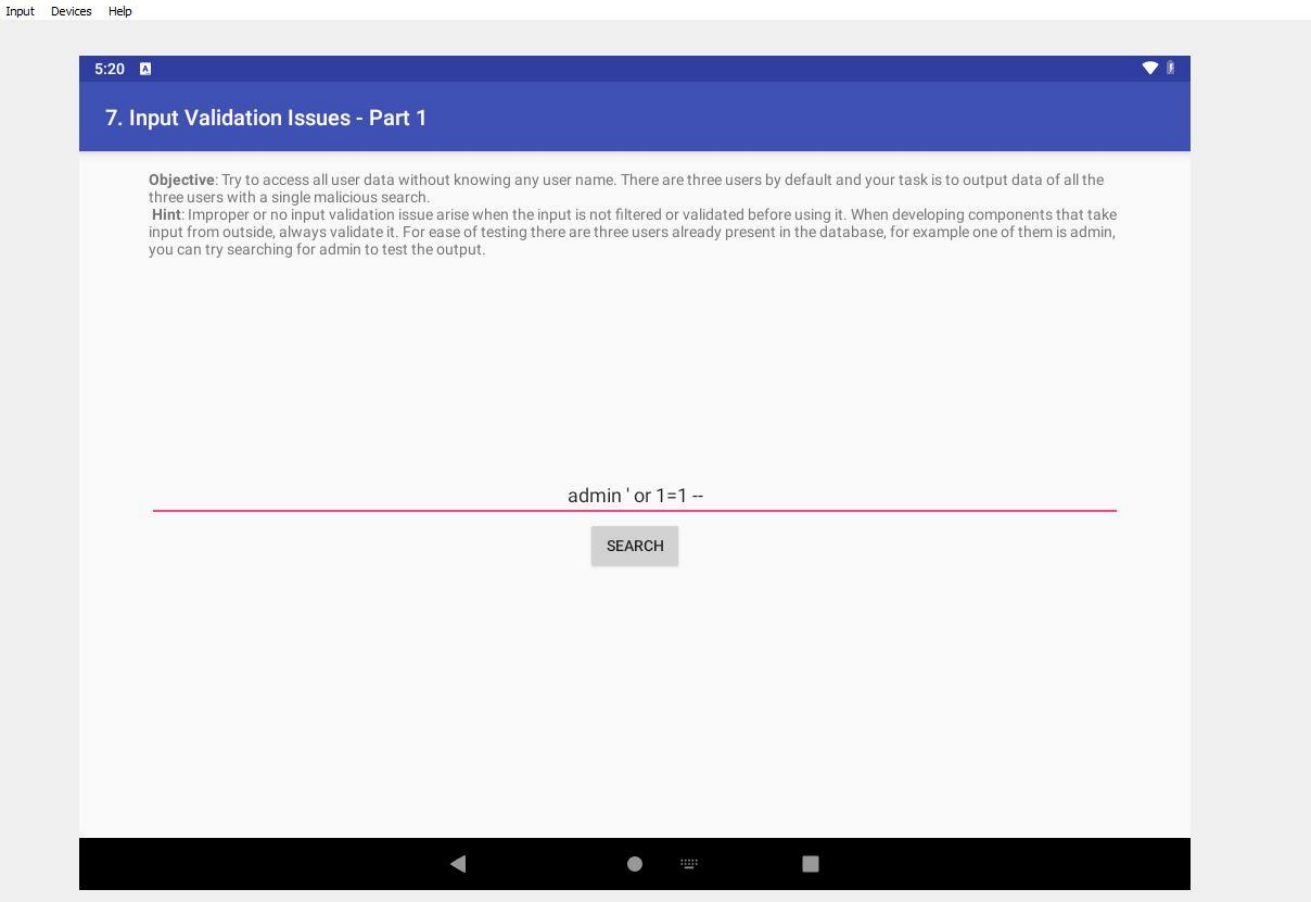

Figure 24. Finally a working line of code.

The output vanished pretty quick so I took a screenshot of the evidence with my phone. I apologize if it is blurry and if it's too blurry, the output below, verbatim:

User: (admin) pass: (passwd123) Credit Card: (1234567812345678)

User: (diva) pass: (p@ssword) Credit Card(1111222233334444)

User: (john) pass: (password123) Credit Card(5555666677778888)

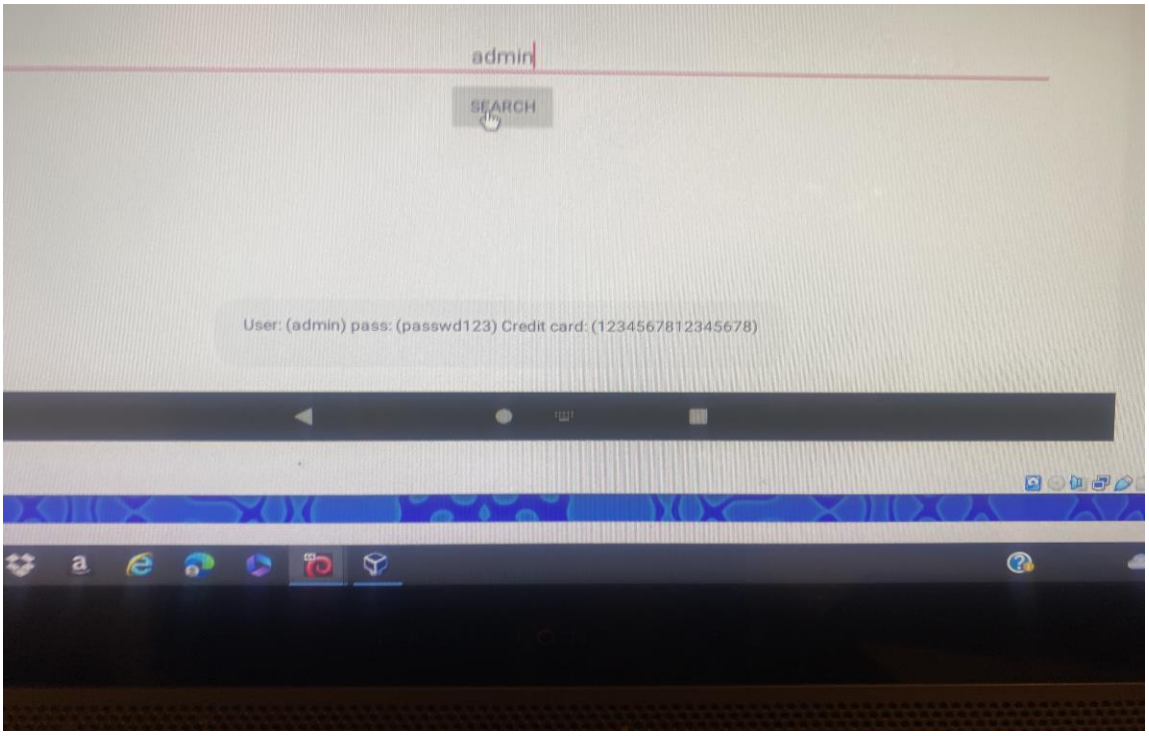

Figure 25. Since admin is a valid user, let's adjust input so that we get the admin's information back as well as the information of the other two users. This is done like so, Figure 26.

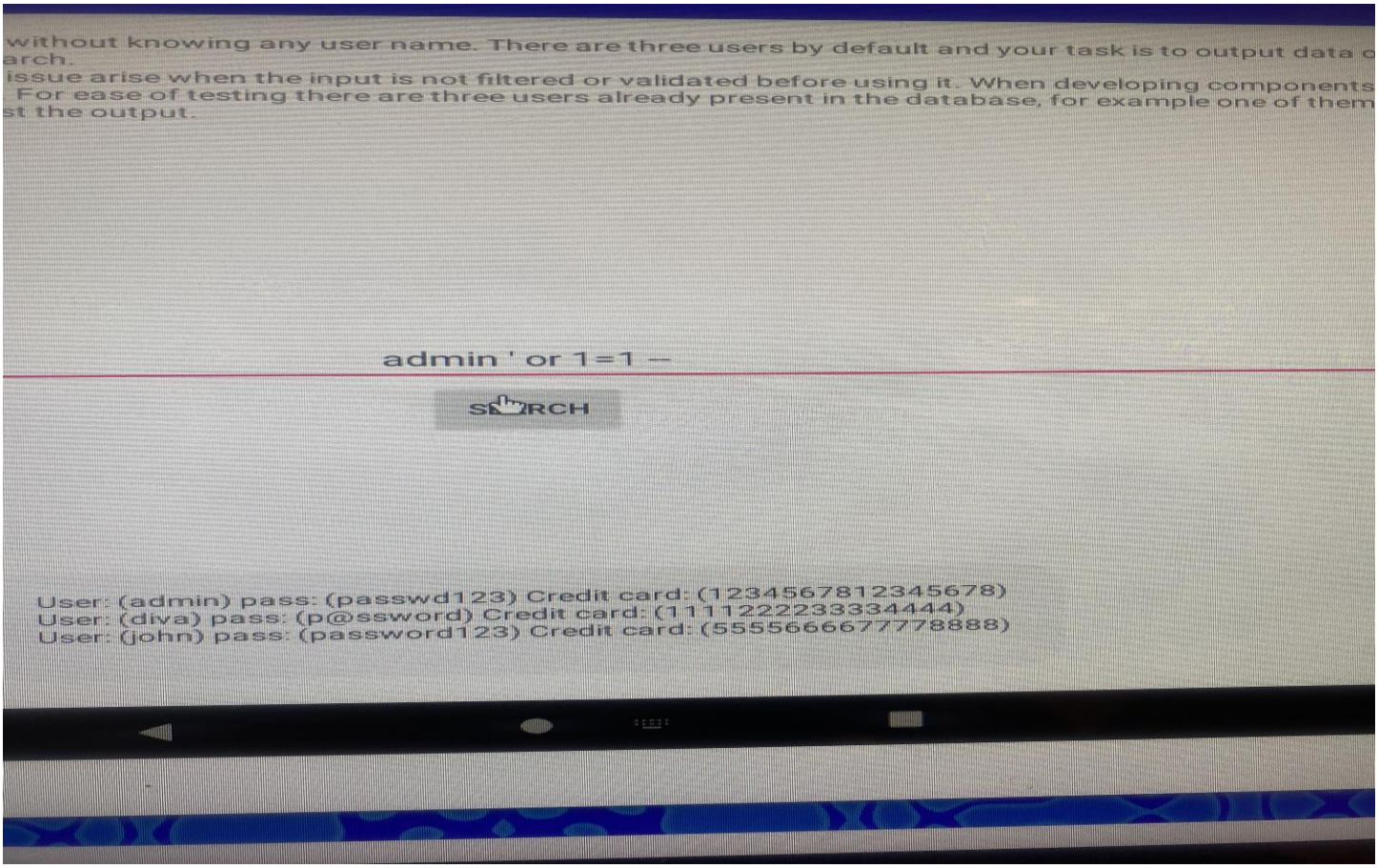

Figure 26… as mentioned above on Figure 24, the verbose output is located there.

Figure 27. I also wanted to verify the information returned by DIVA's servers. With this verification, all there was left was to uninstall the package. If we refer back to Figure 9 where I mentioned the jd-gui tool, you see the JAR file entitled classes loaded into the GUI. In this screenshot, you see me navigate to said file with SqlInjectionActivity.class. There it is, clear as day; the SQL syntax used in the SQLIUSER.lowercase() table.

Figure 28. ADB uninstall Jakhar.aseem.diva package file.