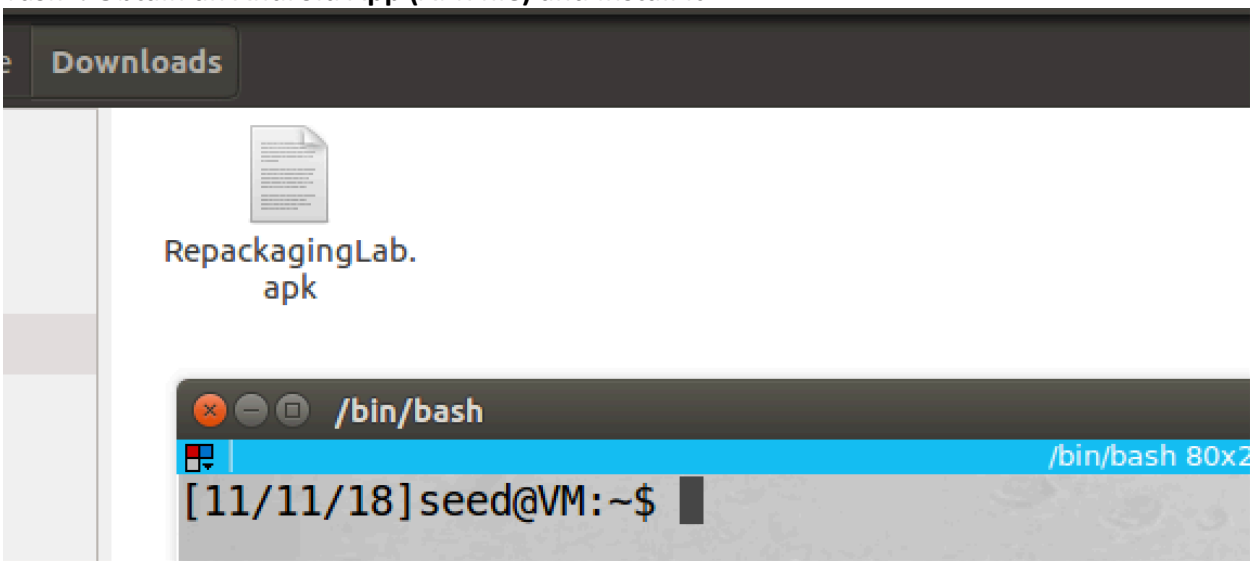
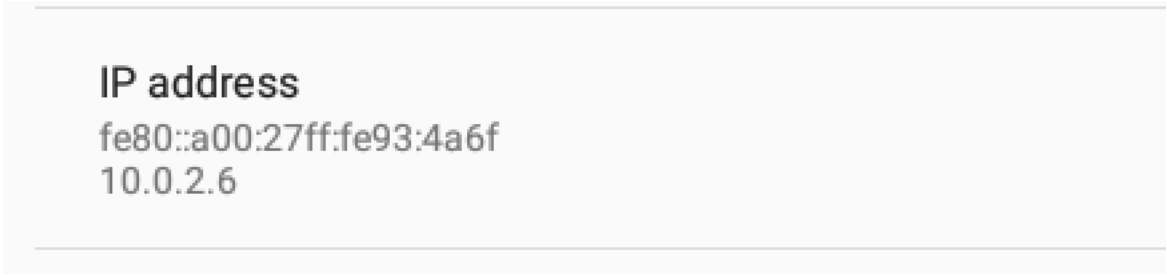
CSE643 Lab12
Yishi Lu
11/16/2018

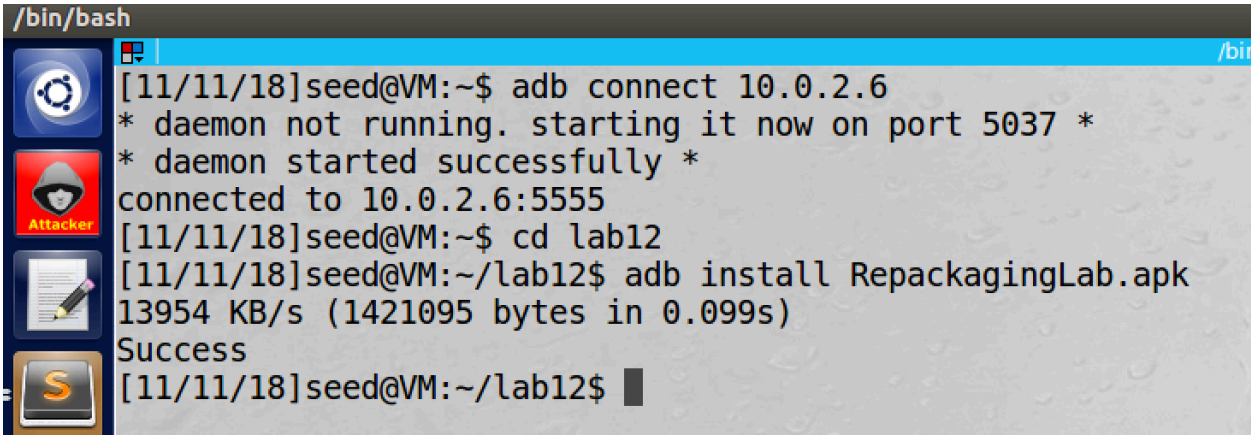
Task1: Obtain an Android App (APK file) and Install It



screenshot1, we download the APK file from lab website



screenshot2, to install the APK file by adb, we need android VM IP address, we can find it in setting->status

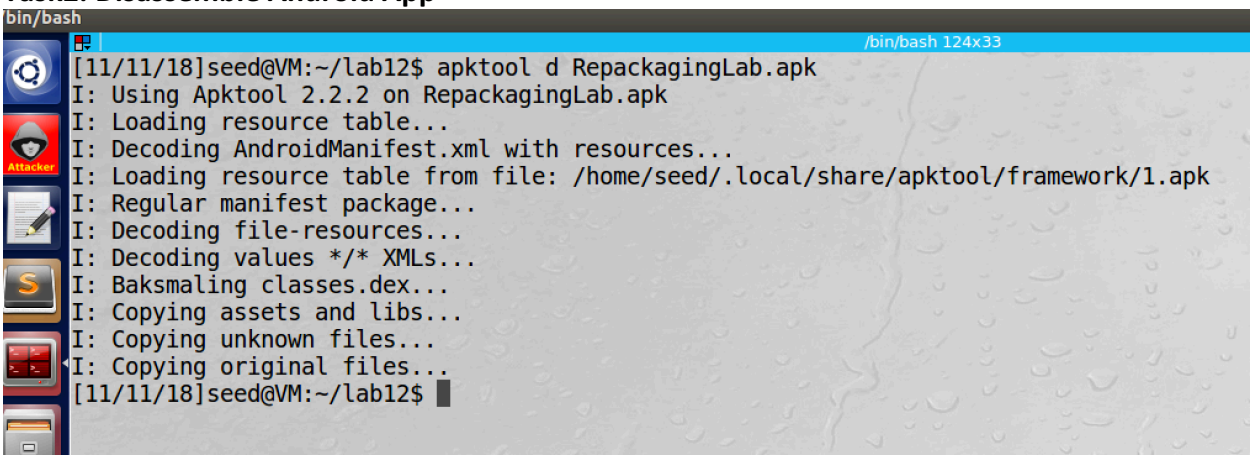


screenshot3, we successfully install the APK on the android VM

Observation and Explanation:

In this task, we install an android app on the android VM. We first download the app from lab website (screensho1). Then we find out the android VM's IP address, which is 10.0.2.6 (screenshot2). Last, we install the app by adb command (screenshot3).

Task2: Disassemble Android App

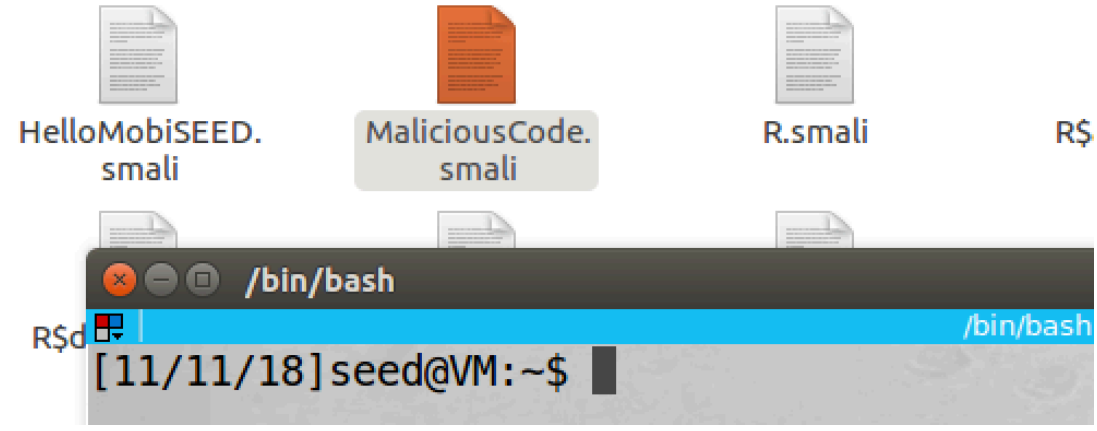


screenshot1, we successfully disassemble RepackagingLab.apk file

Observation and Explanation:

To perform repacking attack on an APK file, we need to modify the file. But APK file contains Dalvik bytecode cod, so it is not readable. Therefore, we need to convert it to Smali, which is readable to human. To do this, we can use APKTool to disassemble dex code to smali code. As screenshot1 shows, we successfully disassemble the APK file, and we get several files and folders.

Task3: Inject Malicious Code



screenshot, we get the smali code from lab website, and we put it in samli/cod/mobiseed/repackaging folder

```
<?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.READ_CONTACTS" />
    <uses-permission android:name="android.permission.WRITE_CONTACTS" />
    <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.MaliciousCode" >
            <intent-filter>
            <action android:name="android.intent.action.TIME_SET" />
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

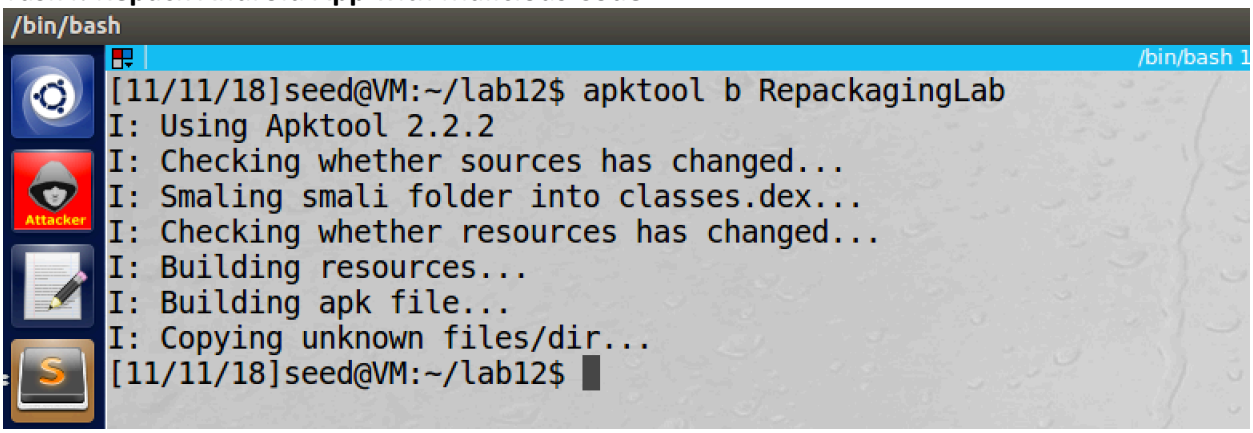
```
/bin/bash
/bin/bash 80x24
[11/11/18]seed@VM:~$
```

screenshot2, we modify AndroidManifest.xml

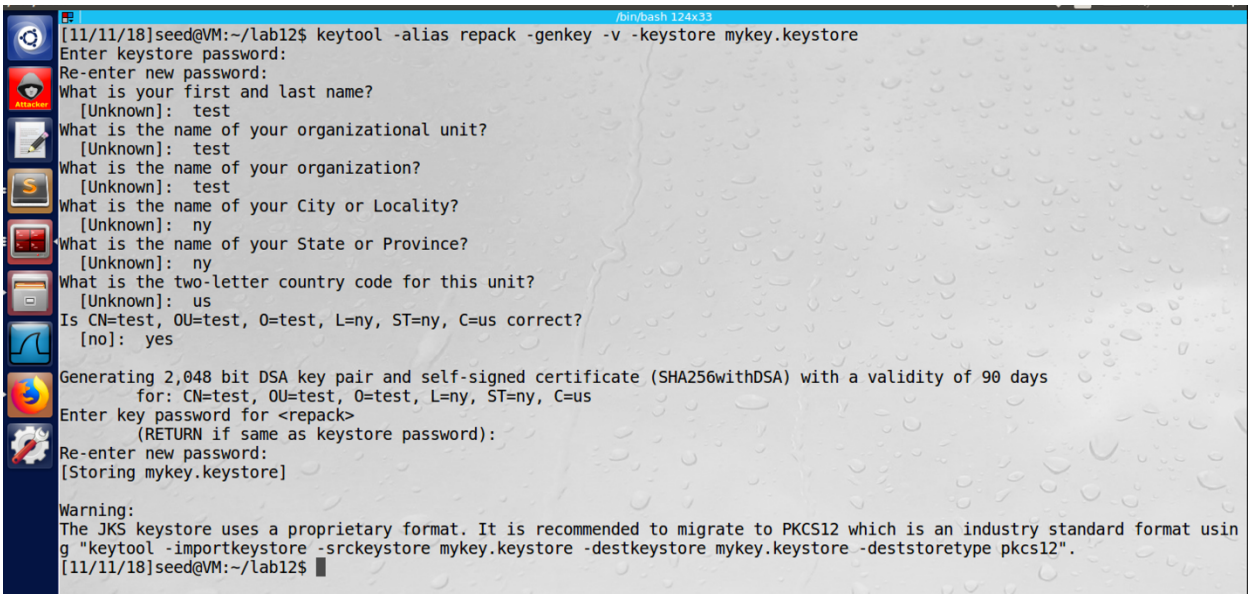
Observation and Explanation:

Now, we disassembled the APK file, we can modify its content or add new content. In our case, we want to add malicious code to the app. After our malicious code is triggered, all contact information of the victim Android system will be deleted. How can the malicious code be triggered? There are several ways, we choose to use broadcast receiver. We can write a broadcast receiver to listen system broadcast (i.e. TIME_SET broadcast). So after such events happen, our malicious code will be triggered. Because I never write any Android program before, I download smali code which provided on the lab website and put it in folder smali/com/mobiseed/repackaging (screensho1). Moreover, we also need to tell the system when to invoke our broadcast receiver. To achieve this, we can add some code in the AndroidManifest.xml file. As screenshot2 shows, we add several lines. The first two lines (in tag user-permission) are user permission, so the app is allowed to read and write on contact. The rest of lines (in tag receiver) is used to register our broadcast receiver on TIME_SET. So when victim changes the time of the phone, our malicious code will be triggered.

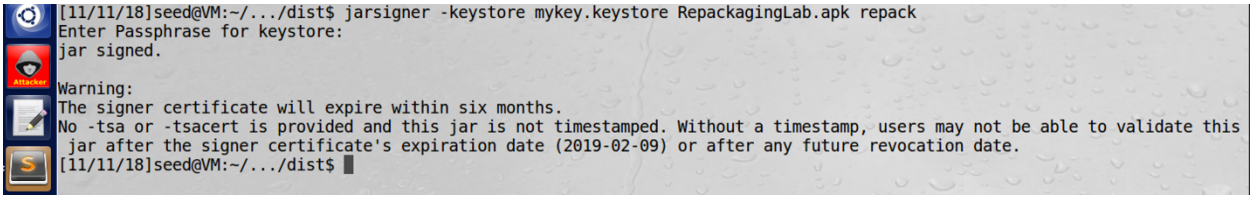
Task4: Repack Android App with Malicious Code



screenshsot1, we successfully assemble the new APK file



screenshot2, we generate the public key and private key pair

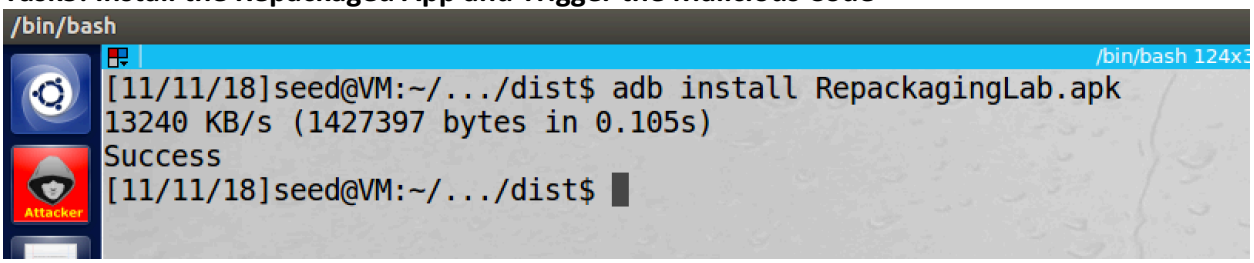


screenshot3, we successfully signed our new app

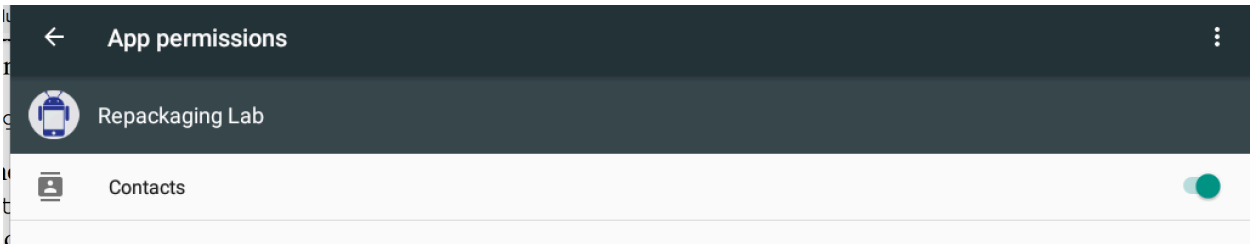
Observation and Explanation:

Now, we already added our malicious code into the new app, so we need to assemble it to be an APK file. As screenshot1 shows, we successfully assemble our new app. Because Android requires all apps to be digitally signed before they can be installed, we need to sign our app with public key certificate. To do this, we first need a public key. For the Android app, the public key is meant to make the app run on Android device, not for security. So we can create a public key certificate by ourselves, and we can put any name on it. As screenshot2 shows, we successfully create a self signed public key certificate. And then we use it to sign our new app (screenshot3).

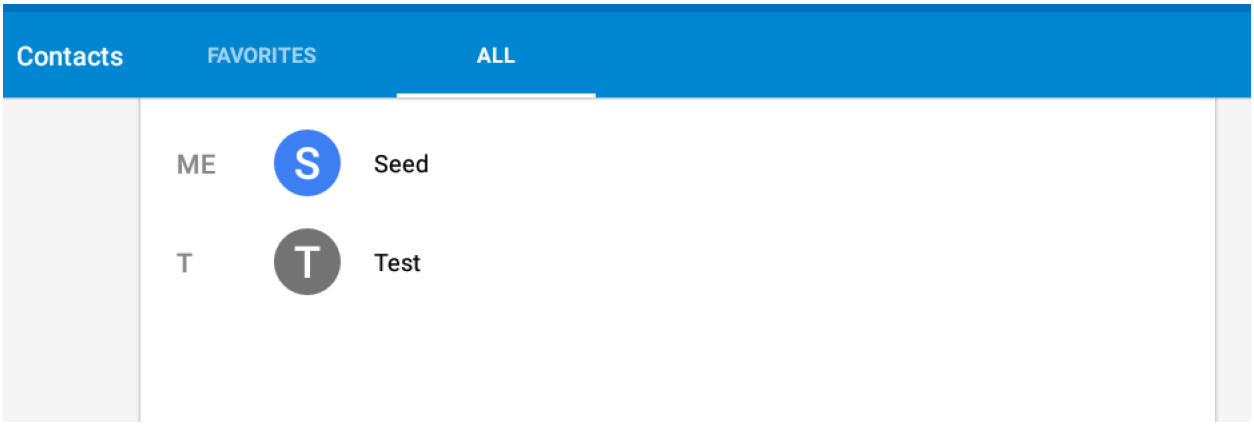
Task5: Install the Repackaged App and Trigger the Malicious Code



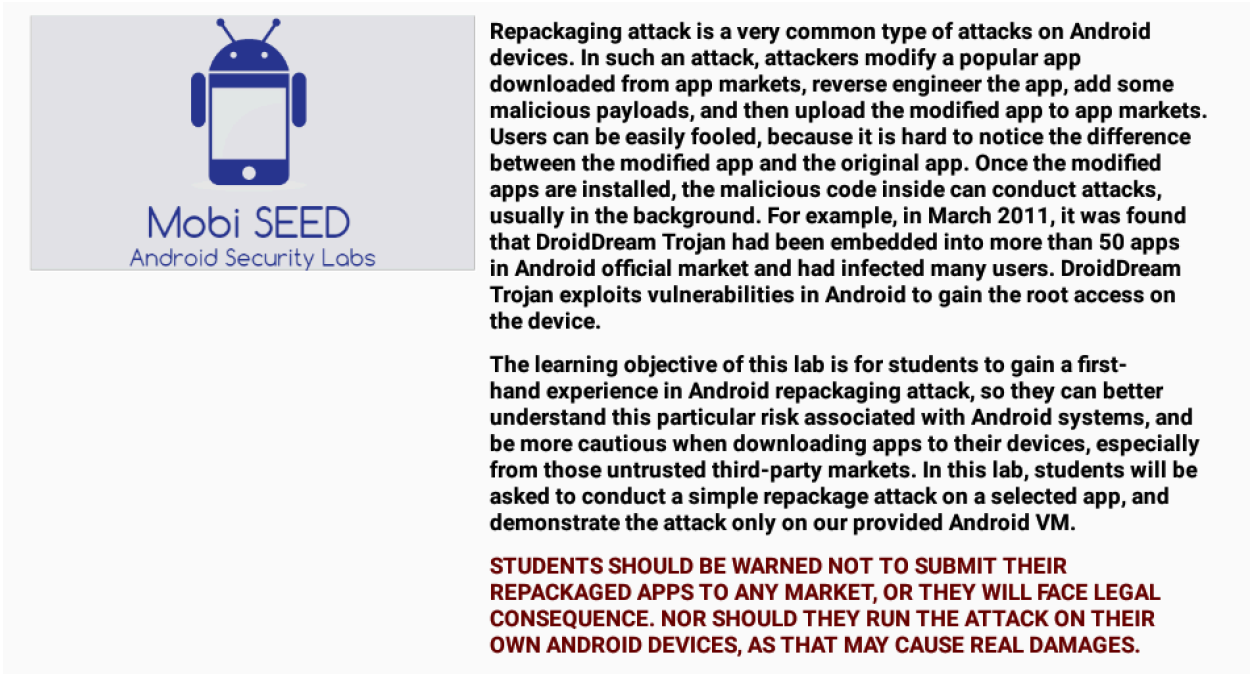
screenshot1, after uninstall the old app on the Android VM, we successfully install the new app on the Android VM



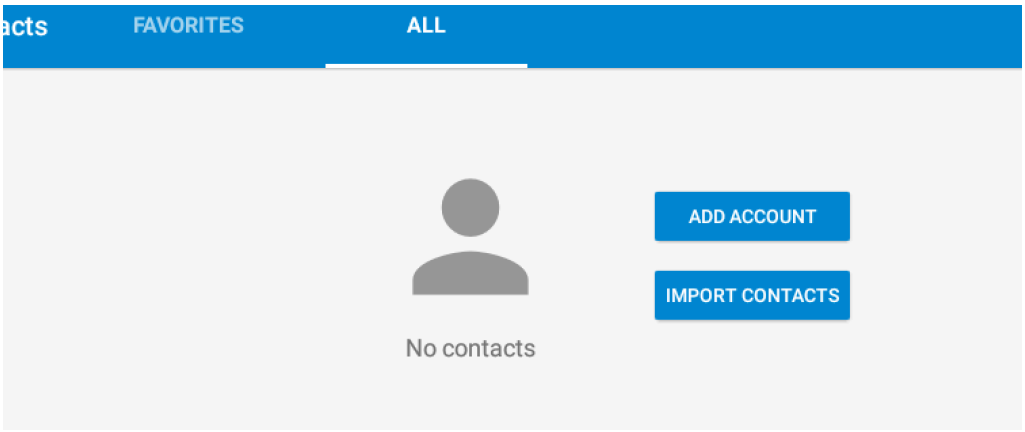
screenshot2, we give permission to the app



screenshot3, we add new contact “Test”



screenshot4, we run the run app

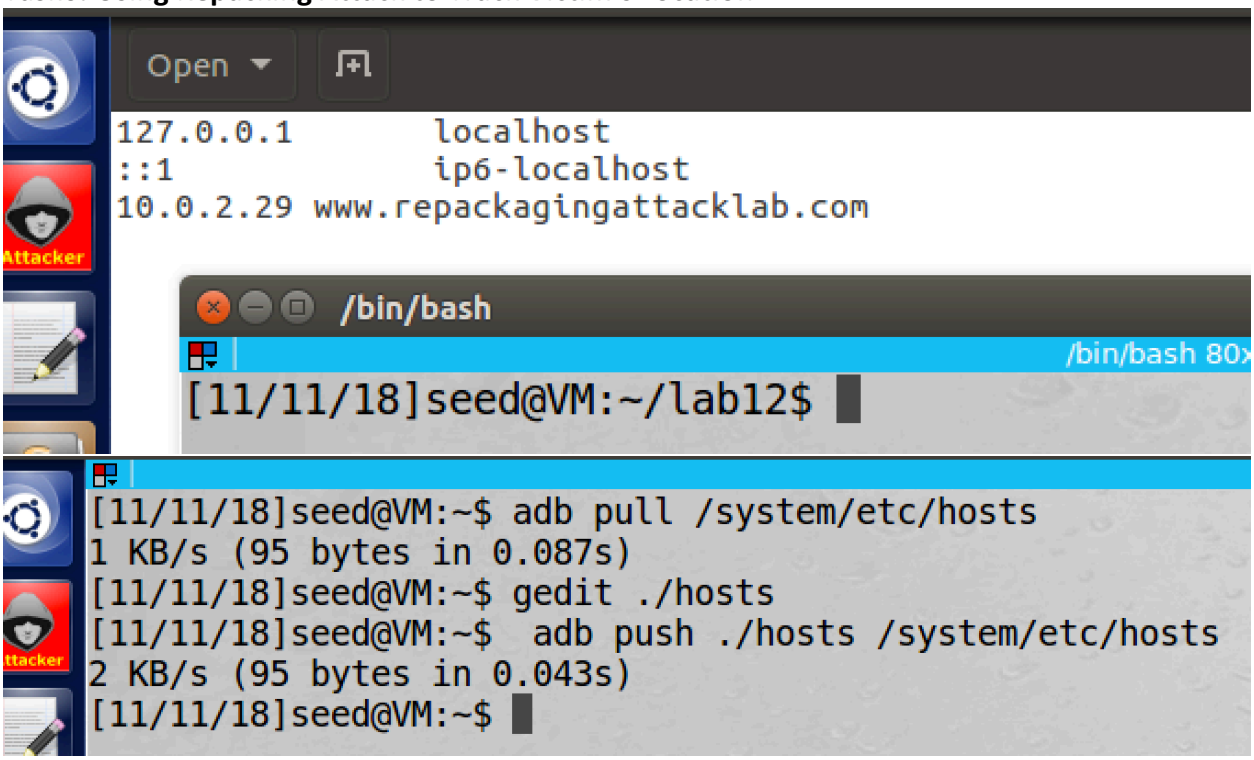


screenshot5, after we set the time on the system, all contacts are deleted

Observation and Explanation:

In this task, we perform Android repacking attack on the VM. First we uninstall the app which we installed in task1. Then we install the new app (screenshot1). We switch to the Android VM, we first give read and write permission on contact to the new app (screenhsot2). Then we also add a new contact (screenshot3). Afterwards, we run the new app (screenshot4). After we set the system time, we check the contact again; as screenshot5 shows, all contacts are removed, so our attack is successful.

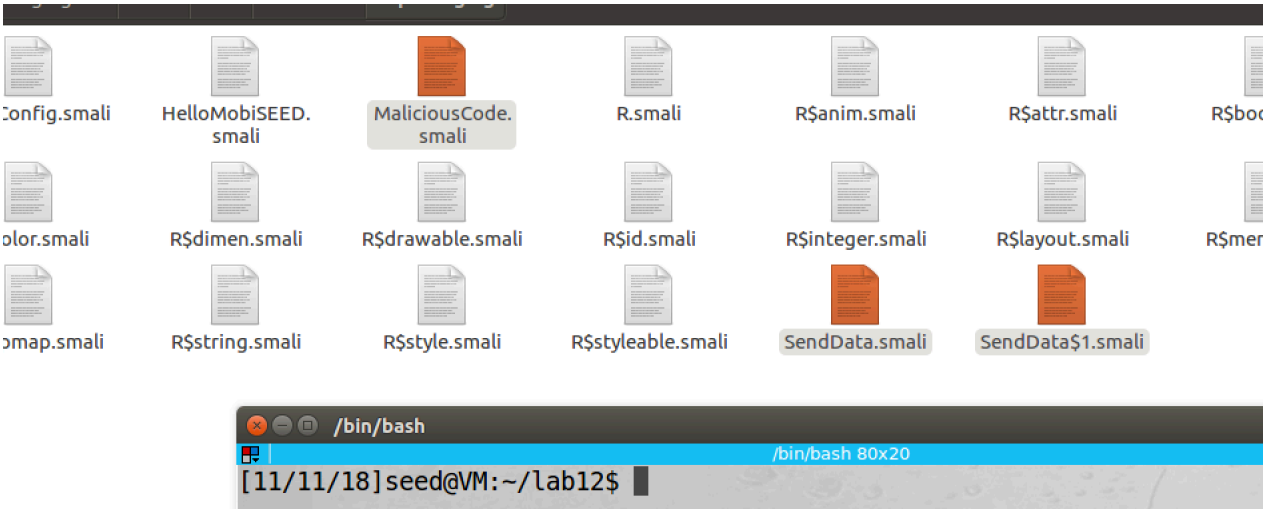
Task6: Using Repacking Attack to Track Victim's Location



screenshot1, we modified the hosts file of the Android VM

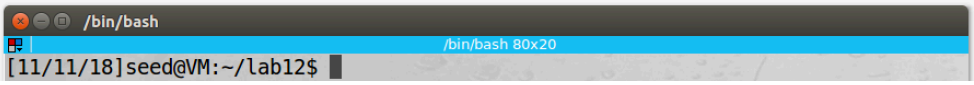
```
I: Using Apktool 2.2.2 on RepackagingLab.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/seed/.local/share/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
[11/11/18]seed@VM:~/lab12$
```

screenshot2, we disassemble the app again

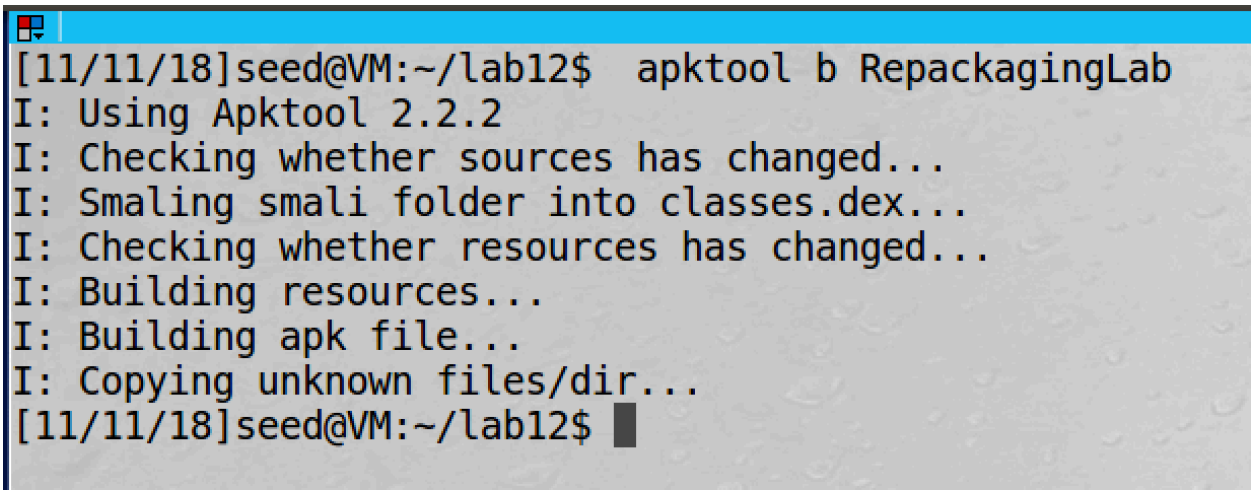


screenshot3, we get three smali files from lab website, and we put them in the folder smali/com/mobiseed/repackaging

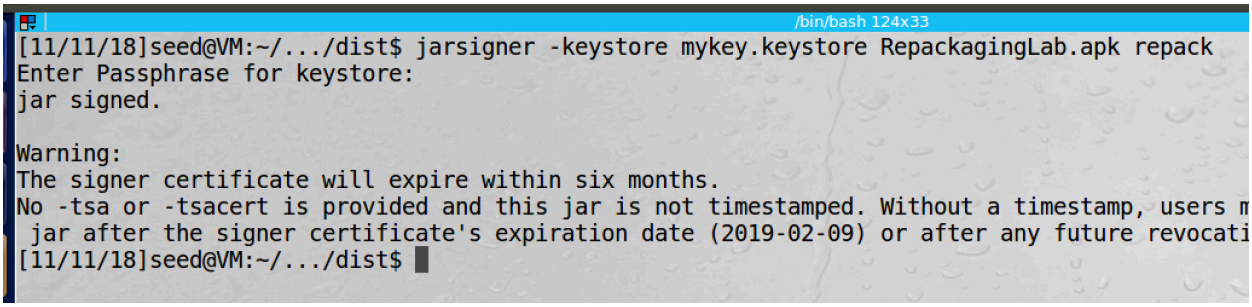
```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.mobiseed.repackaging" platformBuildVersionCode="23"
platformBuildVersionName="6.0-2166767">
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>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



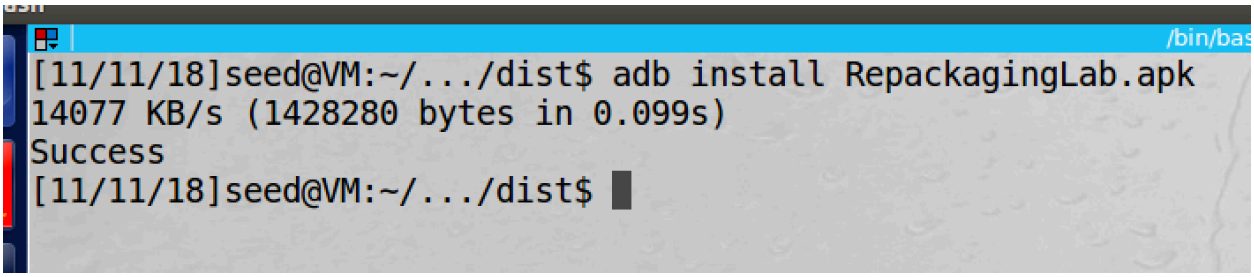
screenshot4, we modified AndroidManifest.xml



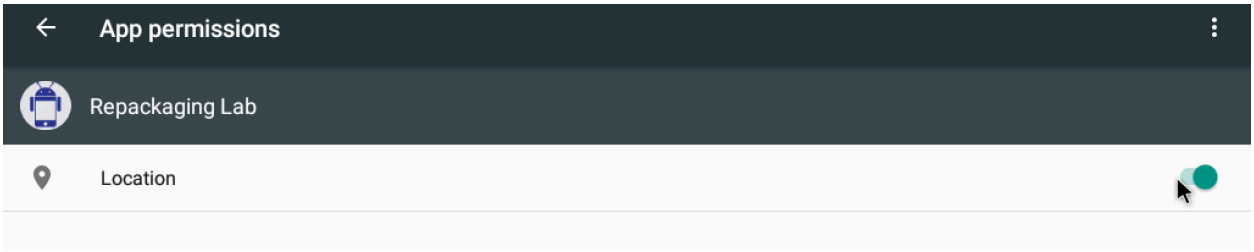
screenshot5, we assemble the new app



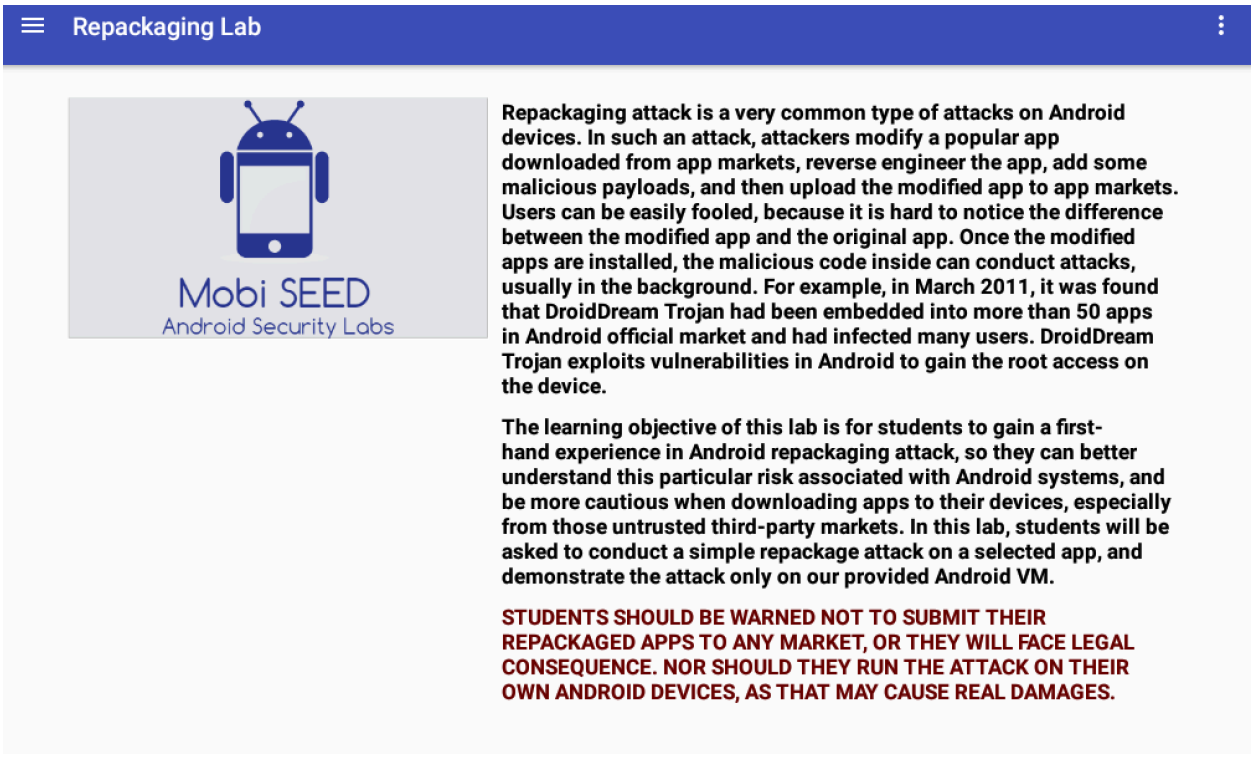
screenshot6, we sign the app by our public key



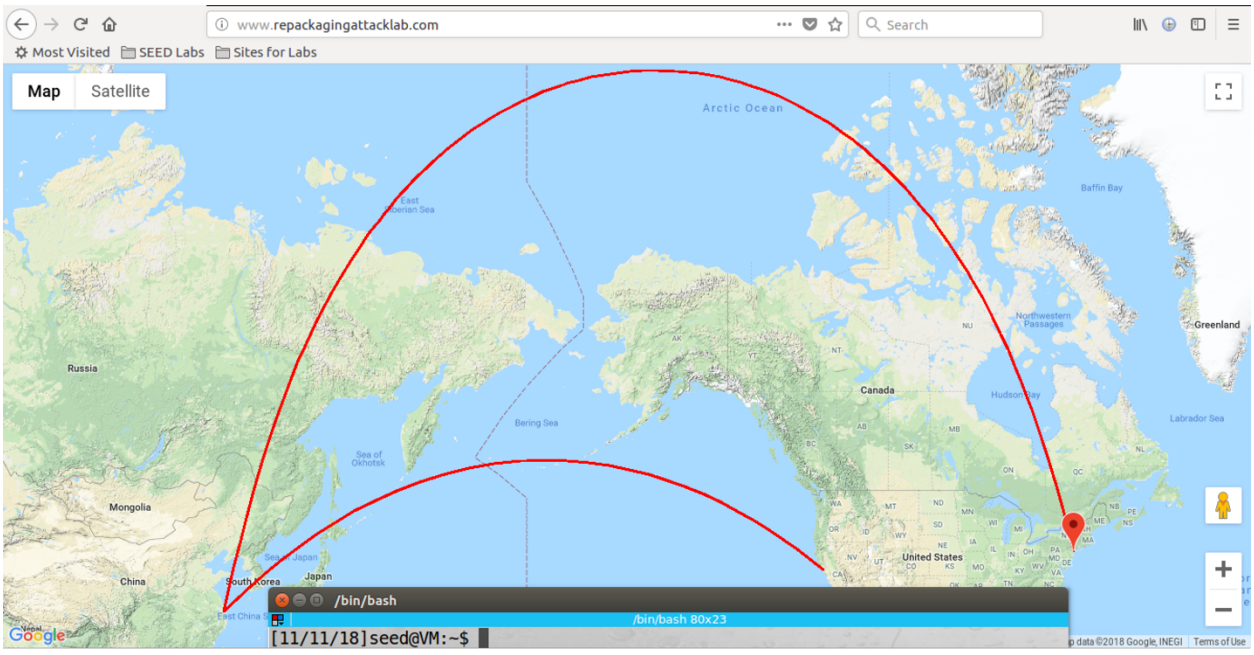
screensho7, we successfully install the new app



screensho8, we toggle on the location for the new app



screenshot9, we run the new app



screenshot10, we start at New York city, then we change to shanghai, finally we move to San Francisco. And our malicious is triggered, so the website display our movement

Observation and Explanation:

In this task, we modify our malicious code; instead of deleting all contact information on the victim, we decide to track the location of the victim. Our malicious code will send the

coordinate to our server, so we need to configure DNS on the victim phone. Our server's IP address is 10.0.2.29, so we add "10.0.2.29 www.repackagingattacklab.com" on the hosts file of the victim Android VM (screenshot1). Then we need to repackage the app, we still use the original app. We first disassemble it (screenshot2). Then we get smali files from lab website, and we put them in the folder smali/com/mobiseed/repackaging (screenshot3). Moreover, we also need to modify AndroidManifest.xml, we need to add three permissions, two for locations and one for internet access, so the app can have permission to access the location of the Android VM and send the location back to our server. We also need to register the broadcast receiver on TIME_SET, so our malicious code can be triggered by setting time (screenshot4). Then we assemble the new app, and we sign it by the public key which we created before (screenshot5 and 6). Then we install it on the Android VM (screenshot7). In the Android, we first need to give location access permission to our new app (screenshot8). Then we run the new app and the mock location app. We also open the website www.repackagingattacklab.com to monitor the Android system location movement. We first select New York city, then we select Shanghai, finally we select San Francisco. After that, we go back to the website, we see the movement of the Android VM (screenshot10), this means that our malicious is triggered, and the coordinate of the victim VM is sent back to our server. So our attack is successful.