Android Repackaging Lab Report

Mudit Vats
mpvats@syr.edu
11/23/2018

Table of Contents

Overview

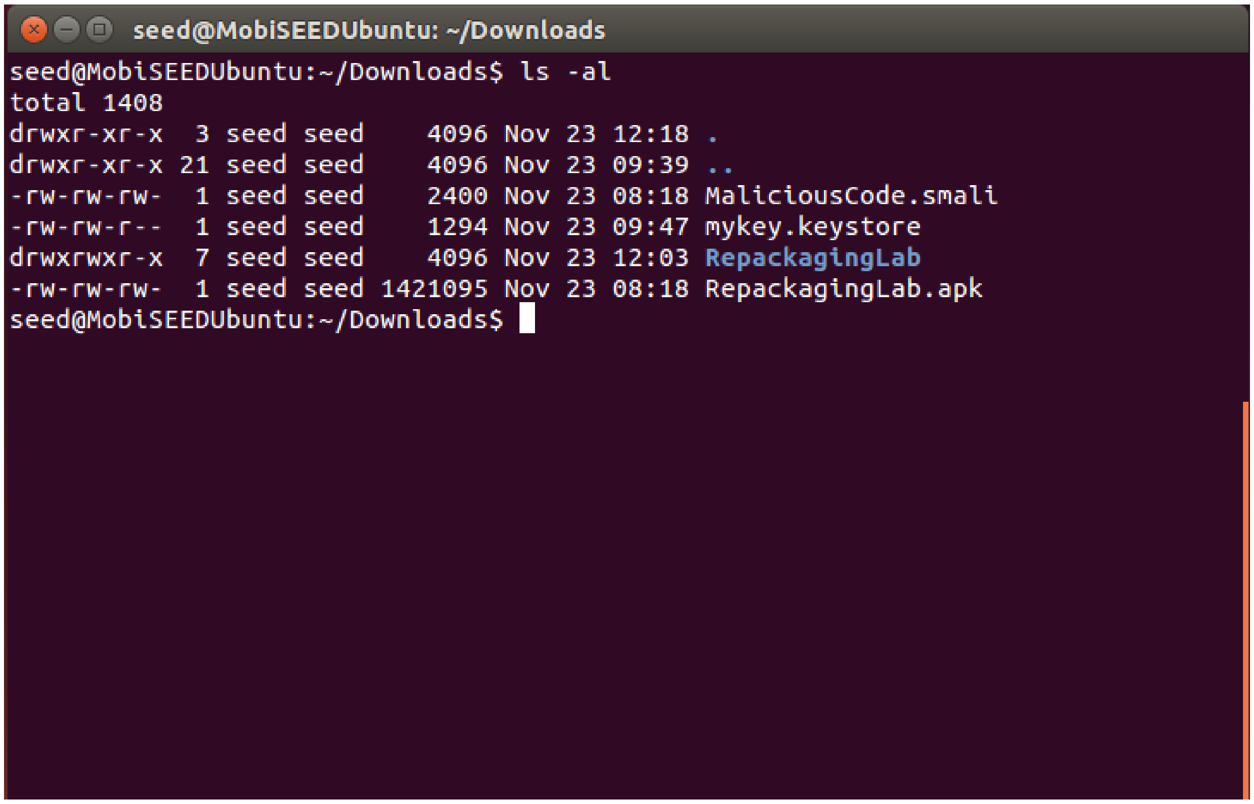
This lab report presents observations and explanations for the tasks described in the Android Repackaging Attack Lab.

Task 1: Obtain an Android App (APK file)

For this lab we are reusing the RepackagingLab.apk and MaliciousCode.smali. Both of these files are downloadable at the Lab Site in the **Files that are needed** section:

http://www.cis.syr.edu/~wedu/seed/Labs_Android5.1/Android_Repackaging/

The figure below shows the RepackagingLab.apk and MaliciousCode.smali files copied to the Ubuntu VM's Downloads directory.



```
seed@MobiSEEDUbuntu:~/Downloads$ ls -al
total 1408
drwxr-xr-x  3 seed seed    4096 Nov 23 12:18 .
drwxr-xr-x 21 seed seed    4096 Nov 23 09:39 ..
-rw-rw-rw-  1 seed seed    2400 Nov 23 08:18 MaliciousCode.smali
-rw-rw-r--  1 seed seed    1294 Nov 23 09:47 mykey.keystore
drwxrwxr-x  7 seed seed    4096 Nov 23 12:03 RepackagingLab
-rw-rw-rw-  1 seed seed 1421095 Nov 23 08:18 RepackagingLab.apk
seed@MobiSEEDUbuntu:~/Downloads$ 
```

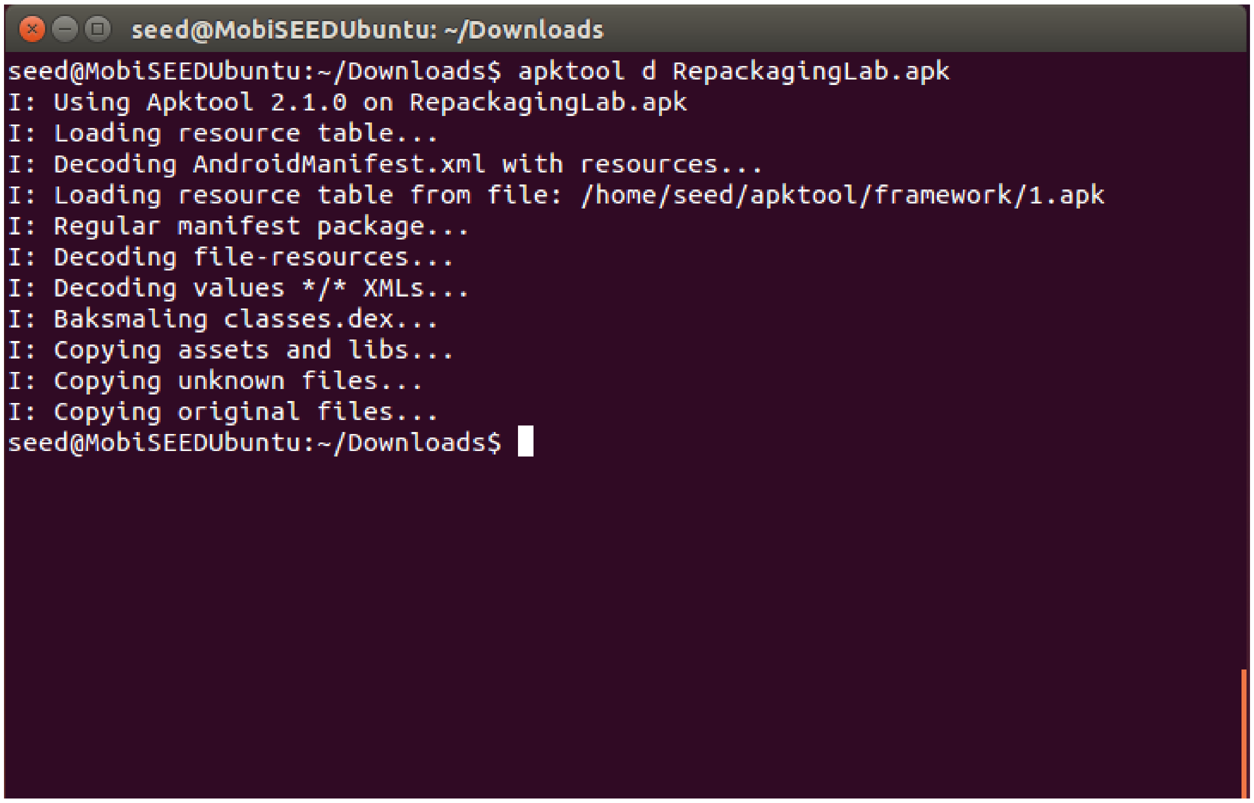
Observations / Explanation

We are reusing the files given to us for this lab. We are given two files –

- MaliciousCode.smali – This is the malicious code which deletes all of the users' contacts whenever the RepackagingLab application receives the BOOT_COMPLETED broadcast message.
- RepackagingLab.apk – This is the RepackaginLab application which we will use to disassemble and insert our malicious code.

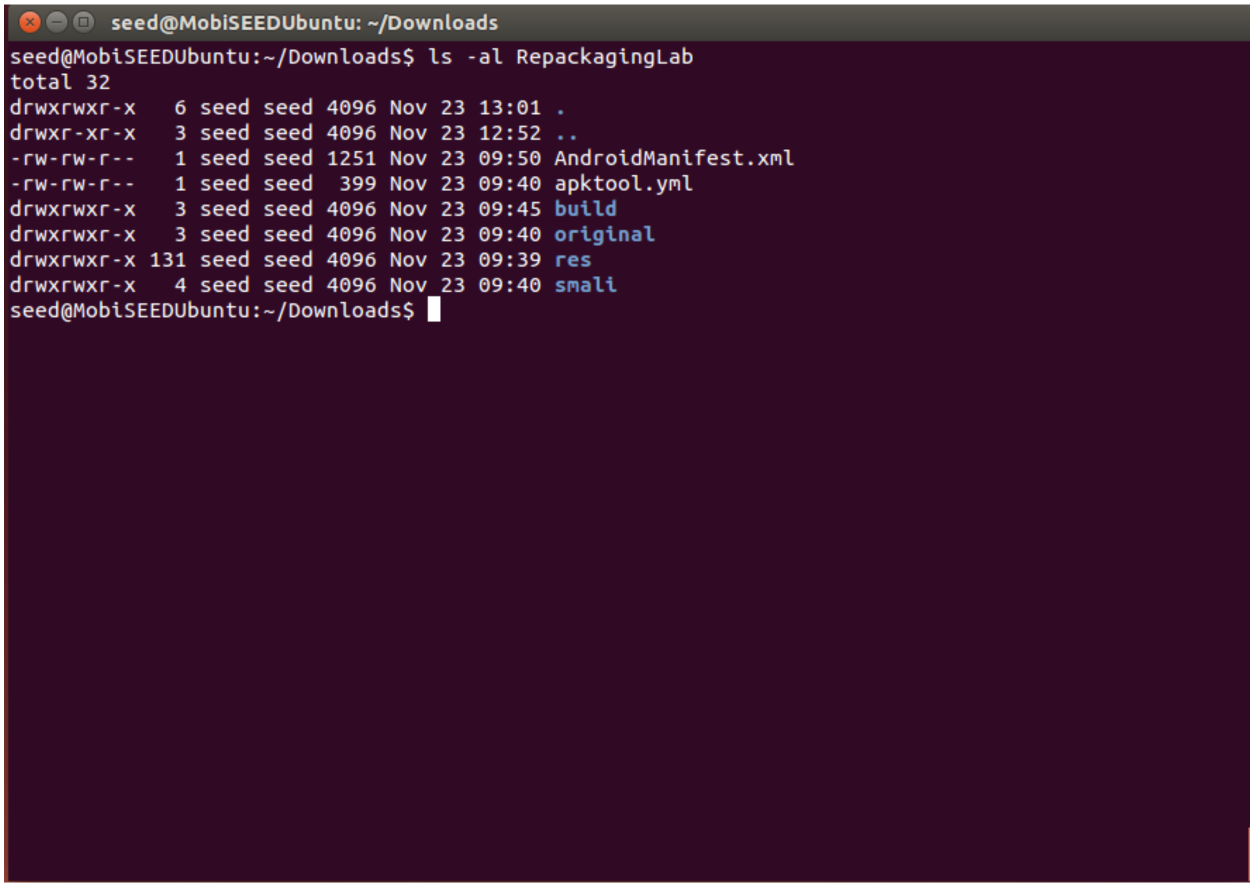
Task 2: Disassemble Android App

The figure below shows the disassembly of the RepackagingLab.apk.



```
seed@MobiSEEDUbuntu: ~/Downloads
seed@MobiSEEDUbuntu:~/Downloads$ apktool d RepackagingLab.apk
I: Using Apktool 2.1.0 on RepackagingLab.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/seed/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...
seed@MobiSEEDUbuntu:~/Downloads$
```

In the figure below, we see the top-level subdirectories of the RepackagingLab.apk disassembly. There are many more subdirectories below these. The primary directories we are concerned with is the root directory for the AndroidManifest.xml and the smali directory as a location to copy our malicious code.



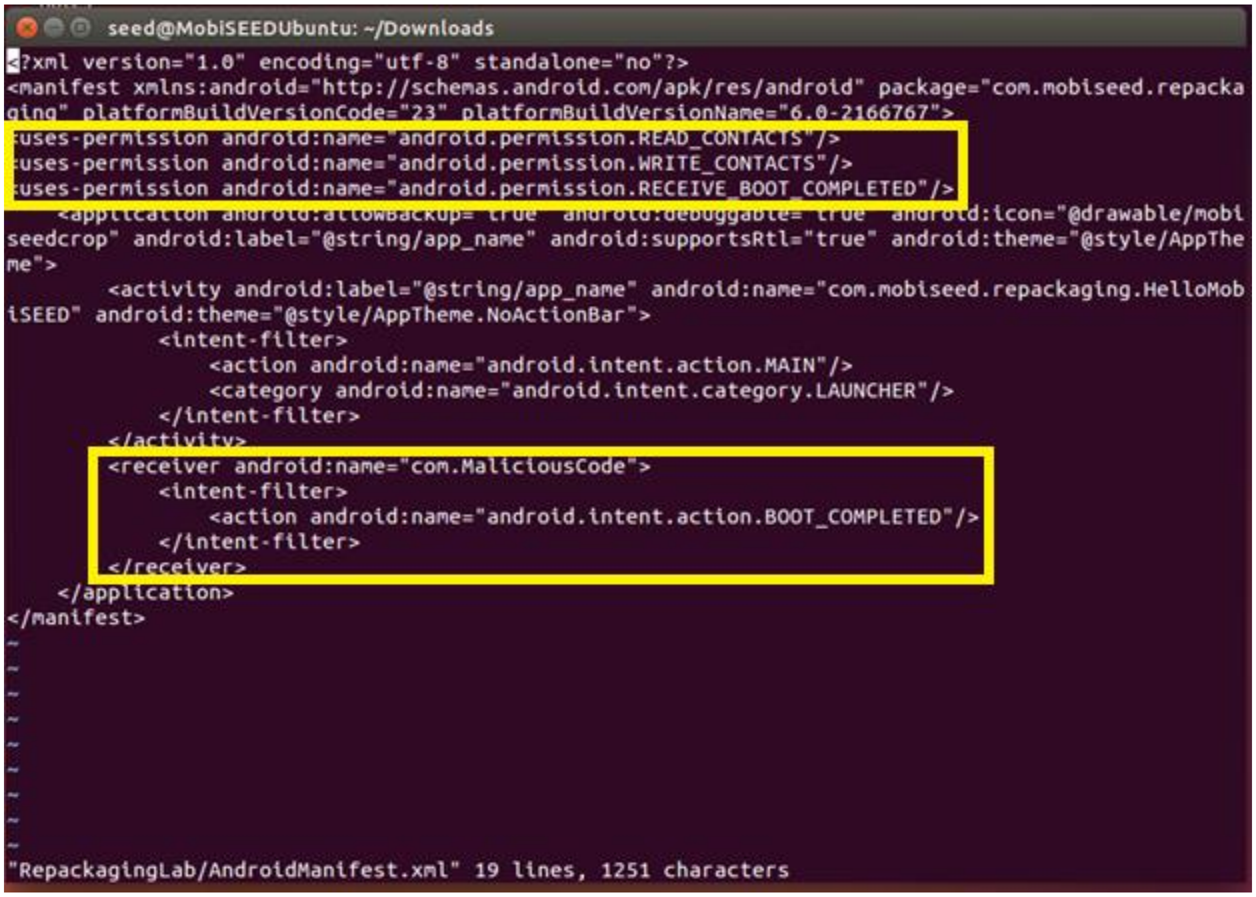
```
seed@MobiSEEDUbuntu:~/Downloads$ ls -al RepackagingLab
total 32
drwxrwxr-x   6 seed seed 4096 Nov 23 13:01 .
drwxr-xr-x   3 seed seed 4096 Nov 23 12:52 ..
-rw-rw-r--   1 seed seed 1251 Nov 23 09:50 AndroidManifest.xml
-rw-rw-r--   1 seed seed  399 Nov 23 09:40 apktool.yml
drwxrwxr-x   3 seed seed 4096 Nov 23 09:45 build
drwxrwxr-x   3 seed seed 4096 Nov 23 09:40 original
drwxrwxr-x 131 seed seed 4096 Nov 23 09:39 res
drwxrwxr-x   4 seed seed 4096 Nov 23 09:40 smali
seed@MobiSEEDUbuntu:~/Downloads$ 
```

Observations / Explanation

By using the apktool with the "d" option, we were able to disassemble the Android package. This allows us to modify the package which we will ultimately sign and install on a target Android system.

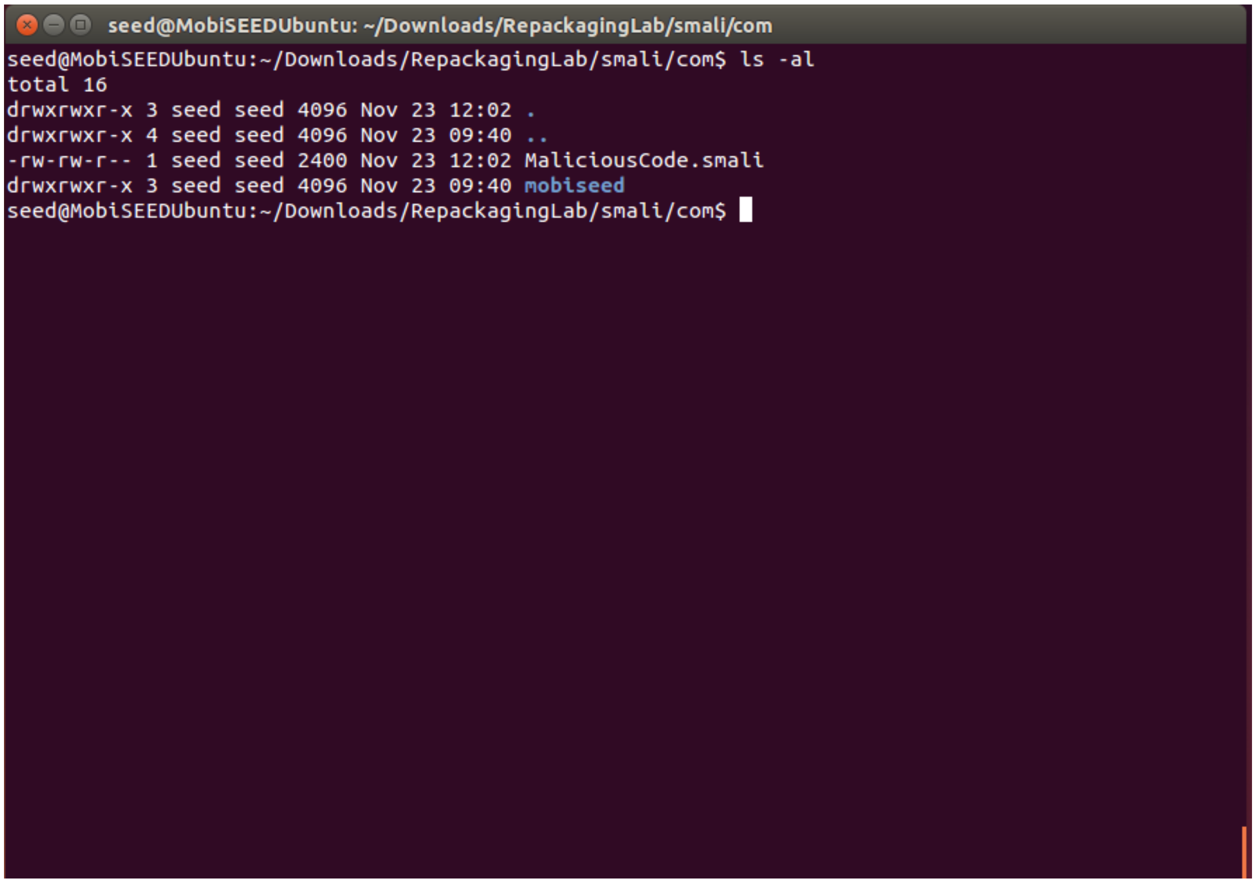
Task 3: Inject Malicious Code

In the figure below, we see the AndroidManifest.xml.



```
seed@MobiSEEDUbuntu: ~/Downloads
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.mobiseed.repacka
ging" platformBuildVersionCode="23" platformBuildVersionName="6.0-2166767">
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <application android:allowBackup="true" android:debuggable="true" android:icon="@drawable/mobi
seedcrop" android:label="@string/app_name" android:supportsRtl="true" android:theme="@style/AppThe
me">
        <activity android:label="@string/app_name" android:name="com.mobiseed.repackaging.HelloMob
iSEED" 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.BOOT_COMPLETED"/>
            </intent-filter>
        </receiver>
    </application>
</manifest>
~
~
~
~
~
~
~
~
~
"RepackagingLab/AndroidManifest.xml" 19 lines, 1251 characters
```

In the figure below, we see the MaliciousCode.smali copied to the RepackagingLab/smali/com directory.



```
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/smali/com$ ls -al
total 16
drwxrwxr-x 3 seed seed 4096 Nov 23 12:02 .
drwxrwxr-x 4 seed seed 4096 Nov 23 09:40 ..
-rw-rw-r-- 1 seed seed 2400 Nov 23 12:02 MaliciousCode.smali
drwxrwxr-x 3 seed seed 4096 Nov 23 09:40 mobiseed
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/smali/com$
```

Observations / Explanation

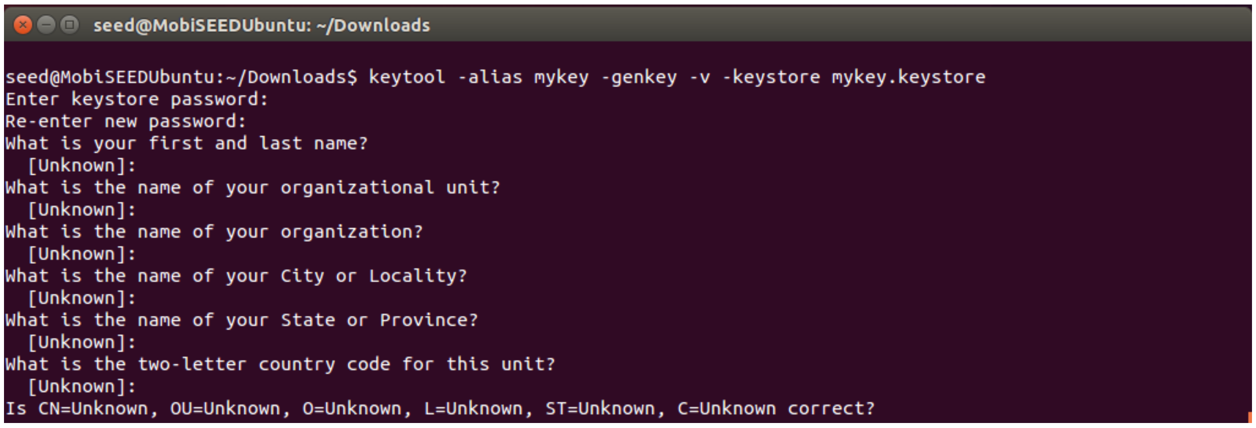
We make the updates to the AndroidManifest.xml which:

1. Gives the RepackagingLab application read/write access rights to the Contacts on the phone.
2. Gives the RepackagingLab application the ability to receive the RECEIVE_BOOT_COMPLETED message.
3. Registers the MaliciousCode class as the recipient of the BOOT_COMPLETED message.

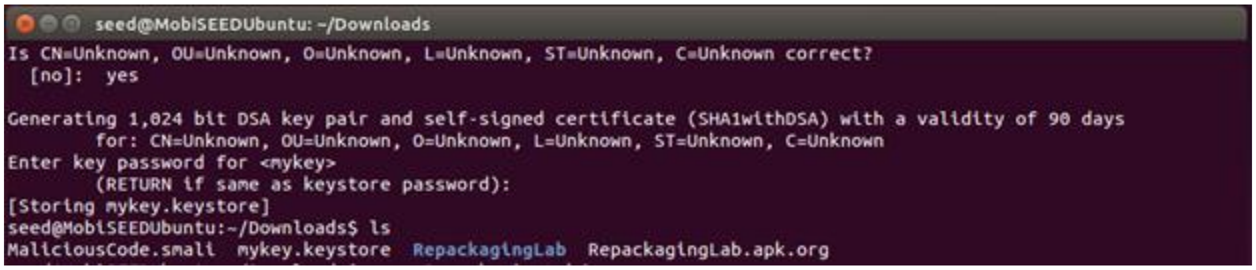
We also copy the MalicousCode class to the com directory which makes the new class available. This class, when executed when receiving the BOOT_COMPLETED message, will access the users' Contacts and delete them all.

Task 4: Repack Android App with Malicious Code

In the two figures below, we see the generation of a self-signed certificate in the mykey.keystore which we later use to sign the repackaged RepackagingLab.apk. Please notice the "mykey.keystore" in the last line of the second figure. This is where the certificate/keys are stored.

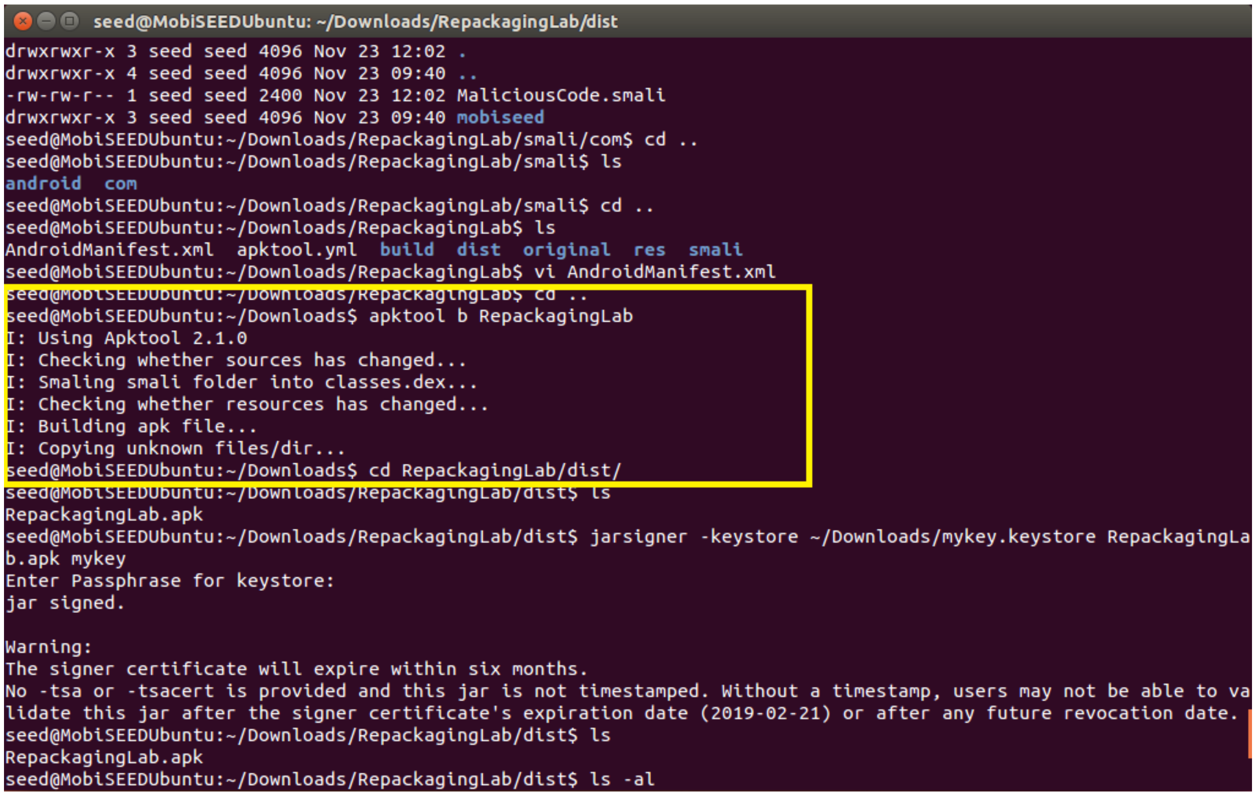


```
seed@MobiSEEDUbuntu: ~/Downloads

seed@MobiSEEDUbuntu:~/Downloads$ keytool -alias mykey -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?
```



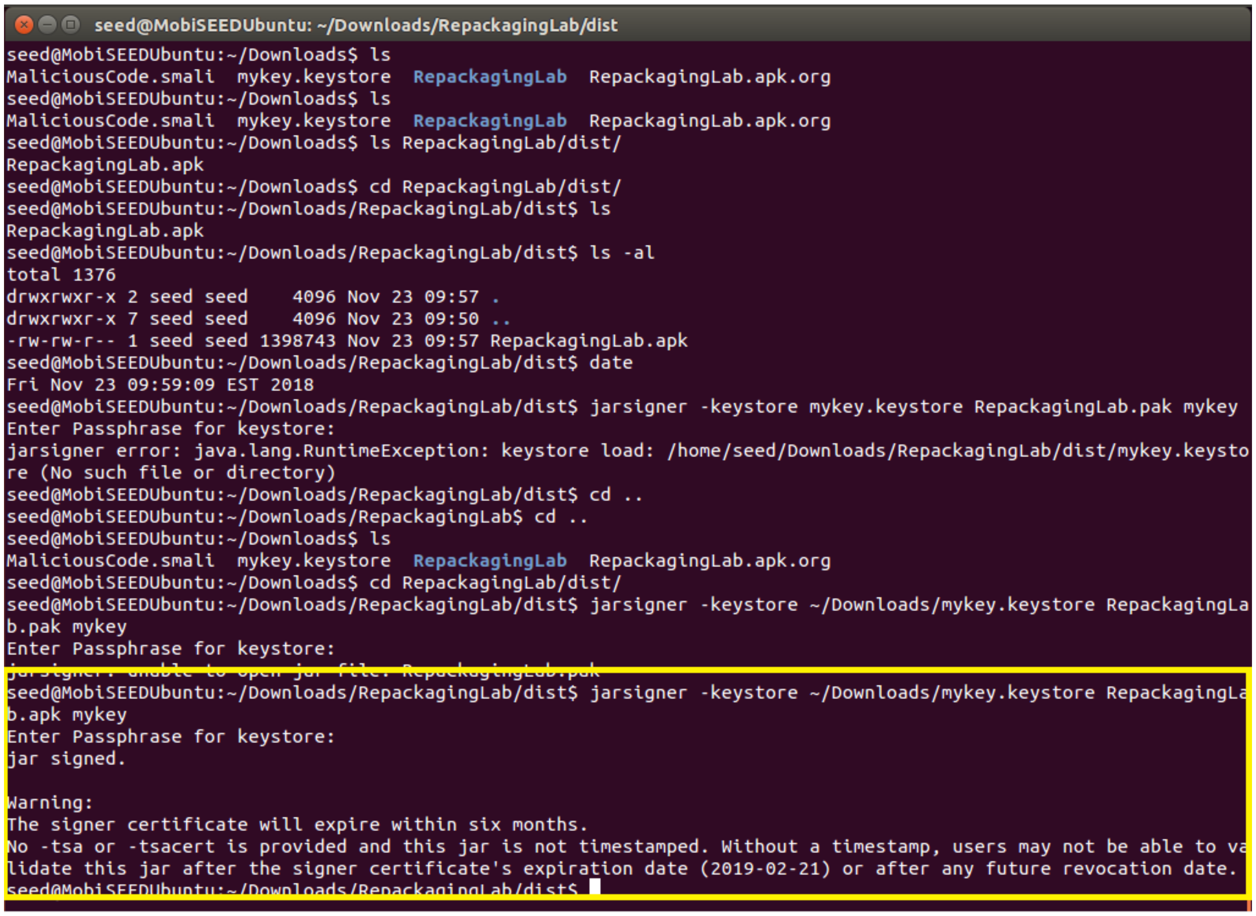
```
seed@MobiSEEDUbuntu: ~/Downloads
Is CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown correct?
  [no]:  yes

Generating 1,024 bit DSA key pair and self-signed certificate (SHA1withDSA) with a validity of 90 days
        for: CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
Enter key password for <mykey>
        (RETURN if same as keystore password):
[Storing mykey.keystore]
seed@MobiSEEDUbuntu:~/Downloads$ ls
MaliciousCode.smali  mykey.keystore  RepackagingLab  RepackagingLab.apk.org
```

In the figure below, we see the repackaging of RepackagingLab.



```
seed@MobiSEEDUbuntu: ~/Downloads/RepackagingLab/dist
drwxrwxr-x 3 seed seed 4096 Nov 23 12:02 .
drwxrwxr-x 4 seed seed 4096 Nov 23 09:40 ..
-rw-rw-r-- 1 seed seed 2400 Nov 23 12:02 MaliciousCode.smali
drwxrwxr-x 3 seed seed 4096 Nov 23 09:40 mobiseed
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/smali/com$ cd ..
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/smali$ ls
android  com
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/smali$ cd ..
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab$ ls
AndroidManifest.xml  apktool.yml  build  dist  original  res  smali
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab$ vi AndroidManifest.xml
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab$ cd ..
seed@MobiSEEDUbuntu:~/Downloads$ apktool b RepackagingLab
I: Using Apktool 2.1.0
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building apk file...
I: Copying unknown files/dir...
seed@MobiSEEDUbuntu:~/Downloads$ cd RepackagingLab/dist/
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ ls
RepackagingLab.apk
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ jarsigner -keystore ~/Downloads/mykey.keystore RepackagingLa
b.apk mykey
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 va
lidate this jar after the signer certificate's expiration date (2019-02-21) or after any future revocation date.
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ ls
RepackagingLab.apk
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ ls -al
```

In the figure below, we see the signing of the package with our certificate.



```
seed@MobiSEEDUbuntu: ~/Downloads/RepackagingLab/dist
seed@MobiSEEDUbuntu:~/Downloads$ ls
MaliciousCode.smali  mykey.keystore  RepackagingLab  RepackagingLab.apk.org
seed@MobiSEEDUbuntu:~/Downloads$ ls
MaliciousCode.smali  mykey.keystore  RepackagingLab  RepackagingLab.apk.org
seed@MobiSEEDUbuntu:~/Downloads$ ls RepackagingLab/dist/
RepackagingLab.apk
seed@MobiSEEDUbuntu:~/Downloads$ cd RepackagingLab/dist/
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ ls
RepackagingLab.apk
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ ls -al
total 1376
drwxrwxr-x 2 seed seed    4096 Nov 23 09:57 .
drwxrwxr-x 7 seed seed    4096 Nov 23 09:50 ..
-rw-rw-r-- 1 seed seed 1398743 Nov 23 09:57 RepackagingLab.apk
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ date
Fri Nov 23 09:59:09 EST 2018
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ jarsigner -keystore mykey.keystore RepackagingLab.pak mykey
Enter Passphrase for keystore:
jarsigner error: java.lang.RuntimeException: keystore load: /home/seed/Downloads/RepackagingLab/dist/mykey.keysto
re (No such file or directory)
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ cd ..
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab$ cd ..
seed@MobiSEEDUbuntu:~/Downloads$ ls
MaliciousCode.smali  mykey.keystore  RepackagingLab  RepackagingLab.apk.org
seed@MobiSEEDUbuntu:~/Downloads$ cd RepackagingLab/dist/
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ jarsigner -keystore ~/Downloads/mykey.keystore RepackagingLa
b.pak mykey
Enter Passphrase for keystore:
jarsigner: unable to open jar file: RepackagingLab.pak
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ jarsigner -keystore ~/Downloads/mykey.keystore RepackagingLa
b.apk mykey
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 va
lidate this jar after the signer certificate's expiration date (2019-02-21) or after any future revocation date.
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$
```

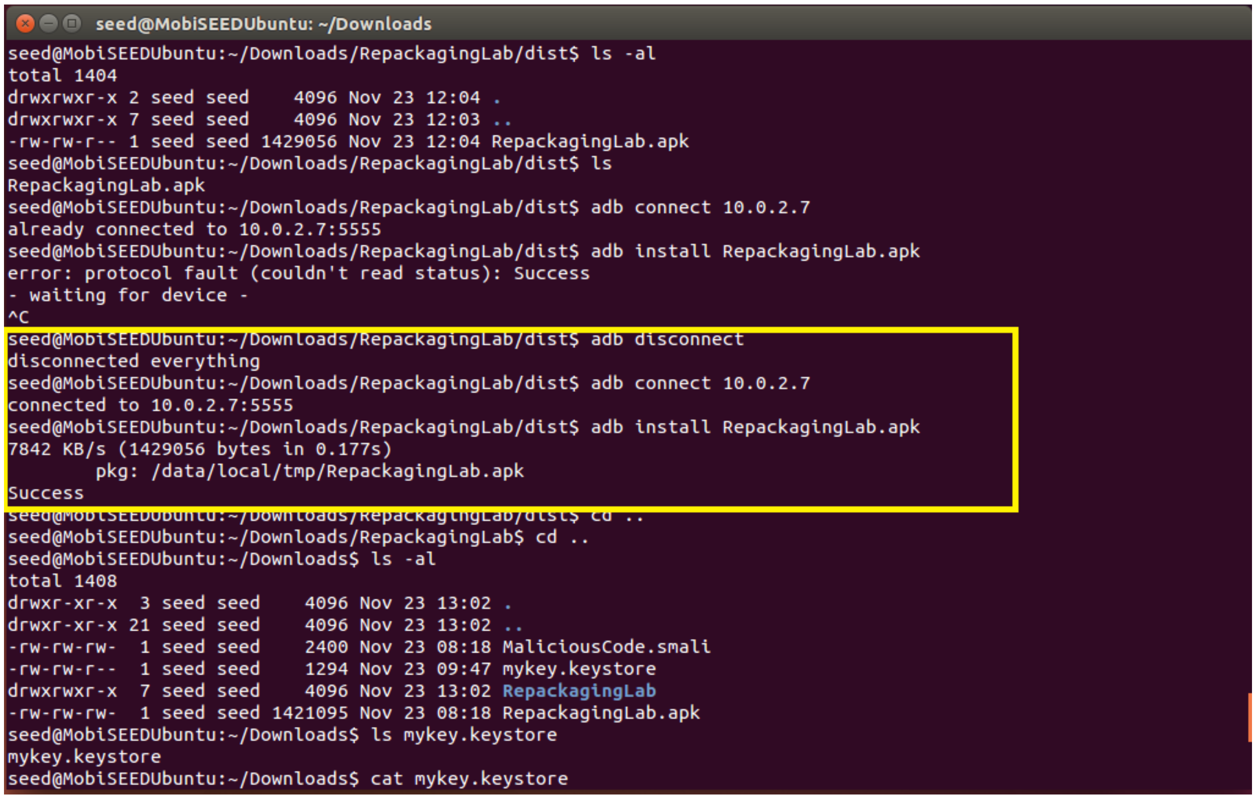
Observations / Explanation

In this task we 1) created a self-signed certificate, 2) regenerated the package and 3) signed the new package with the self-signed certificate. By accomplishing these steps, we were able to re-package our malicious code with updated manifest file and fully-prepare it (by signing it) so that we can install it.

Task 5: Install and Reboot

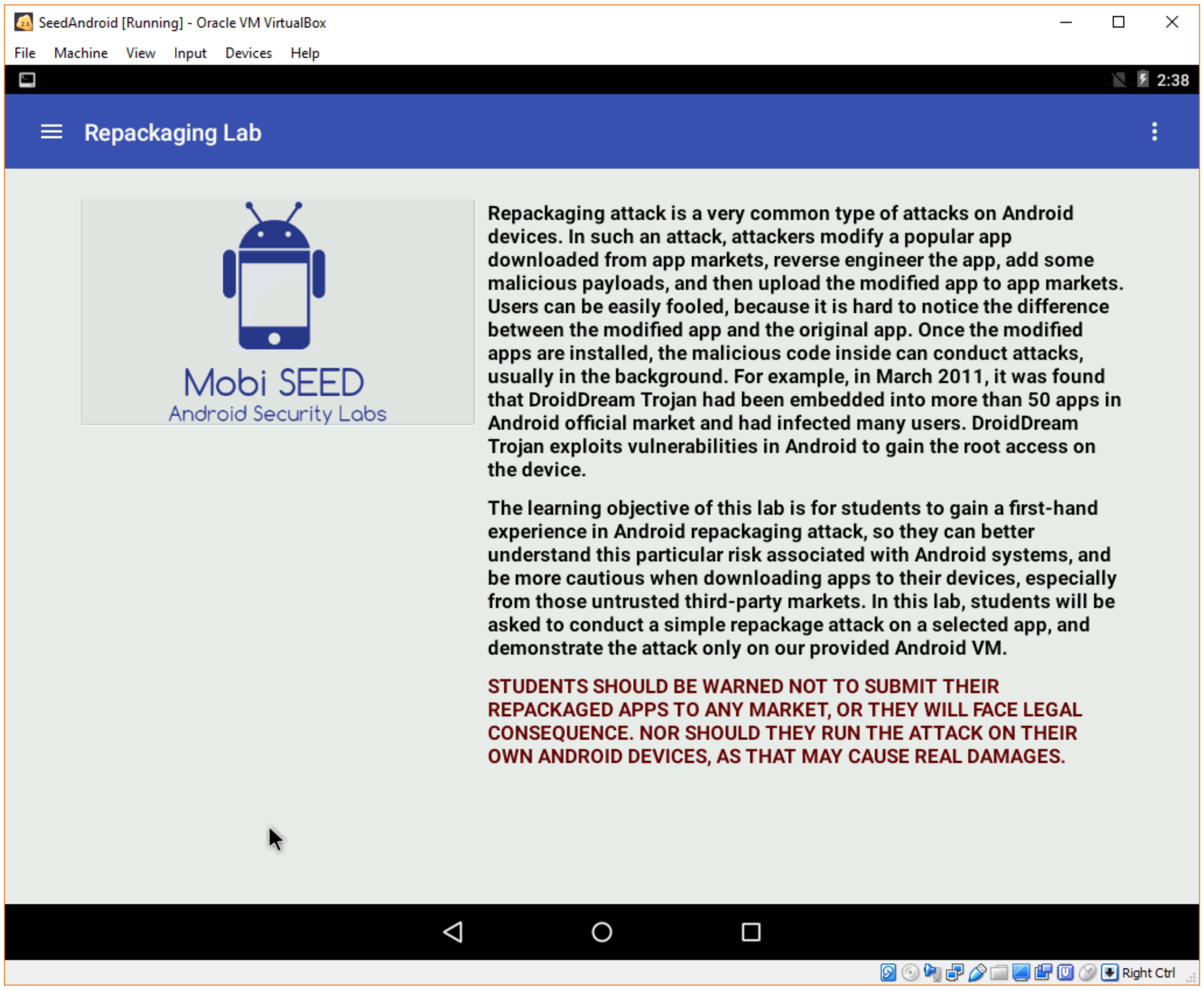
In the figure below, we can se the "adb disconnect" and "adb connect" which connects to the IP address of the Android target. Please note, the Android target is 10.0.2.7. After the connection, we see the "adb install" which installs the RepackagingLab.apk.

Please note, I previously deleted the RepackagingLab application from Android, so I did not have to do the "adb install -r" to replace a previous package.



```
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ ls -al
total 1404
drwxrwxr-x 2 seed seed    4096 Nov 23 12:04 .
drwxrwxr-x 7 seed seed    4096 Nov 23 12:03 ..
-rw-rw-r-- 1 seed seed 1429056 Nov 23 12:04 RepackagingLab.apk
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ ls
RepackagingLab.apk
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ adb connect 10.0.2.7
already connected to 10.0.2.7:5555
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ adb install RepackagingLab.apk
error: protocol fault (couldn't read status): Success
- waiting for device -
^C
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ adb disconnect
disconnected everything
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ adb connect 10.0.2.7
connected to 10.0.2.7:5555
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ adb install RepackagingLab.apk
7842 KB/s (1429056 bytes in 0.177s)
        pkg: /data/local/tmp/RepackagingLab.apk
Success
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab/dist$ cd ..
seed@MobiSEEDUbuntu:~/Downloads/RepackagingLab$ cd ..
seed@MobiSEEDUbuntu:~/Downloads$ ls -al
total 1408
drwxr-xr-x  3 seed seed    4096 Nov 23 13:02 .
drwxr-xr-x 21 seed seed    4096 Nov 23 13:02 ..
-rw-rw-rw-  1 seed seed    2400 Nov 23 08:18 MaliciousCode.smali
-rw-rw-r--  1 seed seed    1294 Nov 23 09:47 mykey.keystore
drwxrwxr-x  7 seed seed    4096 Nov 23 13:02 RepackagingLab
-rw-rw-rw-  1 seed seed 1421095 Nov 23 08:18 RepackagingLab.apk
seed@MobiSEEDUbuntu:~/Downloads$ ls mykey.keystore
mykey.keystore
seed@MobiSEEDUbuntu:~/Downloads$ cat mykey.keystore
```

After installation, on the Android VM, we run the RepackagingLab application to register the receiver.

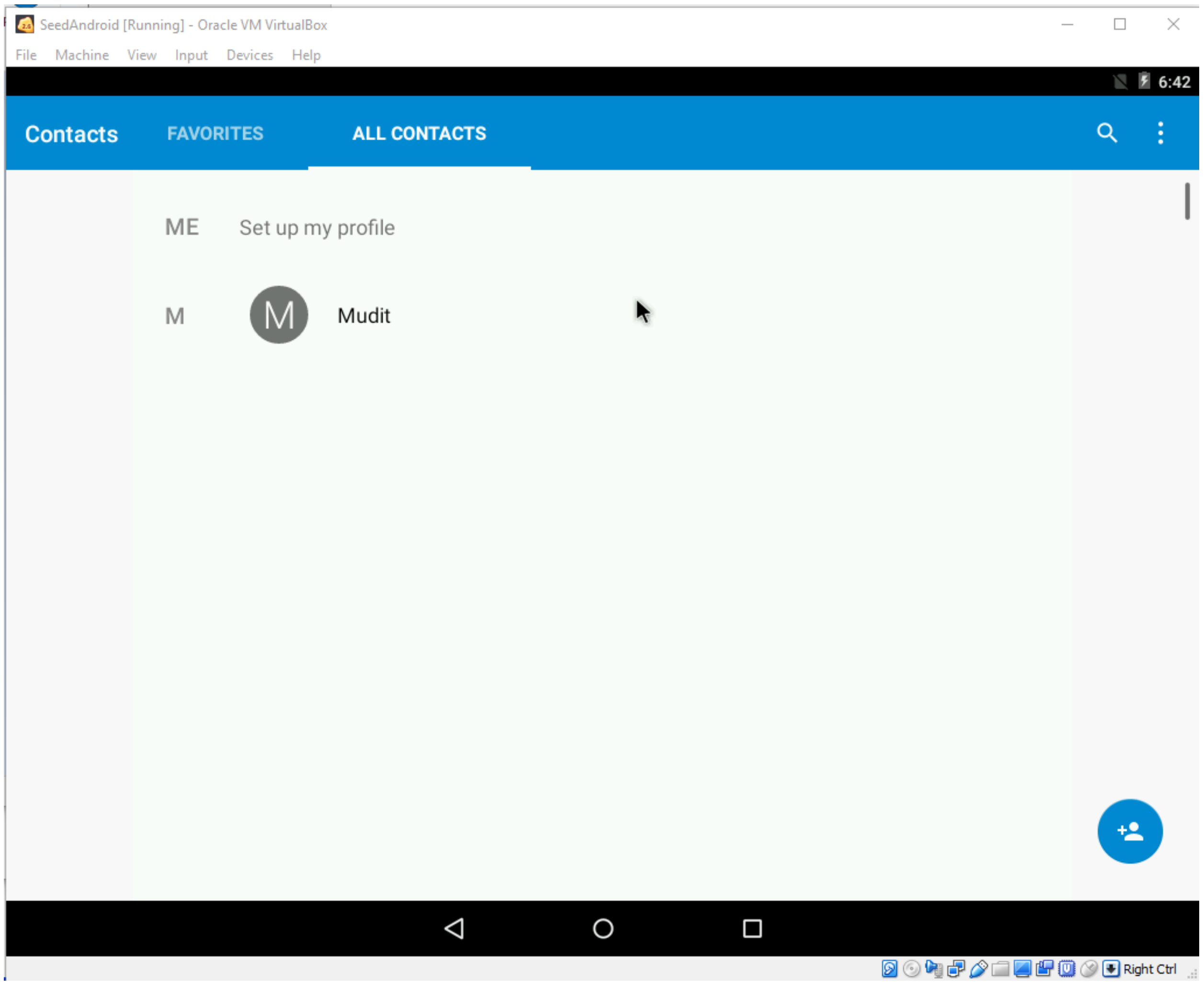

SeedAndroid [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
2:38
Repackaging Lab
Mobi SEED
Android Security Labs
Right Ctrl

Repackaging attack is a very common type of attacks on Android devices. In such an attack, attackers modify a popular app downloaded from app markets, reverse engineer the app, add some malicious payloads, and then upload the modified app to app markets. Users can be easily fooled, because it is hard to notice the difference between the modified app and the original app. Once the modified apps are installed, the malicious code inside can conduct attacks, usually in the background. For example, in March 2011, it was found that DroidDream Trojan had been embedded into more than 50 apps in Android official market and had infected many users. DroidDream Trojan exploits vulnerabilities in Android to gain the root access on the device.

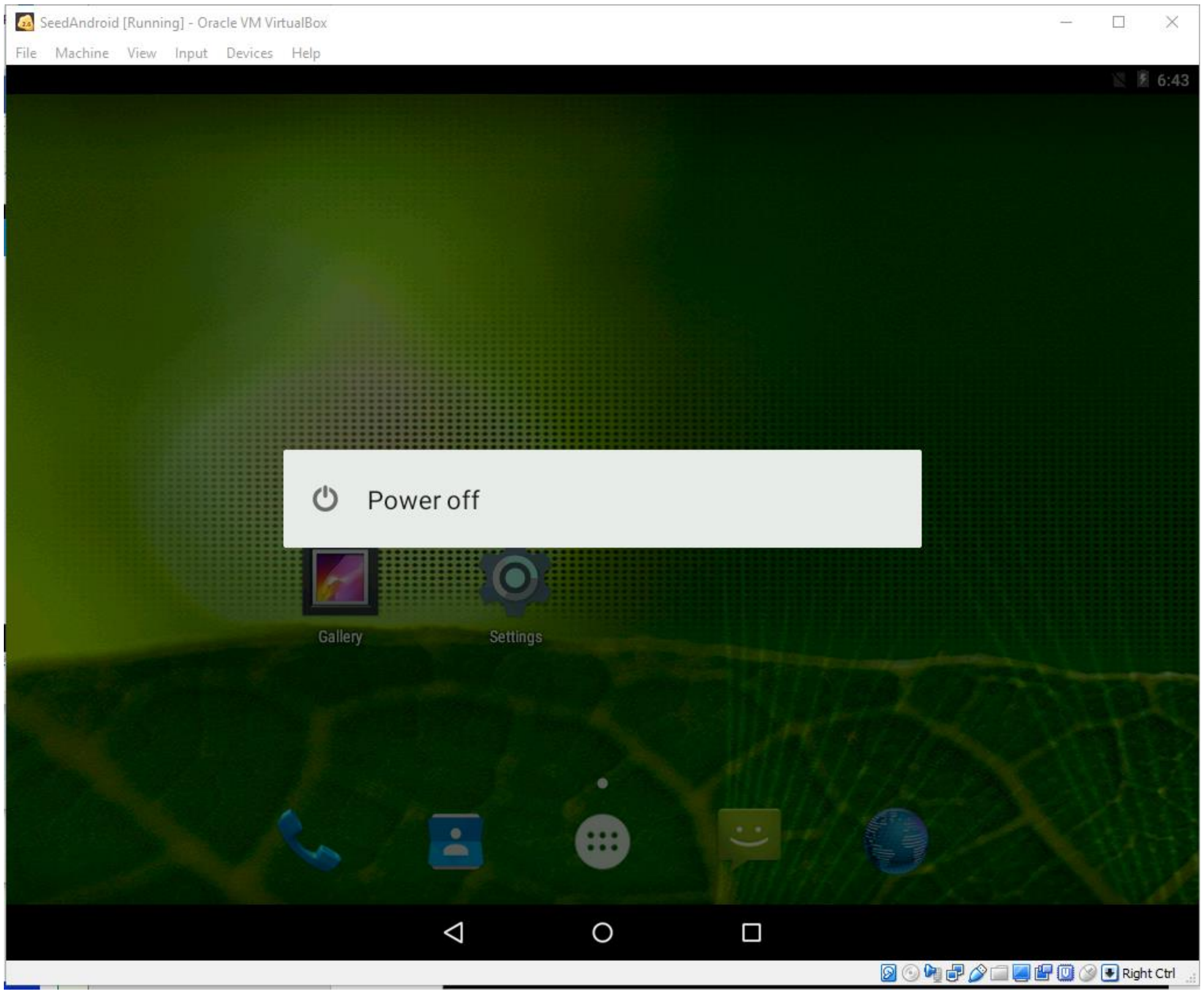
The learning objective of this lab is for students to gain a first-hand experience in Android repackaging attack, so they can better understand this particular risk associated with Android systems, and be more cautious when downloading apps to their devices, especially from those untrusted third-party markets. In this lab, students will be asked to conduct a simple repackage attack on a selected app, and demonstrate the attack only on our provided Android VM.

STUDENTS SHOULD BE WARNED NOT TO SUBMIT THEIR REPACKAGED APPS TO ANY MARKET, OR THEY WILL FACE LEGAL CONSEQUENCE. NOR SHOULD THEY RUN THE ATTACK ON THEIR OWN ANDROID DEVICES, AS THAT MAY CAUSE REAL DAMAGES.

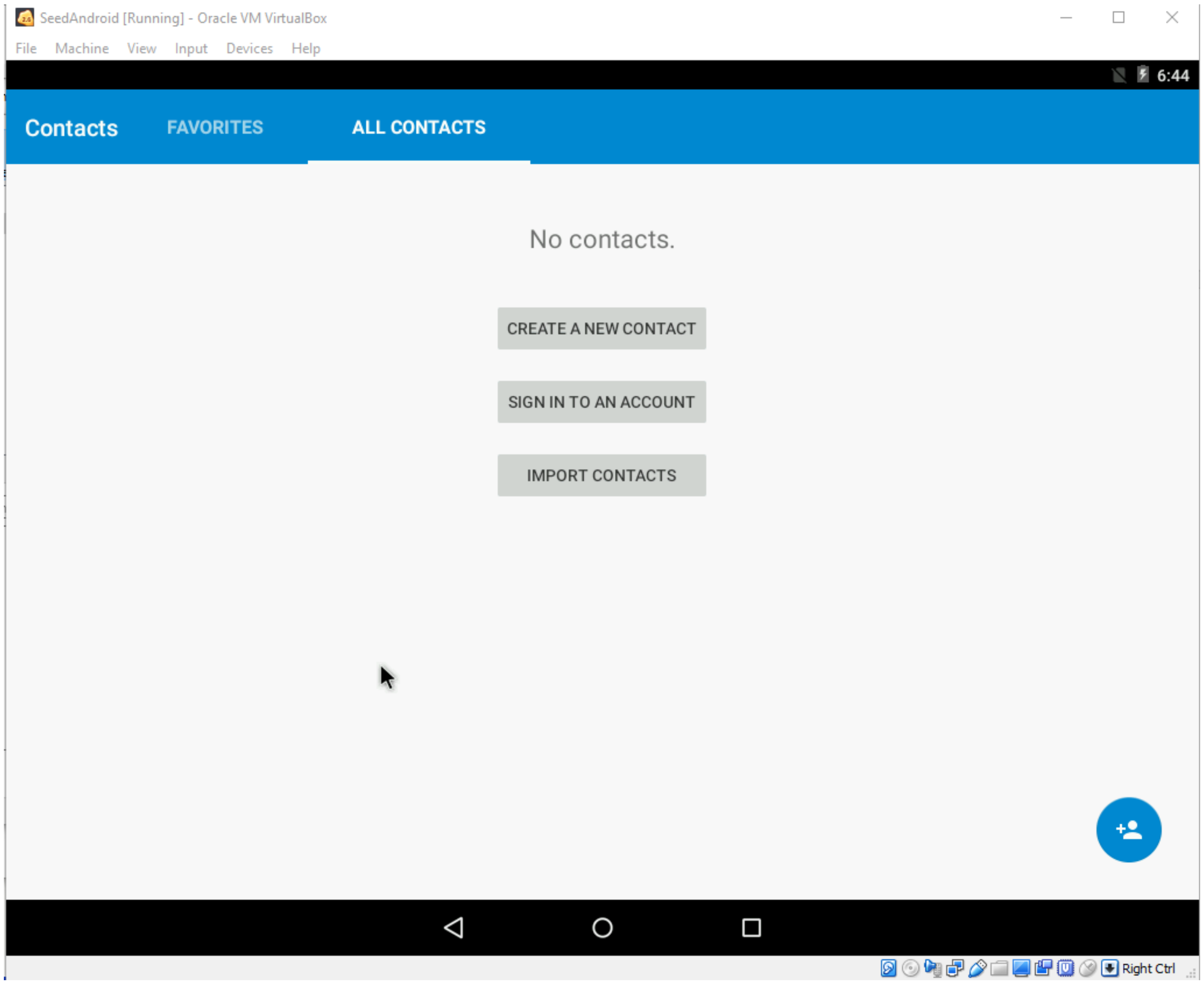
In the figure below, we create a new contact “Mudit”.


SeedAndroid [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
6:42
Contacts
FAVORITES
ALL CONTACTS
ME
Set up my profile
M
Mudit
Right Ctrl

In the figure below, we shutdown the VM. Please note, this is the step before the actual shutdown. When you click on "Power off", the shutdown happens very quickly such that releasing the mouse and getting a screen shot in that time frame is very difficult. Rest assured, the machine was shutdown and restarted.


SeedAndroid [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
6:43
Power off
Gallery
Settings
Right Ctrl

In the figure below, we see that there are no contacts.



Observations / Explanation

We ran the application once to register the event handler, so that MaliciousCode runs when the boot is complete. Further, we created a new contact to test that the MaliciousCode did run. Recall, the MaliciousCode will delete all contacts when executed. We can see that after the shutdown and restart of the Android VM, all contacts (i.e. "Mudit" contact) was deleted.

Bottom-line, installing a repackaged application with a new malicious class that runs independently of the original application was very easy to do. Triggering off the BOOT_COMPLETED was an easy way to kick-off the attack. While this attack itself isn't that interesting – it's "bad", but it could be a lot worse – it still demonstrates the steps / mechanics involved in executing this type of attack…. which can be very dangerous!

Questions

Question 1

Why is the repackaging attack not much a risk in iOS devices?

Answer
iOS devices only have Apple's App Store. There are no third party app stores. This attack relies on third party app stores, where an APK can be re-signed and posted to one of those stores.

Question 2
If you were Google, what decisions you would make to reduce the attacking chances of repackaging attacks?

Answer
Stop allowing third party app stores! If they do support third party app stores, they need a way to vet developers and only allow vetted-trusted developers to upload their applications to the third party app stores. Root of trust should still be in Google or the third party app store owner's hands, but more effort needs to go in place to not allow rogue app stores or developers from simply creating malicious applications and uploading to third party stores. Also, more security testing of the applications needs to go into place, to ensure malicious application do not make it to the app store for public consumption.

Despite the best efforts, malicious code can likely still end-up in an app store, especially considering Google is more "open" then Apple. Still, more effort can be put underway to scrutinize developers, hold app store owners more accountable for the apps on their stores and more security testing of applications themselves.

Question 3
Third-party markets are considered as the major source of repackaged applications. Do you think that using the official Google Play Store only can totally keep you away from the attacks? Why or why not?

Answer
Using only the Google App Store helps control the potential distribution of malware, BUT, it's not the end-all solution to stop all attacks. A malicious developer can be clever and still publish an app and it may take some time for Google to discover the attack. By that time, the attack may have infected / compromised many systems.

Question 4
In real life, if you had to download applications from untrusted source, what would you do to ensure the security of your device?

Answer
I wouldn't install from an untrusted source.

But… if I had to, I would look at the certificate of the signed package and see who signed it and who the certificate authority is. I would do my best to verify that this certificate is from who they say they are and that the CA is legitimate. Based on that I would decide if it's safe enough to download/install.

Further, I would not give the application any additional permissions to the device beyond the application itself; i.e. quarantine it to its JAVA sandbox and limit it's exposure to the file system, devices, user data (contacts, photos, calendar etc).

Demonstration

This lab was demonstrated in class by Mudit Vats on 11/23/19. Please see recorded session for actual demonstration. Thank you!