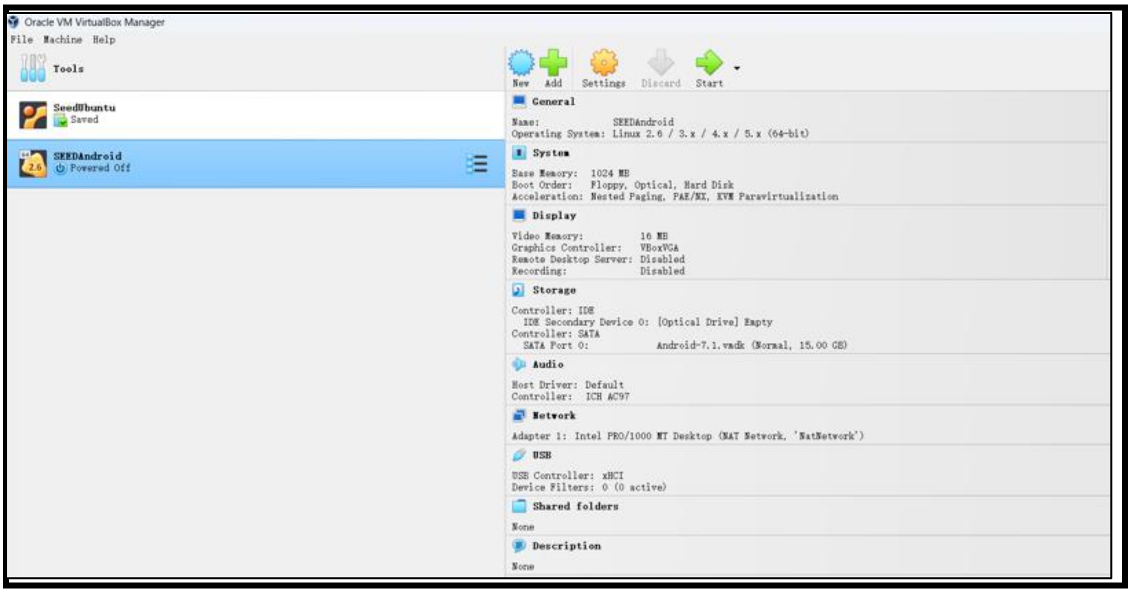
Name: Sushmitha Konduru
GNo: G01456225

ISA 562 HOMEWORK3

ANDROID REPACKAGING ATTACK LAB

=>Before we start, we need to create two virtual machines: SeedUbuntu 16.04-32 to insert the malicious code into and app and repackage it and SeedAndroid to execute the android repackaging attack



```
General
Name:             SEEDAndroid
Operating System: Linux 2.6 / 3.x / 4.x / 5.x (64-bit)
System
Base Memory:  1024 MB
Boot Order:   Floppy, Optical, Hard Disk
Acceleration: Nested Paging, PAE/NX, KVM Paravirtualization
Display
Video Memory:          16 MB
Graphics Controller:   VBoxVGA
Remote Desktop Server: Disabled
Recording:             Disabled
Storage
Controller: IDE
  IDE Secondary Device 0: [Optical Drive] Empty
Controller: SATA
  SATA Port 0:            Android-7.1.vmdk (Normal, 15.00 GB)
Audio
Host Driver: Default
Controller:  ICH AC97
Network
Adapter 1: Intel PRO/1000 MT Desktop (NAT Network, 'NatNetwork')
USB
USB Controller: xHCI
Device Filters: 0 (0 active)
Shared folders
None
Description
None
```

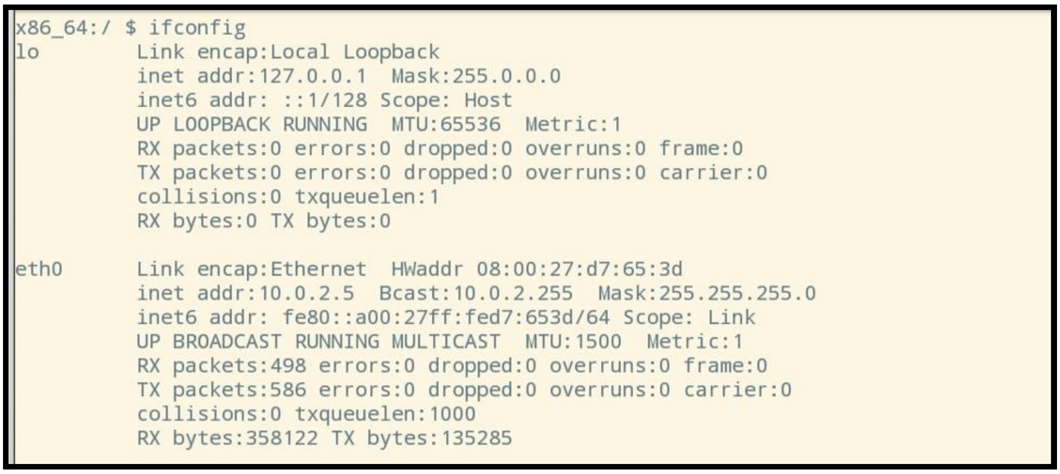
Fig: The created VM's for the lab

Lab Tasks

1)Task 1: Obtain An Android App (APK file) and Install It

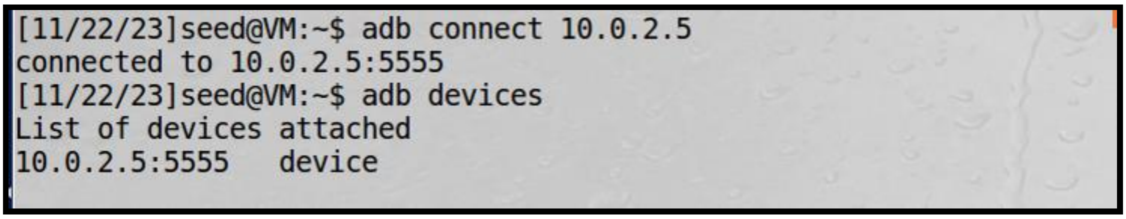
=>Download RepackagingLab.APK from SEEDLabs in SeedUbuntu and store it in a folder. Next, we find the IP address of seedAndroid and use it to connect to SeedAndroid and install the app into seedAndroid.

Name: Sushmitha Konduru
GNo: G01456225



```
x86_64:/ $ ifconfig
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:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 TX bytes:0

eth0      Link encap:Ethernet  HWaddr 08:00:27:d7:65:3d
          inet addr:10.0.2.5  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fed7:653d/64 Scope: Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:498 errors:0 dropped:0 overruns:0 frame:0
          TX packets:586 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:358122 TX bytes:135285
```

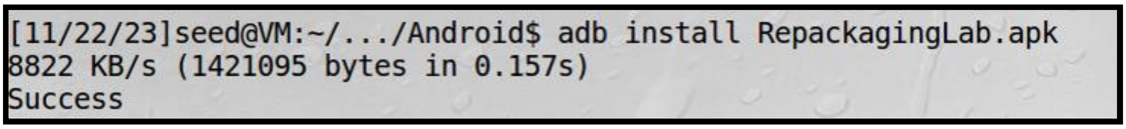
=>The IP address of SeedAndroid is 10.0.2.5.
=>We use the command "adb connect 10.0.2.5" to connect with SeedAndroid



```
[11/22/23]seed@VM:~$ adb connect 10.0.2.5
connected to 10.0.2.5:5555
[11/22/23]seed@VM:~$ adb devices
List of devices attached
10.0.2.5:5555   device
```

Fig: The IP address of AndroidVM

=>We install RepackagingLab.APK into SeedAndroid as shown below:

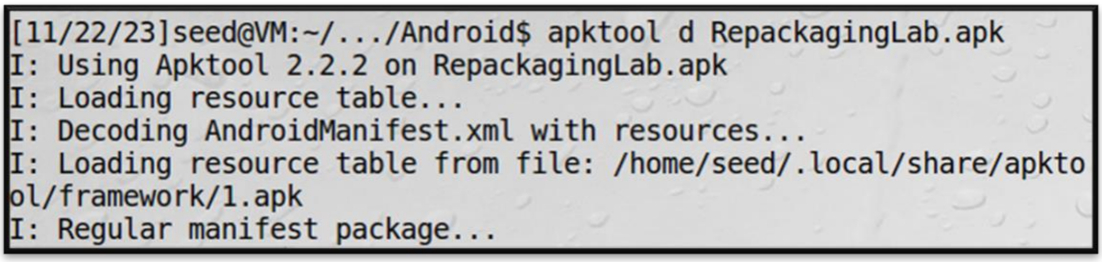


```
[11/22/23]seed@VM:~/.../Android$ adb install RepackagingLab.apk
8822 KB/s (1421095 bytes in 0.157s)
Success
```

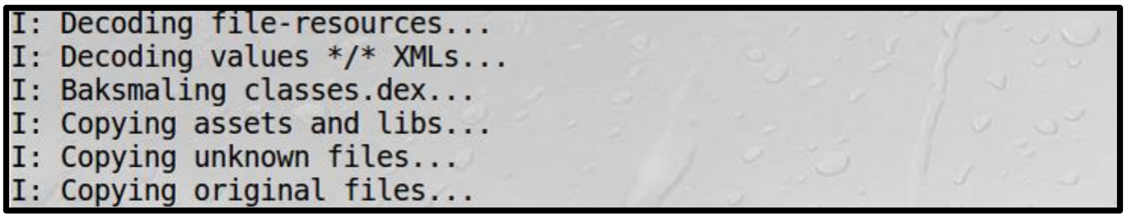
Fig: Installing RepackagingLab.APK in AndroidVM

2)Task 2: <u>Disassemble Android App</u>

=>The original RepackagingLab.APK is in a format that is not understandable by Humans therefore   we cannot modify it.So we disassemble the APK file into smali format.A tool named APKTool is used in this process.



```
[11/22/23]seed@VM:~/.../Android$ 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/apkto
ol/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...
```

Fig: Disassembling the APK

=>The APKTool disassembles RepackagingLab.APK into a folder with the same name and it contains the following files

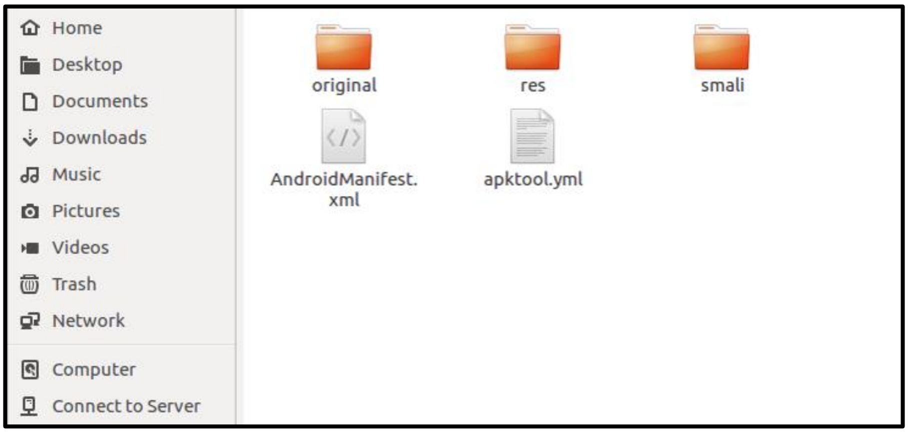
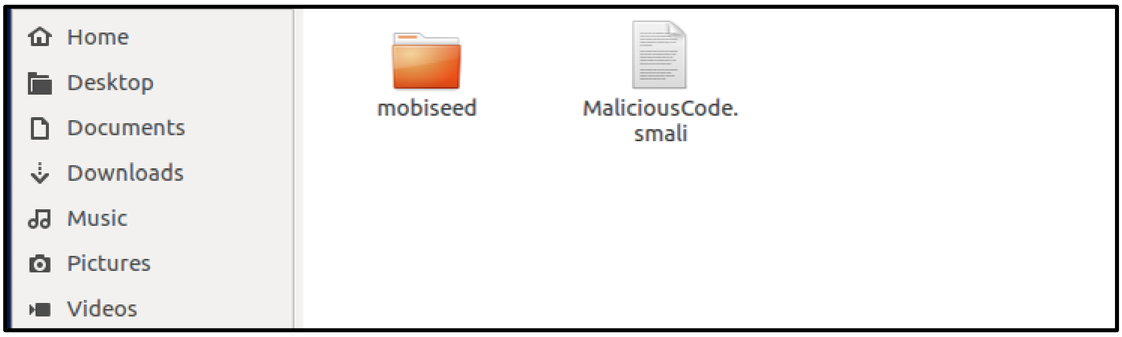


Fig: The disassembles APK

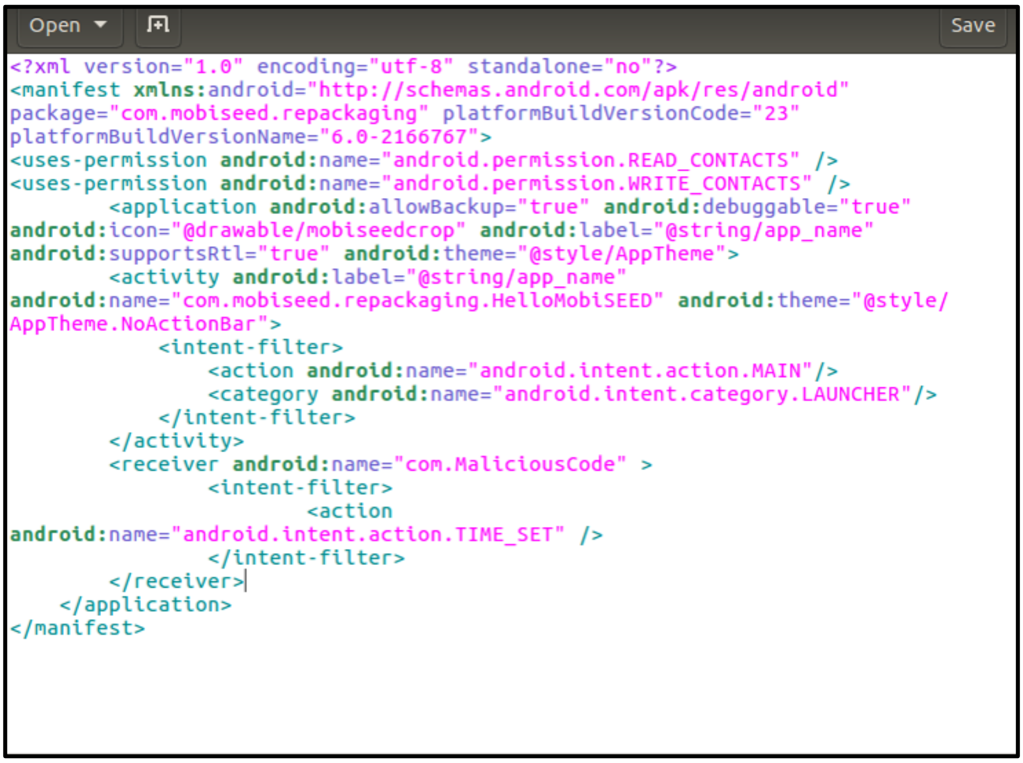
3)Task 3: <u>Inject Malicious Code</u>

=>We add an additional file to the disassembled app, which is the malicious code. The malicious code is such that when it is triggered by the broadcast receiver when the time changes in the VM it deletes all the contacts on the device.
=>The Malicious code is placed in the smali/com folder. Then we modify the AndroidManifest.xml to add permissions for this app to read and write into contacts and also register the trigger that will cause the malicious code to be executed.
=>In the below code the broadcast signal is taken as time set therefore when the time is set by the user it triggers to malicious code to execute the attack.



Name: Sushmitha Konduru
GNo: G01456225

Fig: Location of malicious code



```
<?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>
```

Fig: Modified AndroidManifest.xml file

4)Task 4: <u>Repack Android App with Malicious Code</u>

=>After inserting the malicious code file in to the smali folder, RepackagingLab.APK needs to be reassembled into a single APK file. The reassembling of code can be performed using the APKTool again. when this command is executed, it reassembles the APK and saves it in a file called dist.

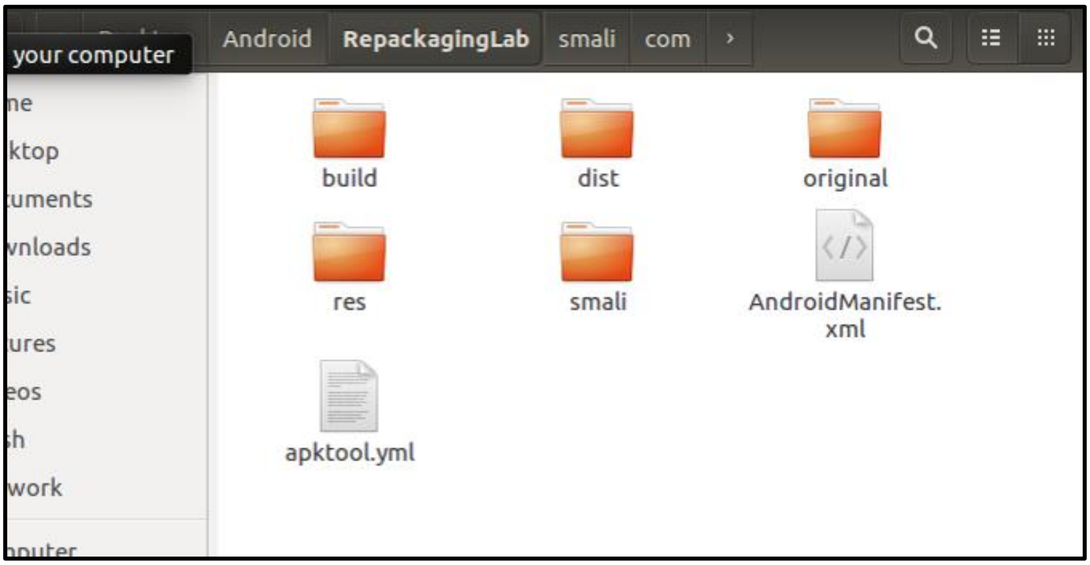


Fig: The newly created dist folder

Name: Sushmitha Konduru
GNo: G01456225

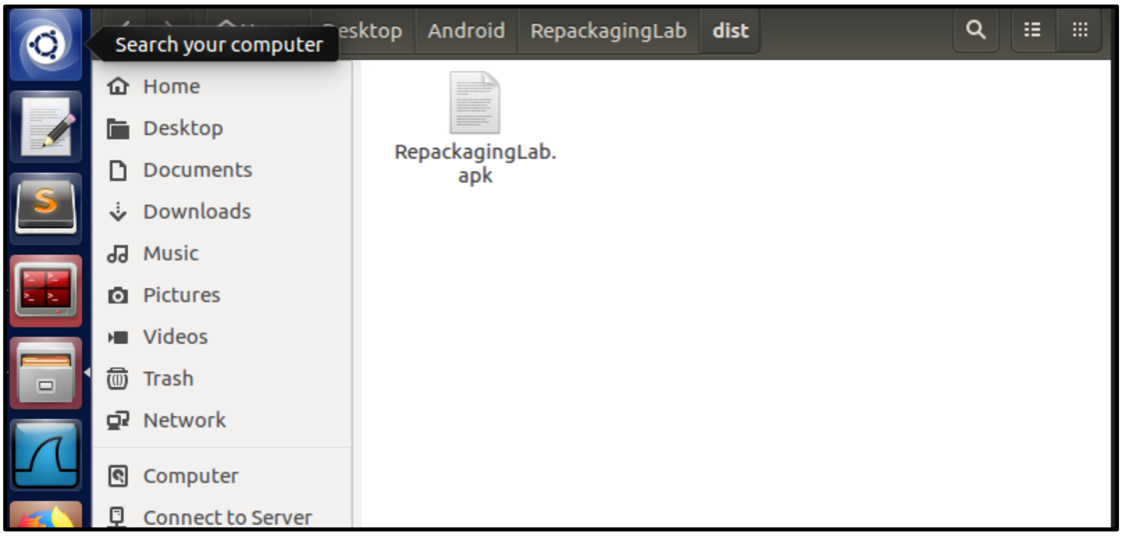
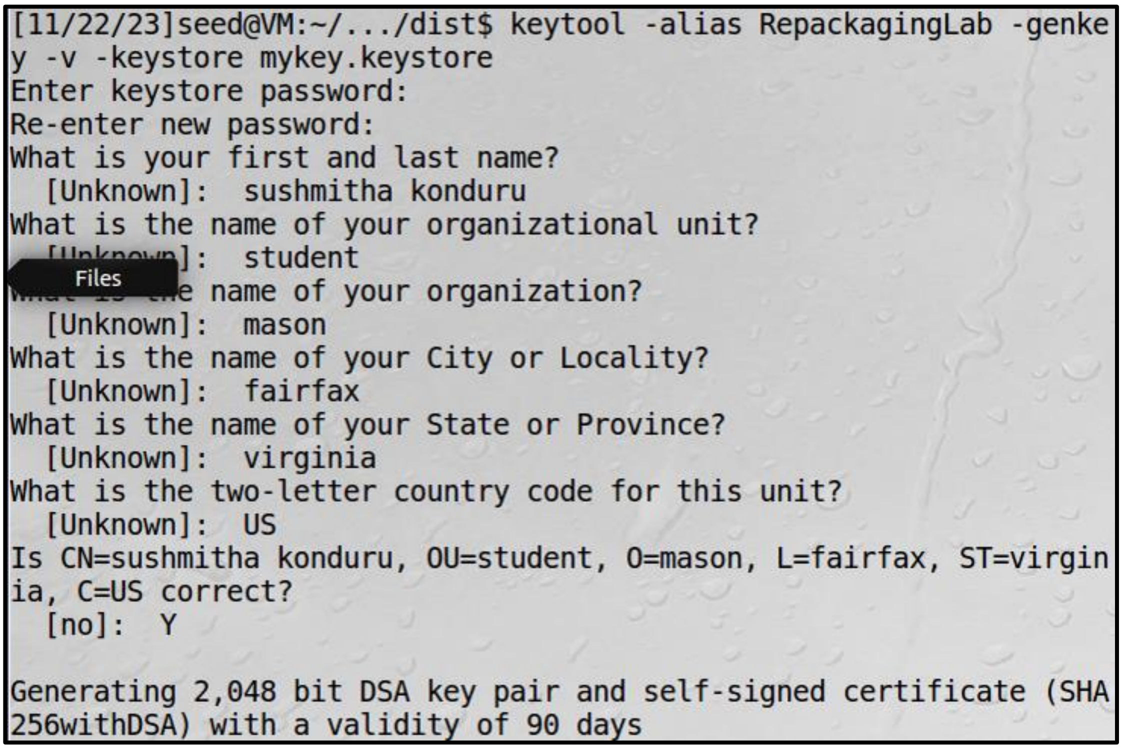


Fig: The repackaged APK file in dist folder

=>In order for this APK file to be installed it needs to be digitally signed. This is providing a digital signature and public key certificate for the APK. This process is performed using the “keytool” command.

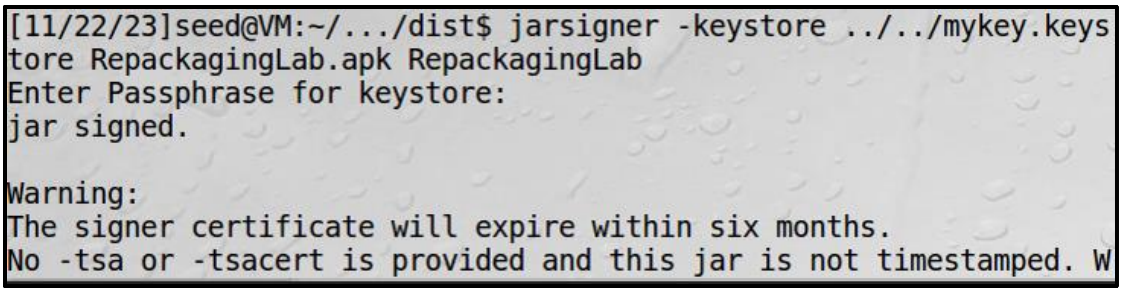


```
[11/22/23]seed@VM:~/.../dist$ keytool -alias RepackagingLab -genke
y -v -keystore mykey.keystore
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  sushmitha konduru
What is the name of your organizational unit?
  [Unknown]:  student
What is the name of your organization?
  [Unknown]:  mason
What is the name of your City or Locality?
  [Unknown]:  fairfax
What is the name of your State or Province?
  [Unknown]:  virginia
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=sushmitha konduru, OU=student, O=mason, L=fairfax, ST=virgin
ia, C=US correct?
  [no]:  Y

Generating 2,048 bit DSA key pair and self-signed certificate (SHA
256withDSA) with a validity of 90 days
```

Fig: Generates the public key and digital certificate

=>As shown in the above image keytool command prompts for a password which is used to protect the mykey-keystore file which stores the public and private keypair. The key pair is generated using the additional information provided by the user.

=>After generating the public-private keypair, we sign the APK file using “jarsigner”. This command prompts the user for a password needed to access the mykey. keystore then access the key and signs the APK file.



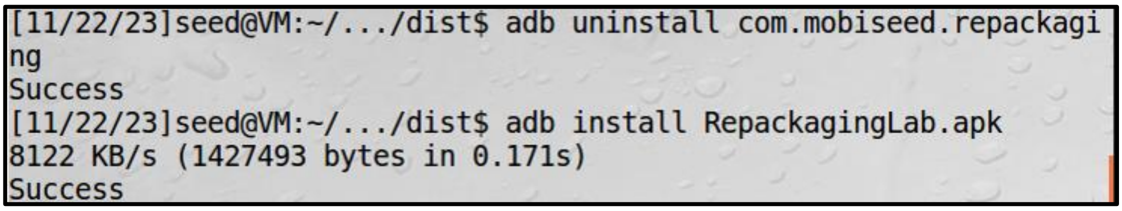
```
[11/22/23]seed@VM:~/.../dist$ jarsigner -keystore ../../mykey.keys
tore RepackagingLab.apk RepackagingLab
Enter Passphrase for keystore:
jar signed.

Warning:
The signer certificate will expire within six months.
No -tsa or -tsacert is provided and this jar is not timestamped. W
```

Fig: Signing the APK

=>The interesting observation from the above process is how easy it is to generate a digital signature for an APK file. Therefore, anyone can easily perform any malicious activity by inserting a piece of malicious code into the APK file .The Apps do not go through high levels of checking the contents of the app before making it available in app stores. So, anyone can create a malicious app have it self-certified without any third-party authority and make it available for the public.

5)Task 5: Install the repackaged App and trigger the Malicious Code

=>The repackaged APK containing of the malicious code can now be installed into seedAndroid using adb tool. Before we install this APK, we need to uninstall the app if it priorly installed in SeedAndroid.



```
[11/22/23]seed@VM:~/.../dist$ adb uninstall com.mobiseed.repackagi
ng
Success
[11/22/23]seed@VM:~/.../dist$ adb install RepackagingLab.apk
8122 KB/s (1427493 bytes in 0.171s)
Success
```

Fig: Uninstallation and installation of repackaged APK

Name: Sushmitha Konduru
GNo: G01456225

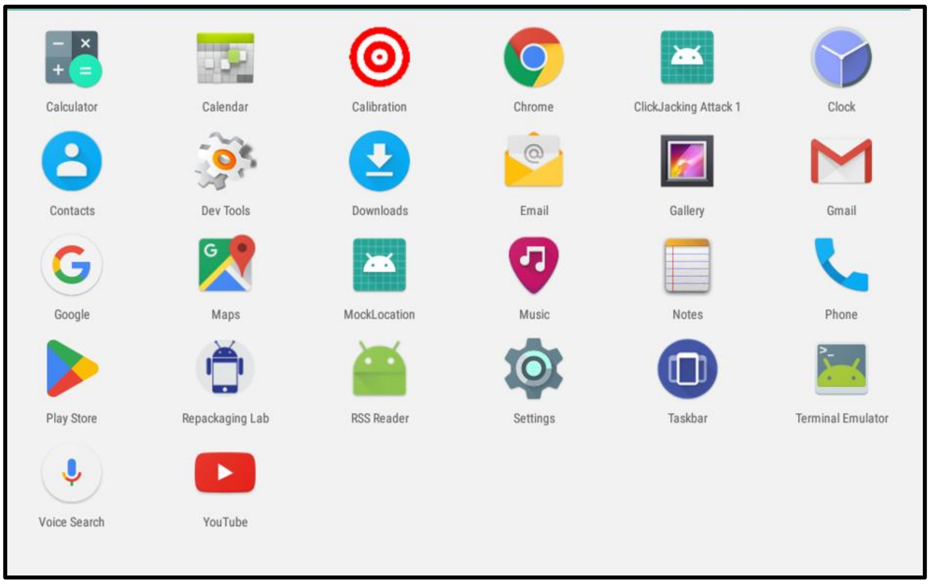
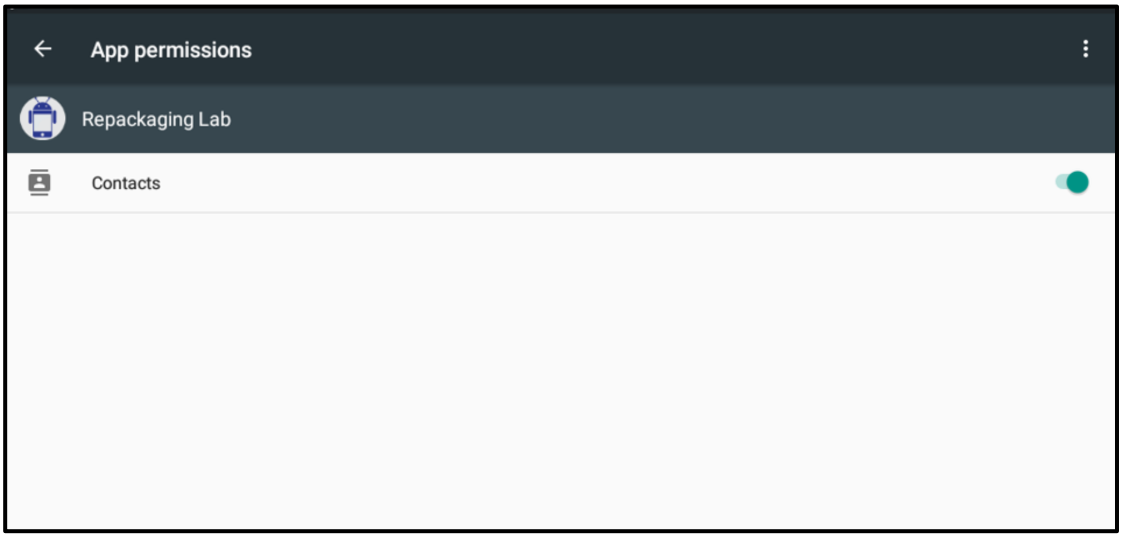


Fig: Installed Repackaged App in SEEDAndroid

=>After the installation we need to go to settings and grant the required the permissions. The App asks permission to access contacts which should be given in order to execute the attack.



App permissions

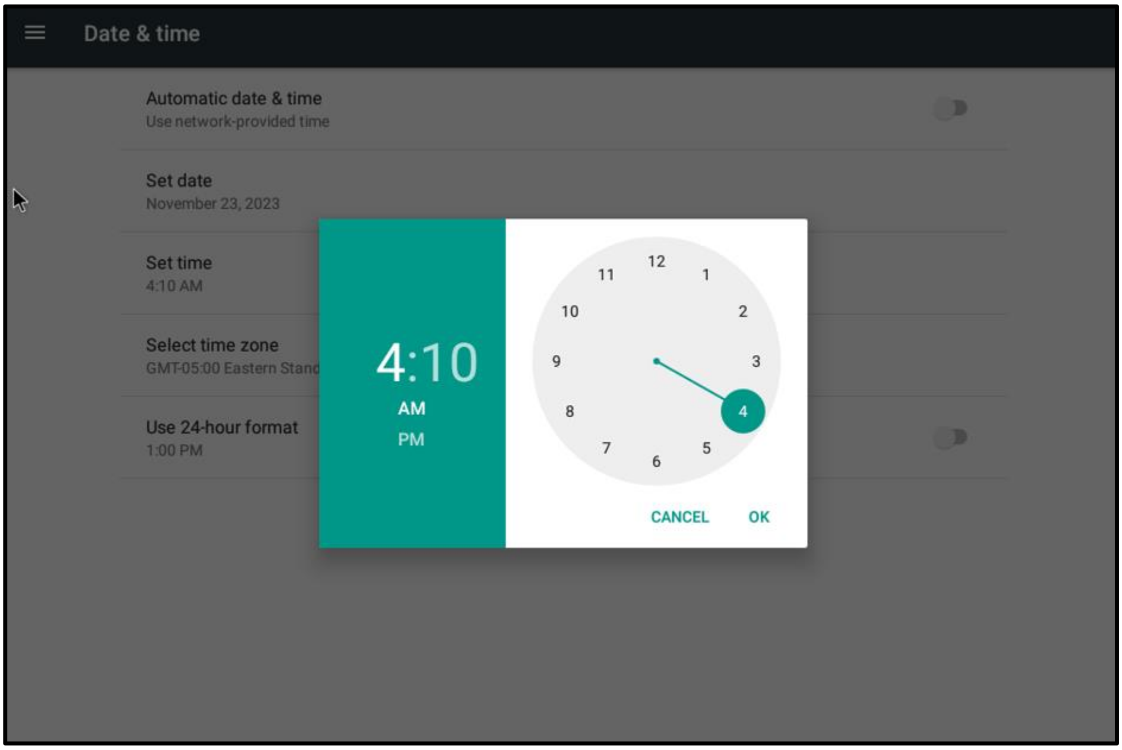
Repackaging Lab

Contacts

Fig: Granting required permissions for the App

=>Before triggering the Attack add some contacts in SEEDAndroid so that they can be deleted by the malicious code. To trigger the attack, we change the time on the Android VM.

Name: Sushmitha Konduru
GNo: G01456225



Date & time

Automatic date & time
Use network-provided time

Set date
November 23, 2023

Set time
4:10 AM

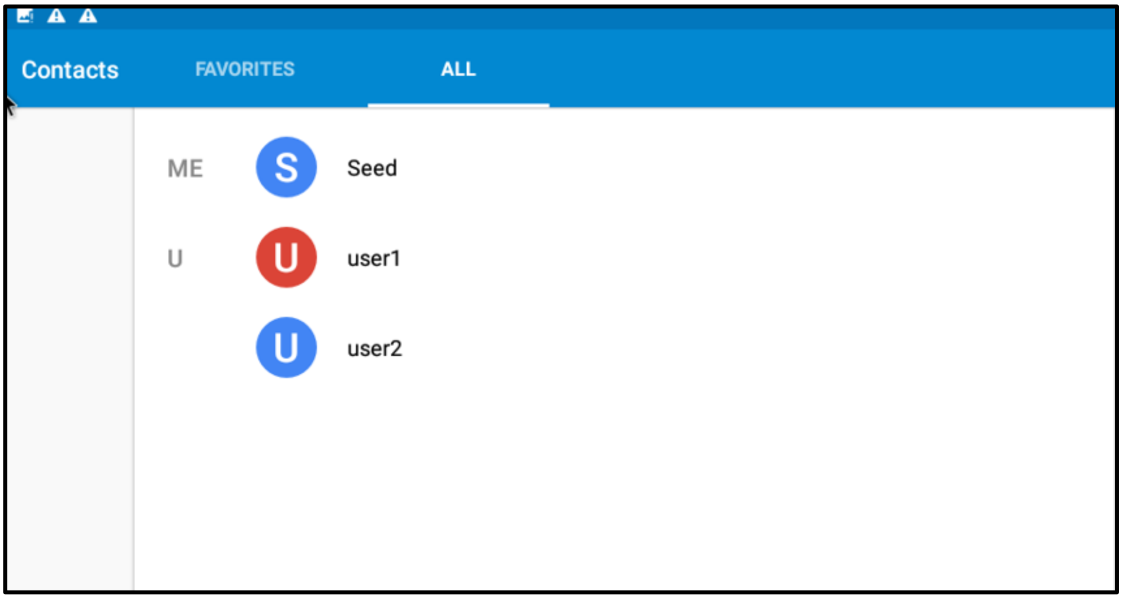
Select time zone
GMT-05:00 Eastern Stand

Use 24-hour format
1:00 PM

4:10
AM
PM

Fig: Changing the time on android to trigger the attack

=>On changing the time, the attack is triggered and we lose all the contacts on Android VM.



ME Seed

U user1

user2

Fig: Added contacts before triggering the attack

Name: Sushmitha Konduru
GNo: G01456225

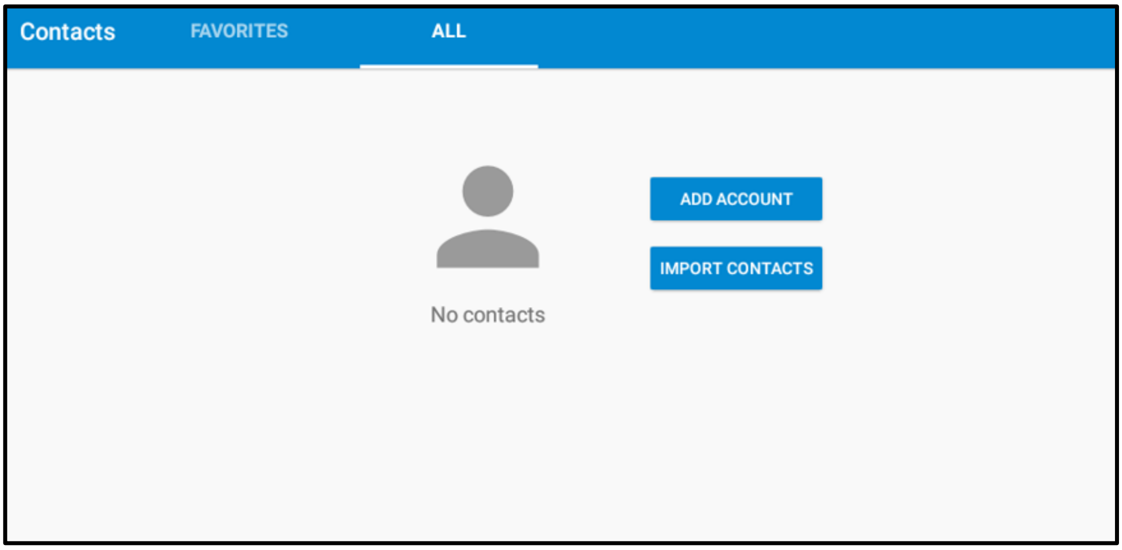


Fig: Contacts after changing the time

=>As shown in the above figure the attack has been successfully executed thereby deleting all the contacts on android VM.

SUMMARY:

=>All the tasks executed shows us the process of executing an Android Repackaging Attack. As seen above it is not a complicated process and therefore anyone can perform it.
=>Prominent App stores perform checks on the apps being available but other third-party store do not take these precautions so they are readily available for the vulnerable public.

=>Most of the applications may not have been signed by a certificate authority, they are just self-signed by the developers so the user should download any Application with care and be aware of the permissions being granted to the app. As seen above the malicious has access to the contacts and deleted them but in other cases the consequences may be severe leading to leak of sensitive information etc.