**CY-5005 - Advanced Information Security**

**ASSIGNMENT #01**

Group Members:

* Abdullah Khan

(24I-7732)

* Javeria Tabassum

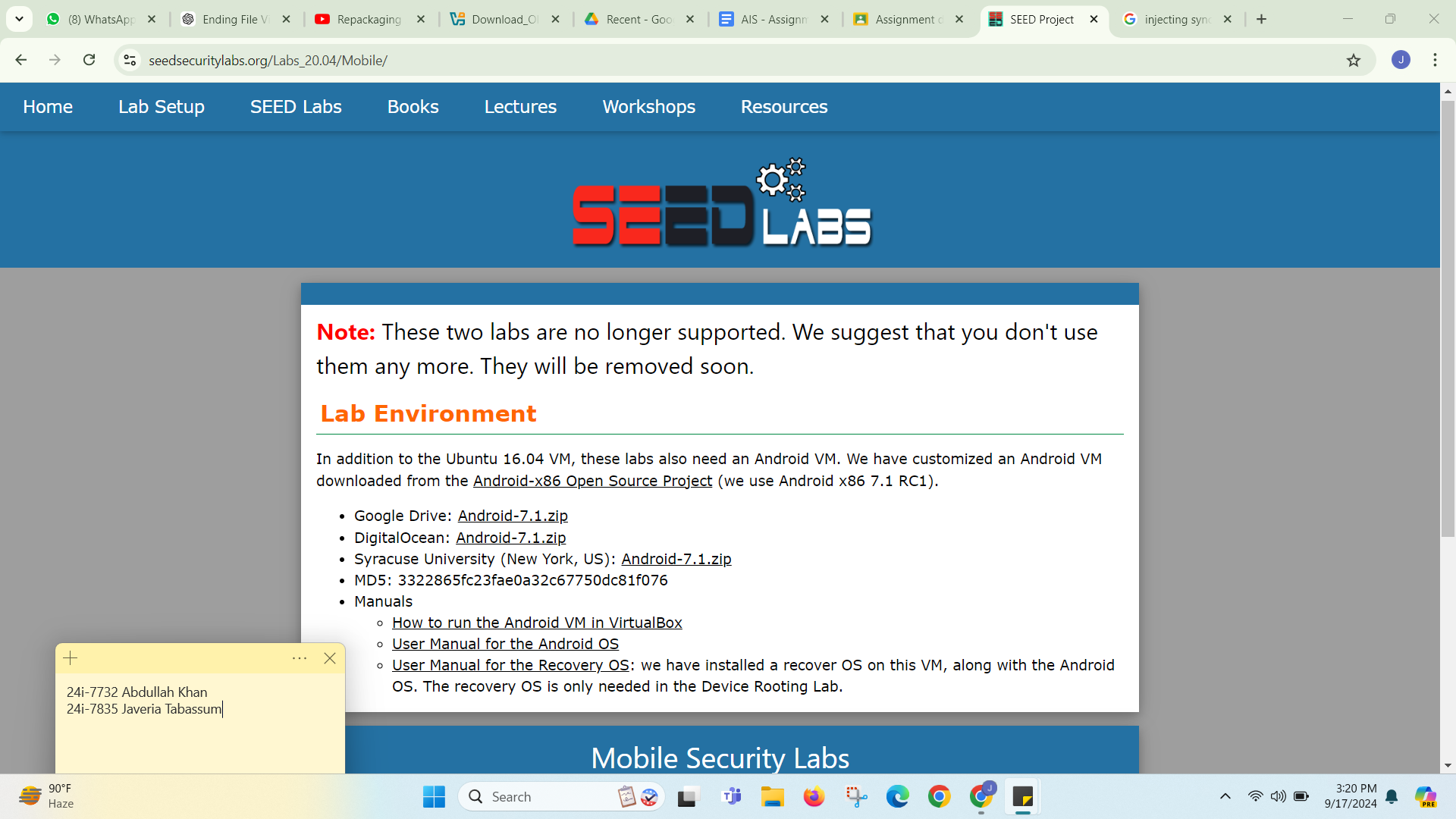
(24I-7835)

Title: **“Writing Android Malware”**

In this assignment, we will be manipulating an android application by inserting malicious code into it through SEED Lab’s Android Repackaging Attack Lab. We will be explaining each step we did and how we completed the process of repackaging an Android APK to inject malware.

<https://seedsecuritylabs.org/Labs_20.04/Mobile/Android_Repackaging/>

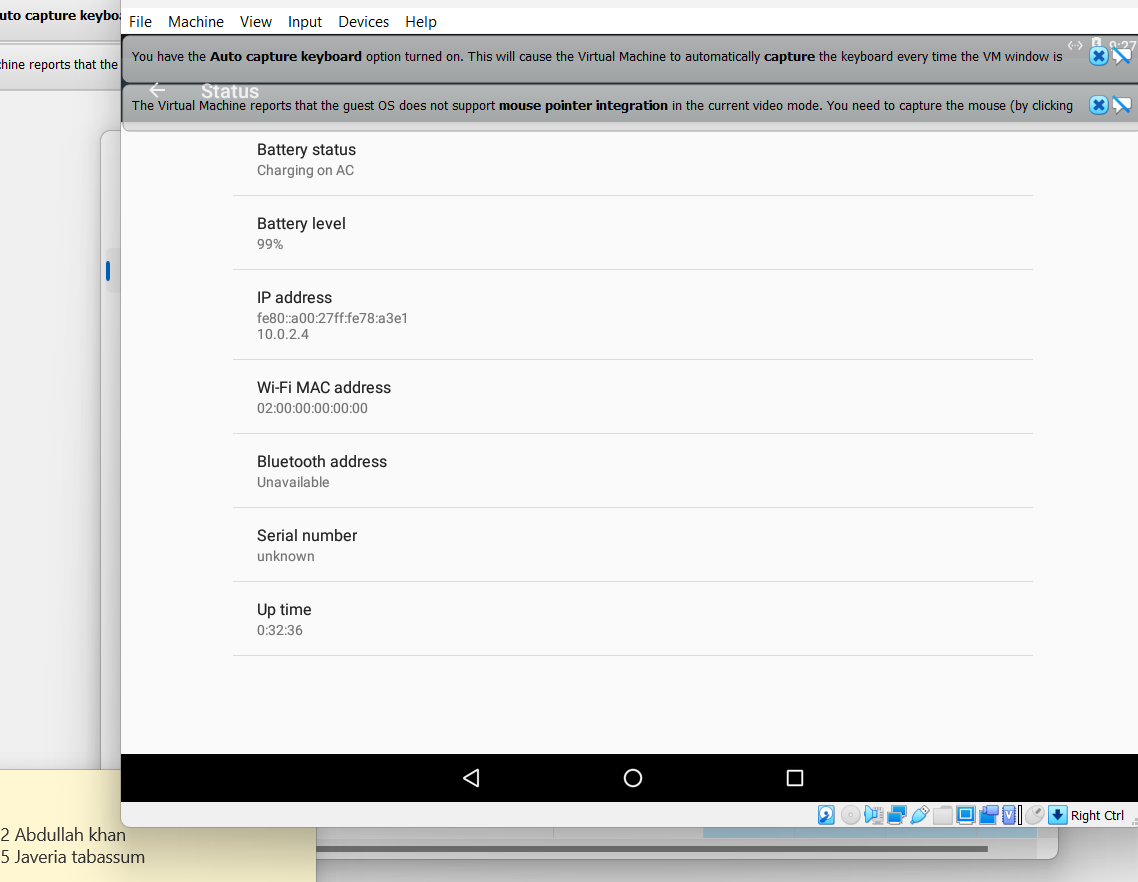
For the **Android Repackaging Attack Lab,** we will need both SEED Lab Ubuntu VM and the Android 7.1 VM. We downloaded both from the SEED Lab website and we will be running them using a hypervisor like VirtualBox.



**Task#01: “ Installing a Host Application”:**

The first step is to find the IP address of the android device we will be using.

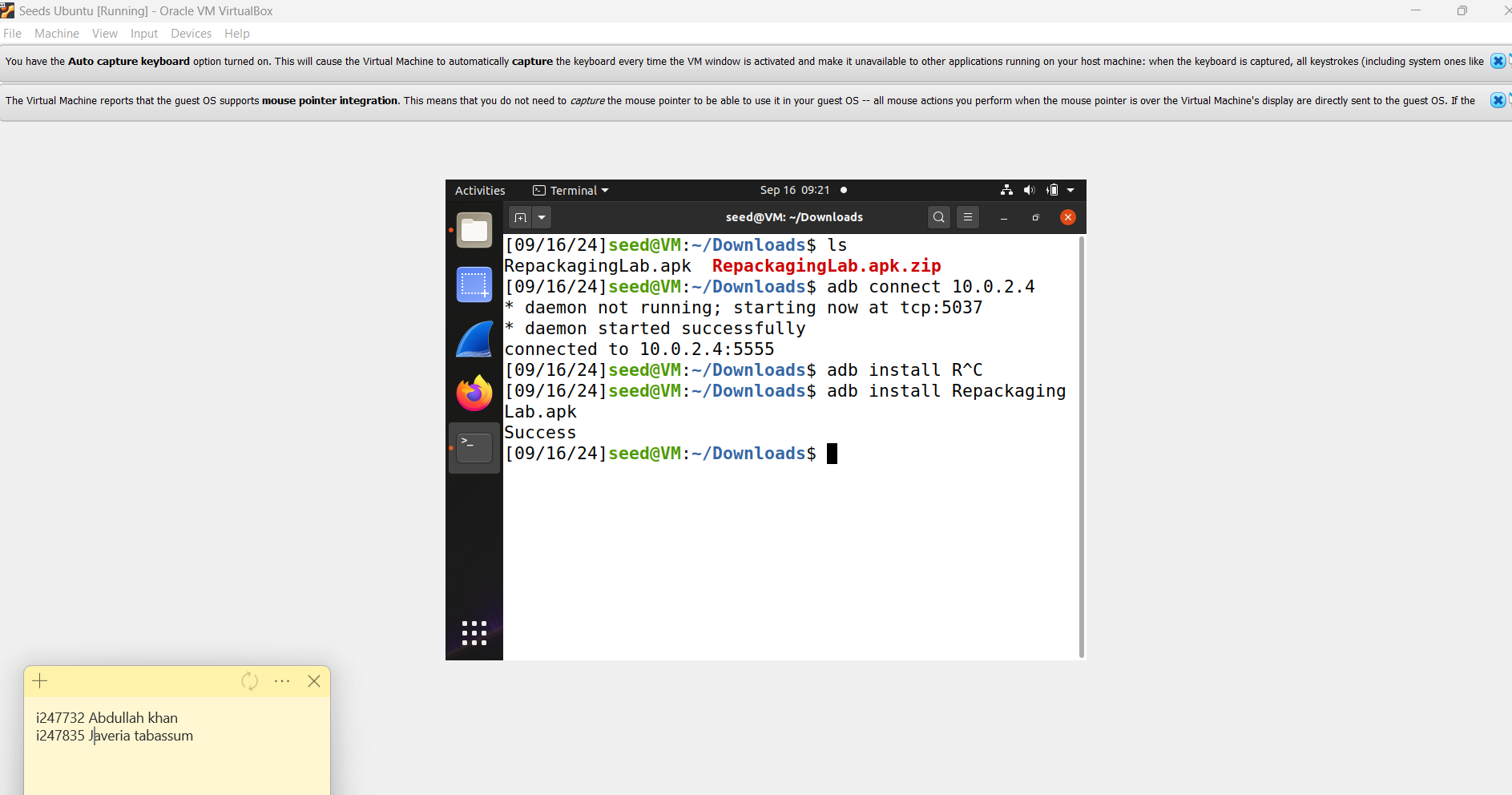
Get an APK File: We can choose the app from any source, but for the lab we will be using the RepackagingLab.apk file from SEED Labs.



**Installation:** The next step is to install an apk file from Ubuntu, to do that use the adb (Android Debug Bridge) command to install the APK file on the Android VM. Type the following command on terminal :

adb connect 10.0.2.4 ( to connect to our android device)

adb install RepackagingLab.apk (installing repackaging lab in our connected device)



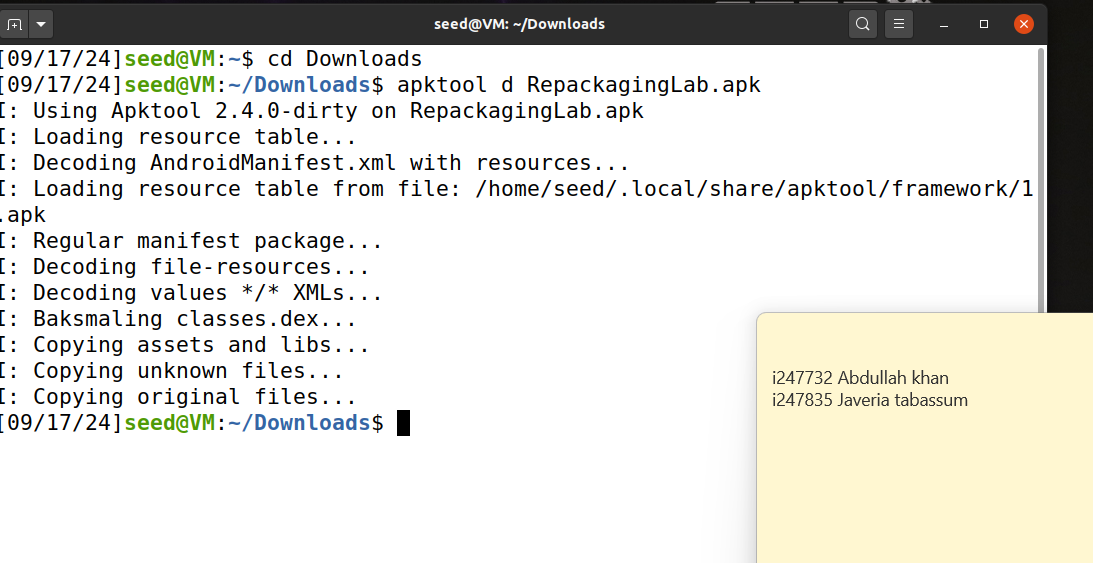
**Task#02: “ Disassembling the application”:**

The next step is to disassemble the app using the apk tool in order to access and modify the source code of the app. To do this, run the following command on terminal:

apktool d RepacakgingLab.apk

// (apktool d RepackagingLab.apk)

This command will decompile the apk into code and resources and that is where we will be injecting our malicious code.

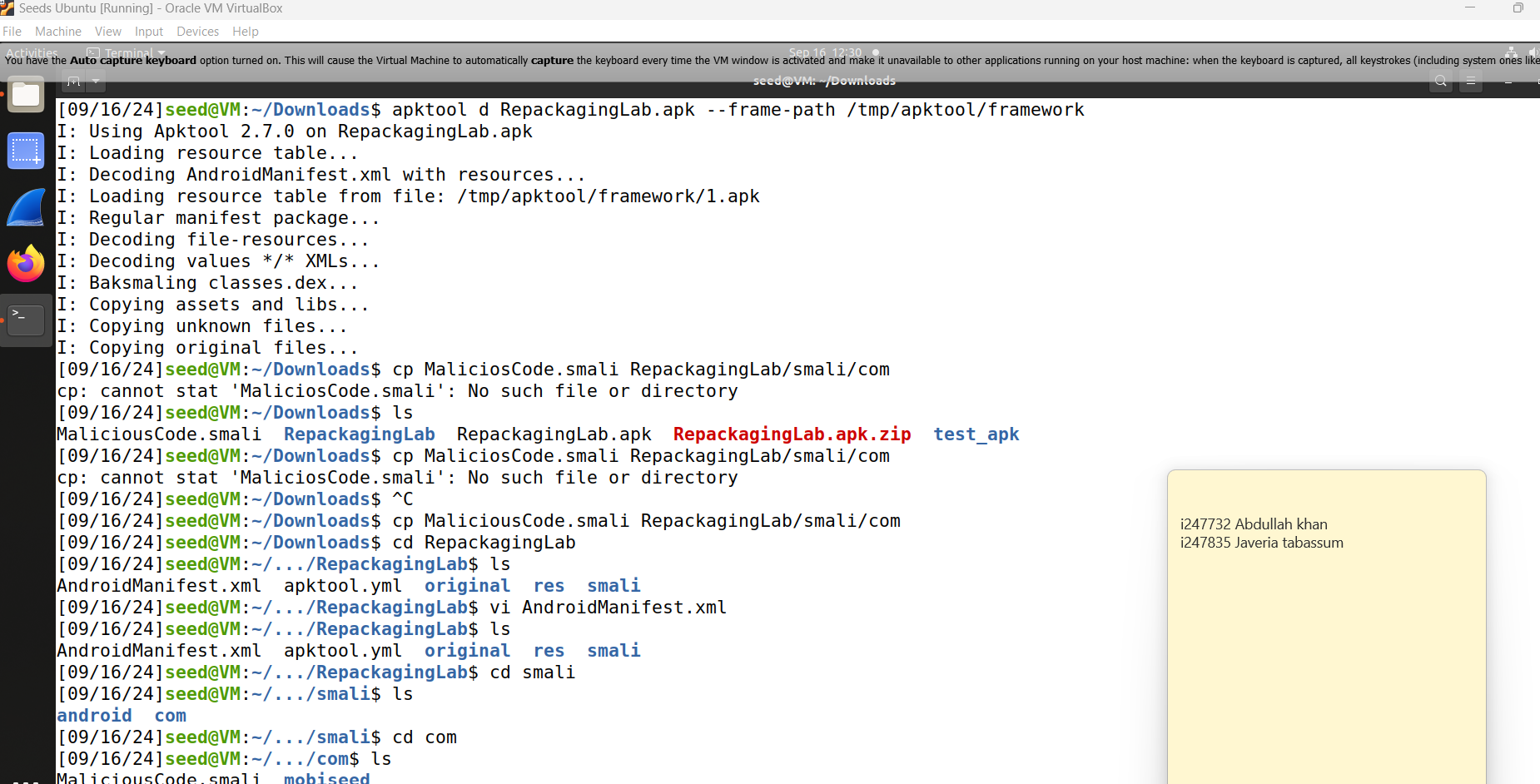


**Task#03: “Injecting Malicious Code into smali”:**

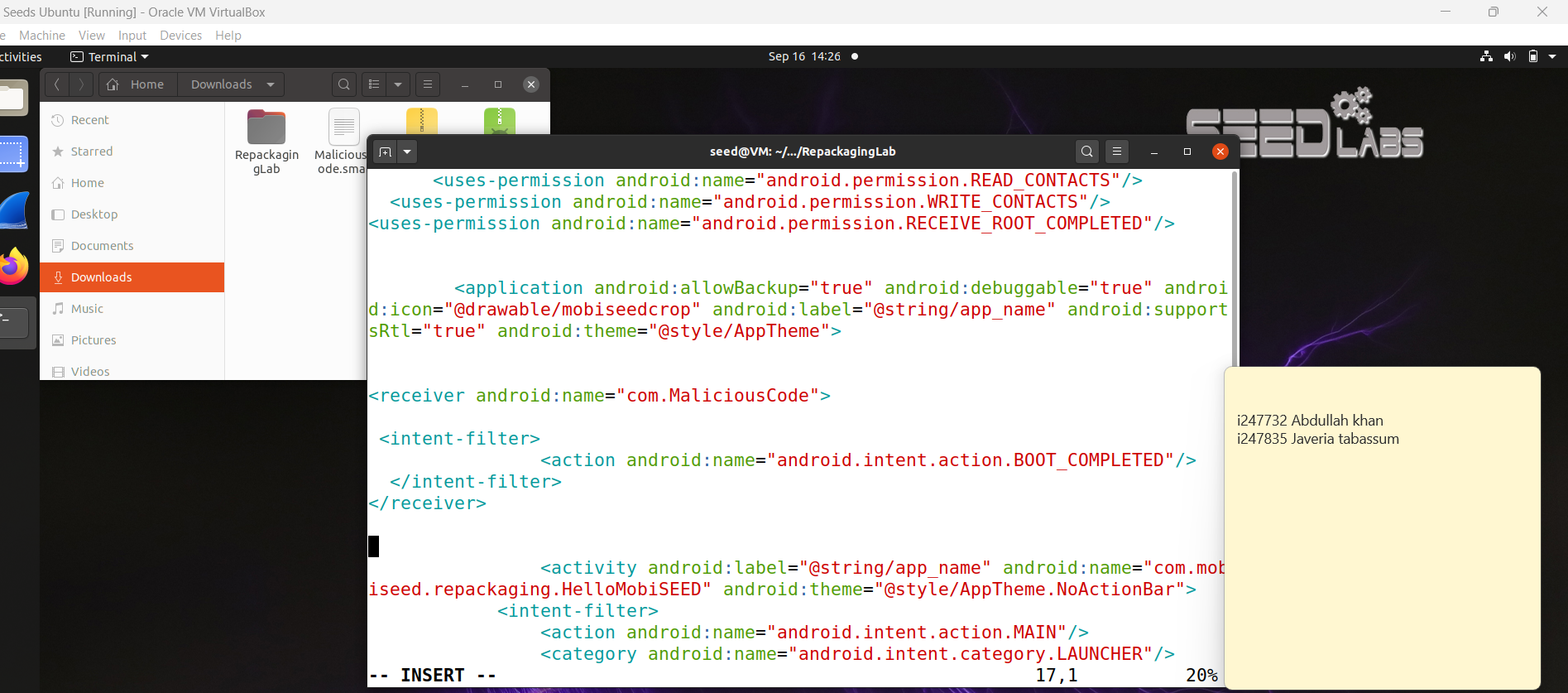
In this step we will be inserting malicious code into the extracted smali folder.

**Inject Code:** Inject the provided code ie MaliciousCode.smali at a specific location in the app. Write the following command on the terminal: cp MaliciousCode.smali RepacakagingLab/smali/com

Screenshots of where we injected the code and what changes were made to the code are:



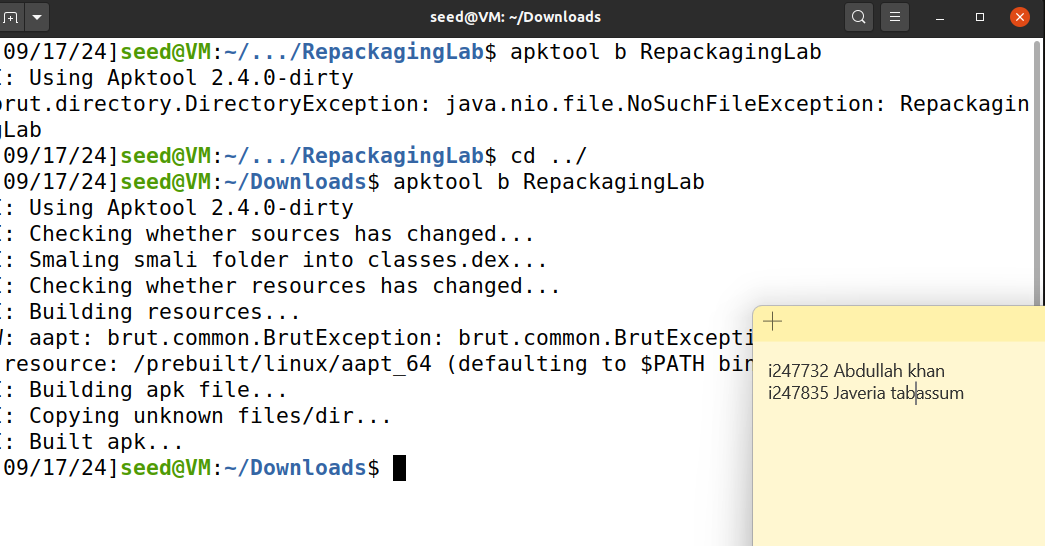
We set the permissions and set the action trigger when boot is completed in our androidmanifest.xml file



**Task#04: “Repackaging the Android App”:**

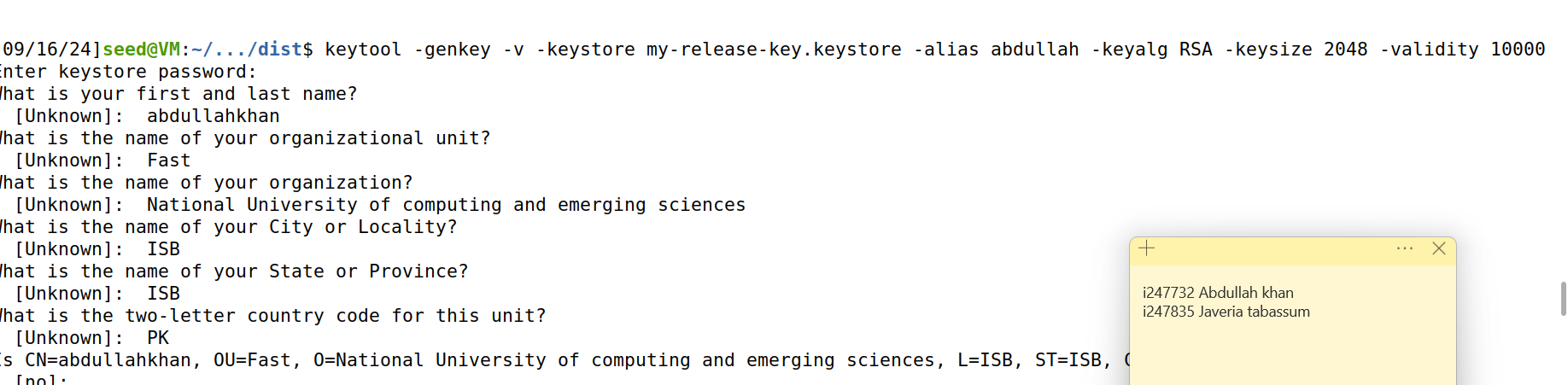
In this step, we will rebuild the apk which will have a malicious file injected in it. Now that the code is injected we will rebuild apk using apk tool. The decompiled app structure has a smali folder along with AndroidManifest.xml file. Use the following command on terminal to rebuild apk:

apktool b RepackagingLab



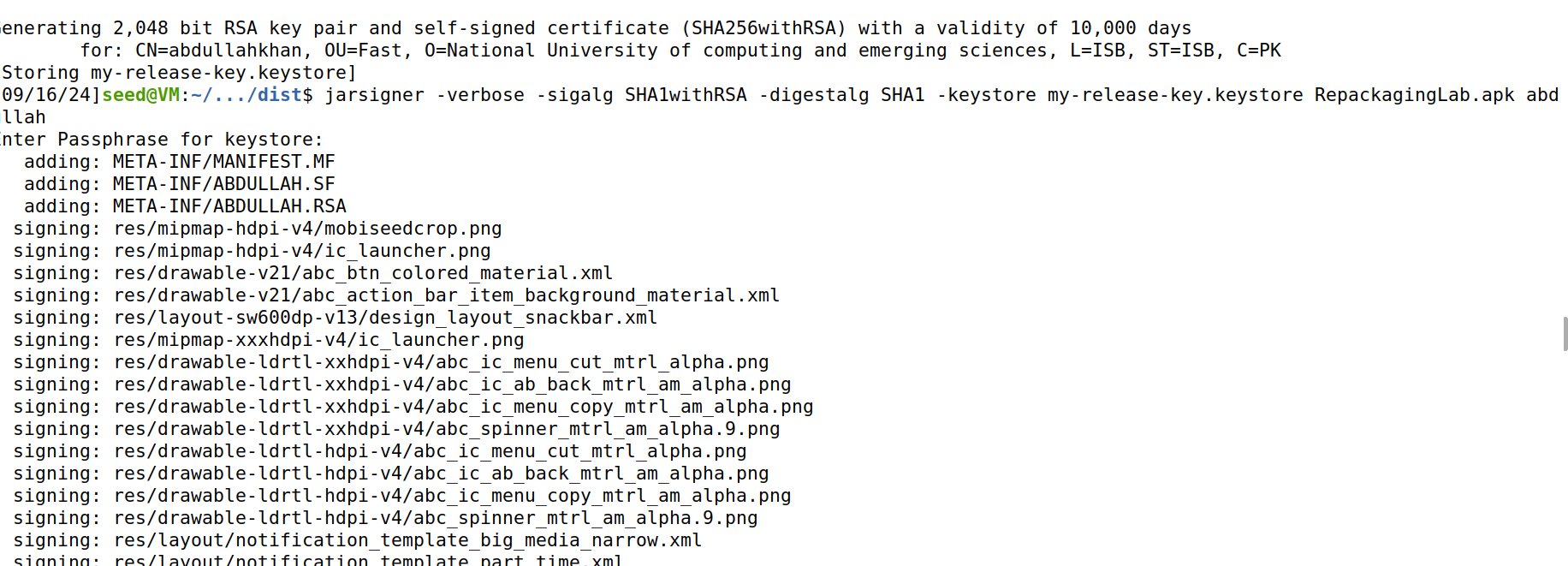
The next thing to do is signing the apk tool because we cannot install the repackaged APK on any android device which is not signed. To sign the apk we can create or use an existing key store, in our case we need to create a new one using keytool command in the terminal :

(keytool -genkey -v -keystore my-release-key.keystore -alias alias\_name -keyalg RSA -keysize 2048 -validity 10000)



Now that the key is generated, we will sign it using jarsigner command:

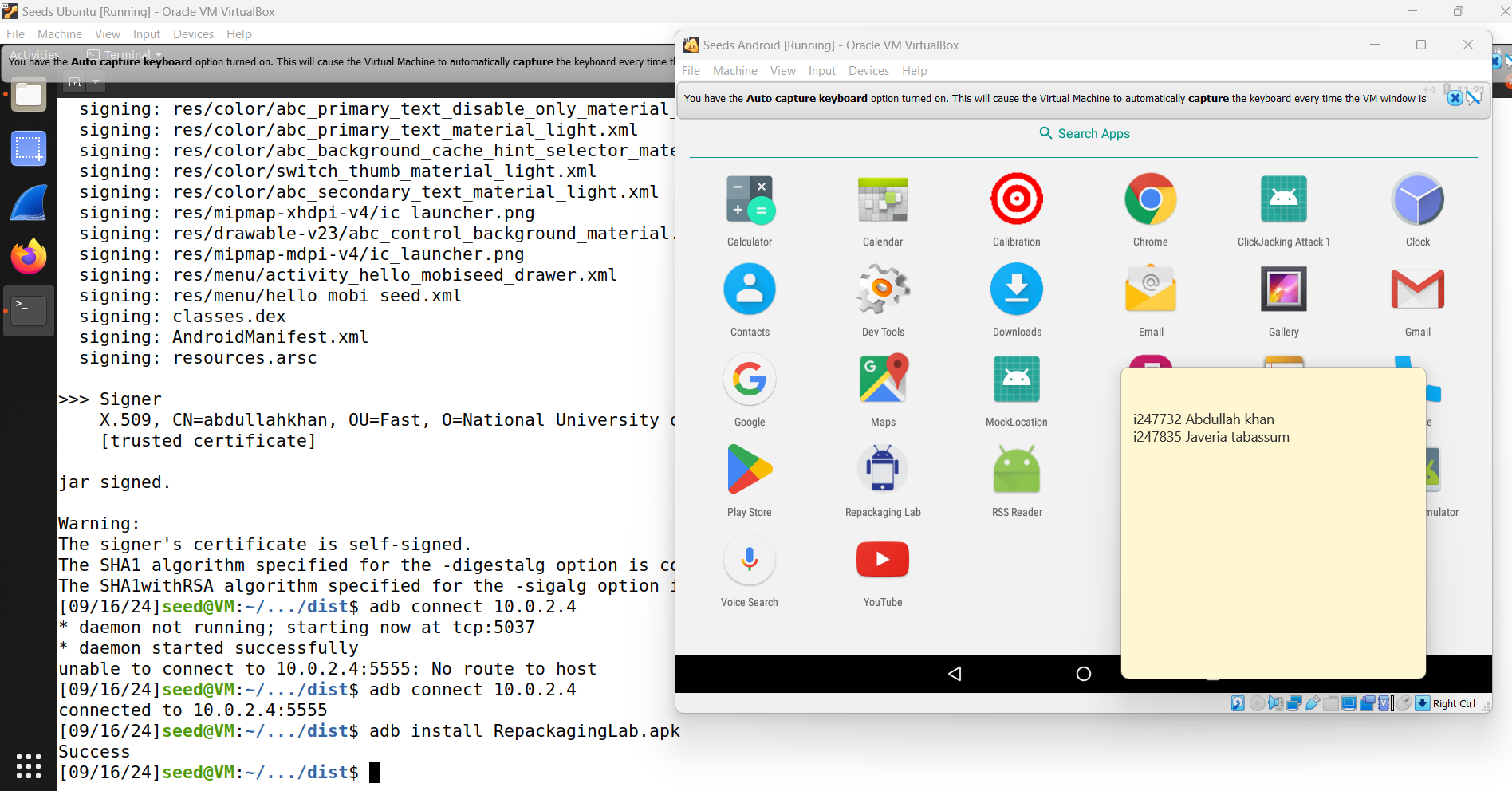
(jarsigner -verbose -keystore my-release-key.keystore repackaged\_app.apk alias\_name)



**Task#05: “Installing the Android App & Triggering the Attack”:**

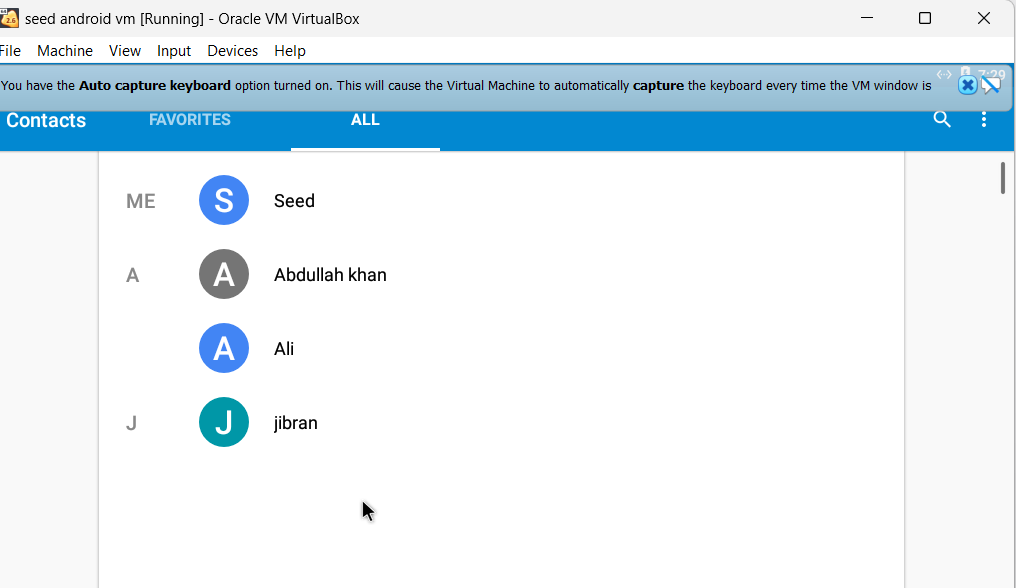
In this step, we will install the application and trigger an attack which involves deleting contacts from the device because of the malicious code injected. Before that we need to first add some dummy contacts on the android VM which are intended to be deleted later on. Use the adb command on terminal to install the app:

adb install RepackagingLab.apk



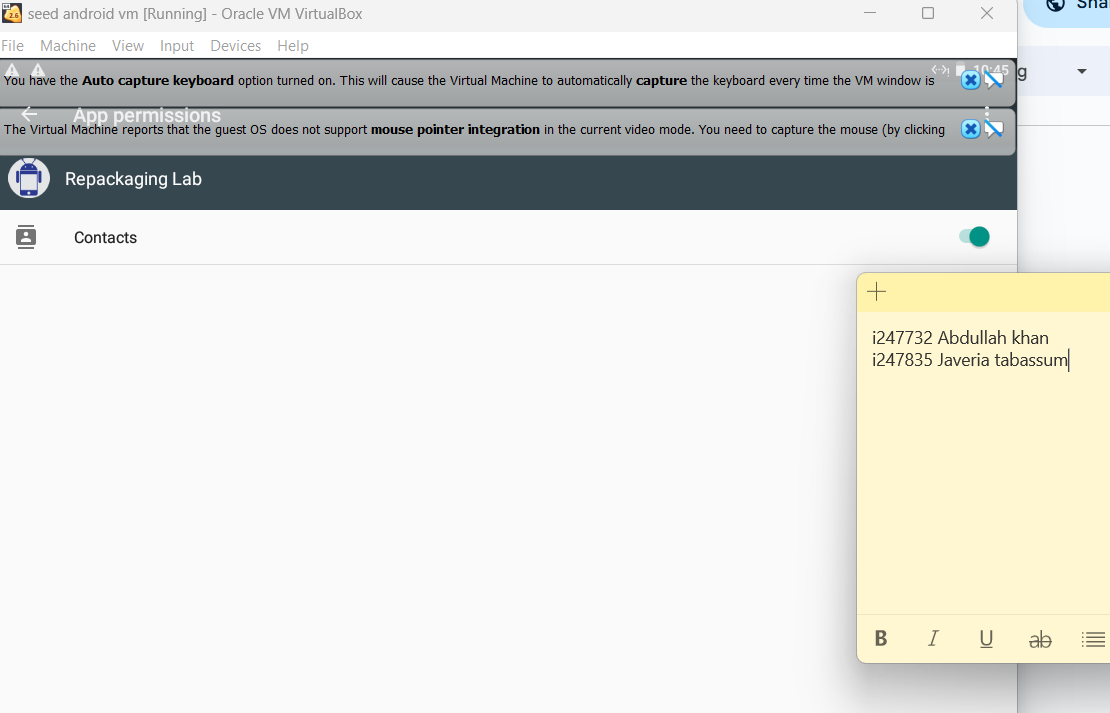
Now to add dummy contacts on the android VM’s contact app:

// Screenshot of the Added contacts.

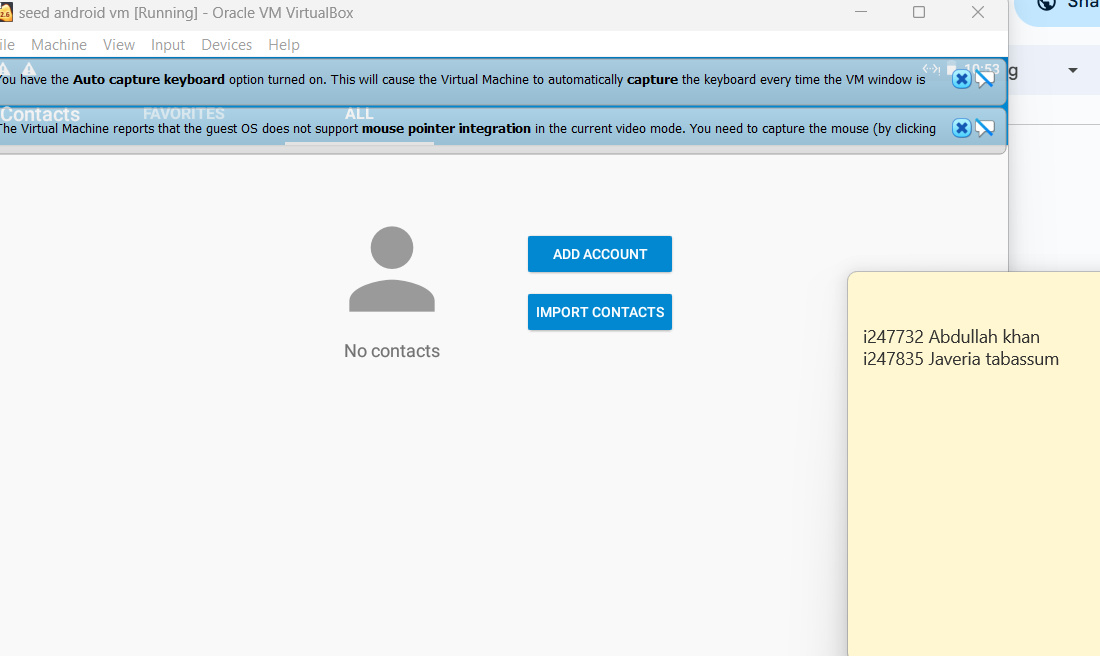


We also need to ensure that the permissions are set, in order to delete the contacts the malicious code needs the permissions to be granted from the settings.

// Screenshot of the permissions granted.

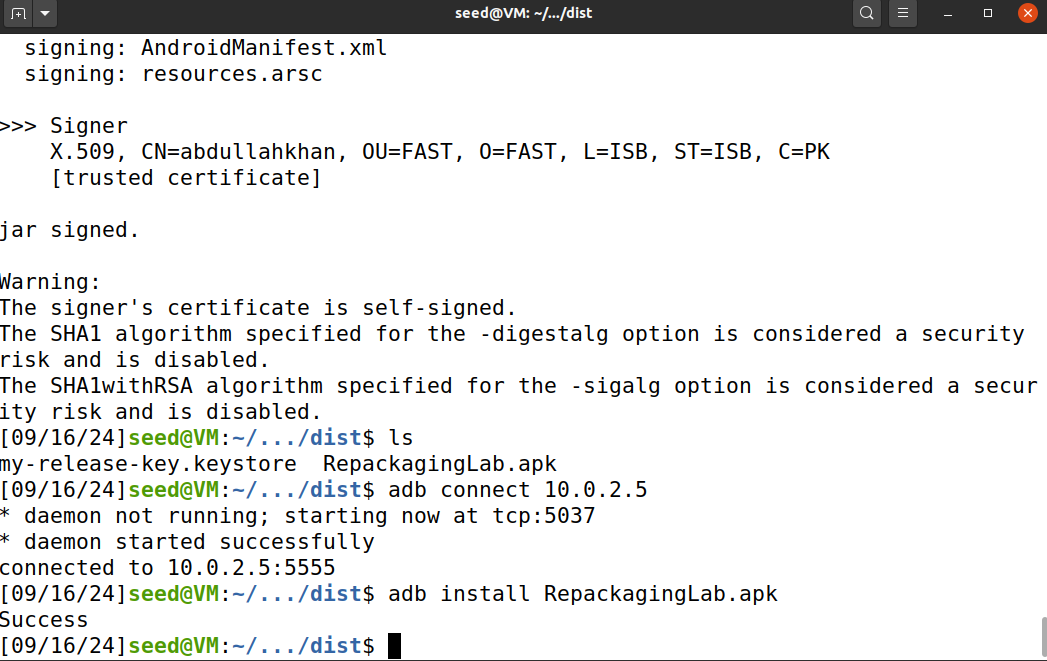


Now as per malicious code injection our contacts should have been deleted on rebooting the Android VM and installing the app. On running the app we can see that the contacts have been deleted by the malicious code.



we were facing some problems in the previous machine so we had to create another virtual machine on android having the ip address :10.0.2.5

—



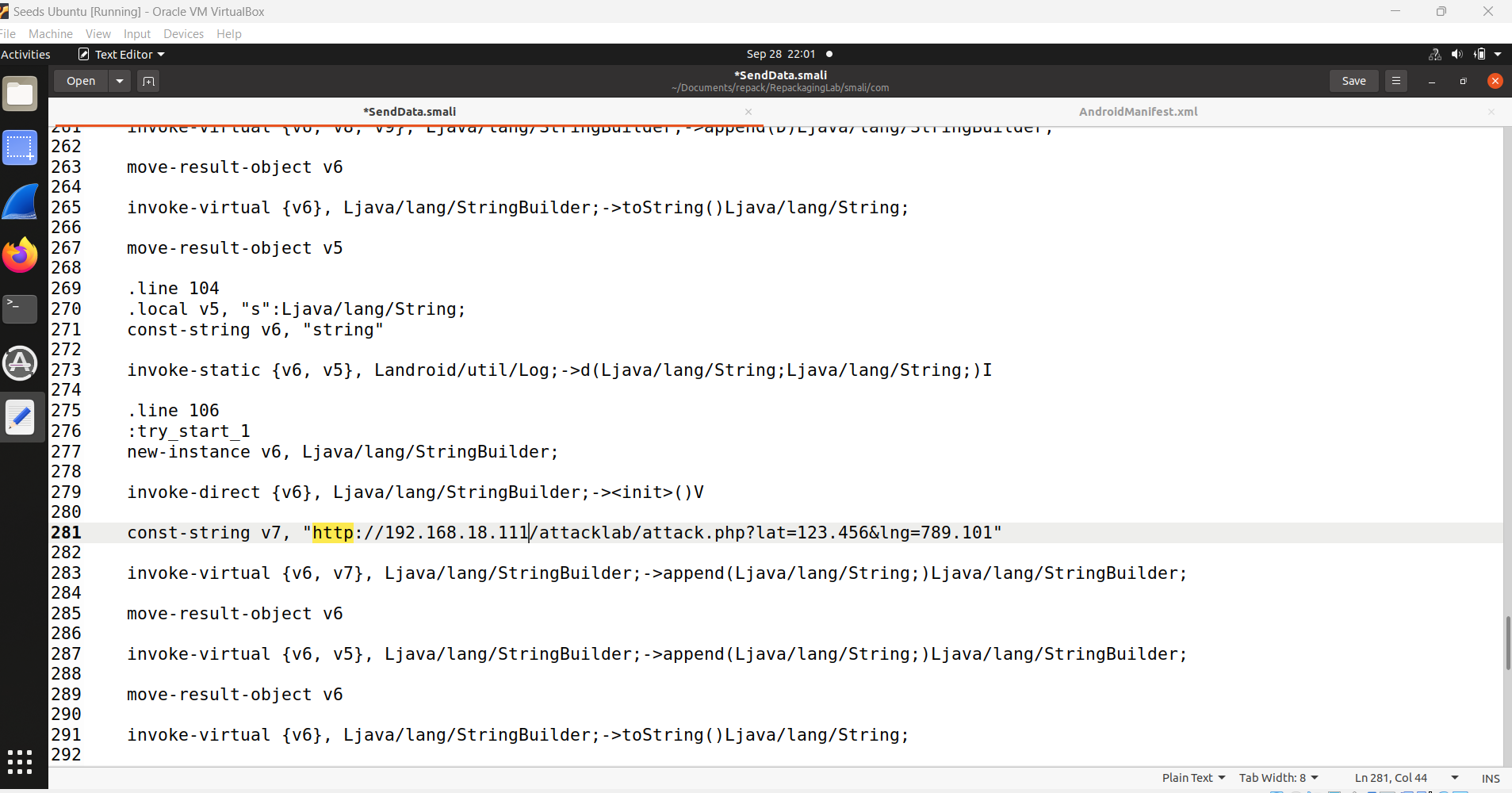
**Task#06: “Using a Repackaged App to Track Victim’s Location”:**

In this task, we will be using a repackaged app to track the victim’s location, to do that we will first repackage the android app, inject code for location tracking functionality and then monitor the victim’s location. This involves following steps:

**6.1. Setting Up Locations**

For this purpose we need to disassemble the app to inject the malicious code so we will disassemble the app like we did earlier (ref task 2), then add the logic that enables location tracking and communication with the DNS server (we will configure in **Sub-task 6.2**). The code in these files sends the victim’s location (latitude and longitude) to the server.

// sendData.smali file sending data to our DNS server



Description of three malicious files injected:

// sendData.smali file :

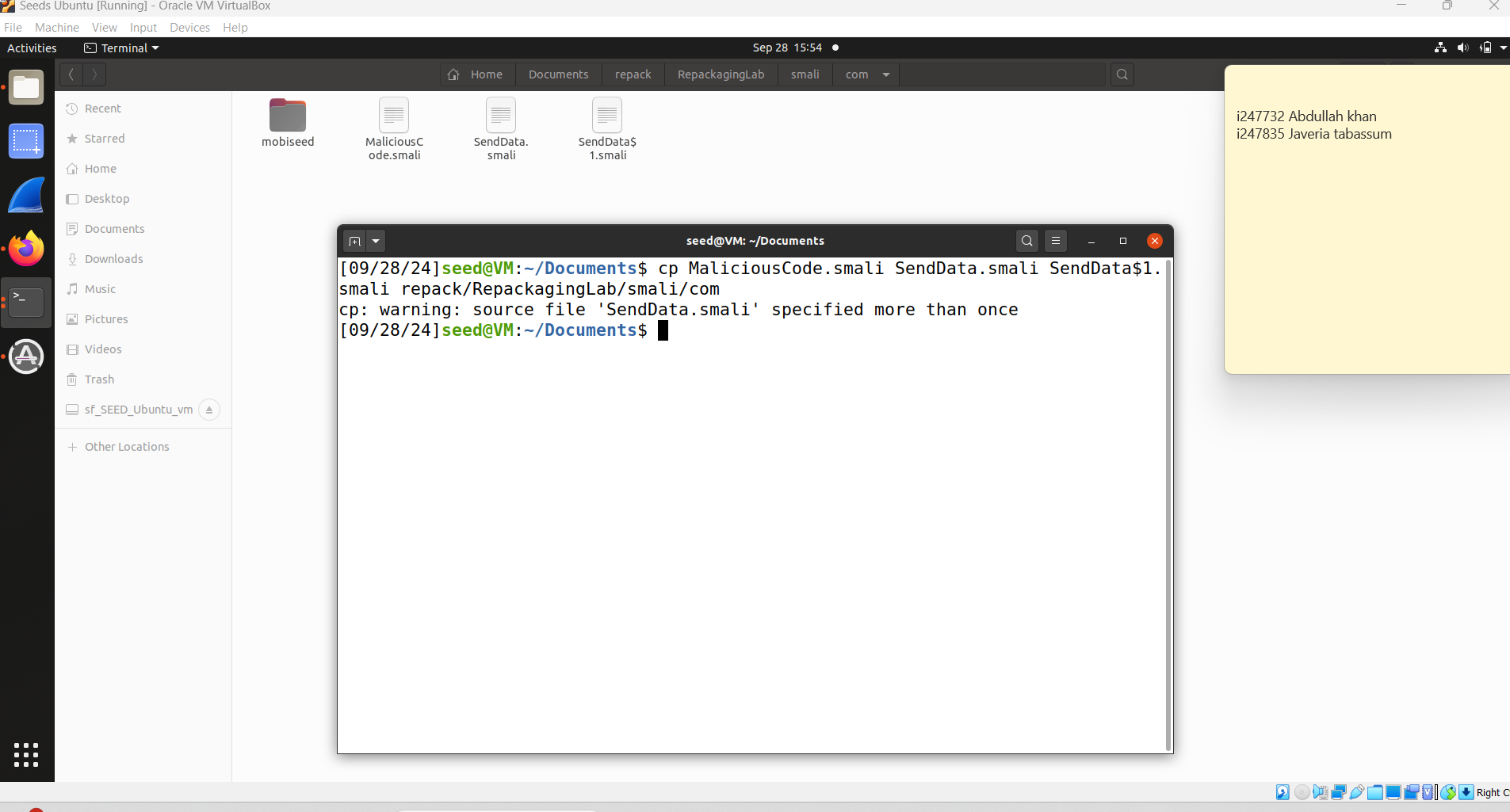
This file collects the device's location, by sending sensitive data of a victim to a remote server which in this case is controlled by our system.

// sendData$1.smali file:

This file sends data file sends the location data of the victim to sendData.smali. We can use this file to launch multiple attacks to the victim’s device.

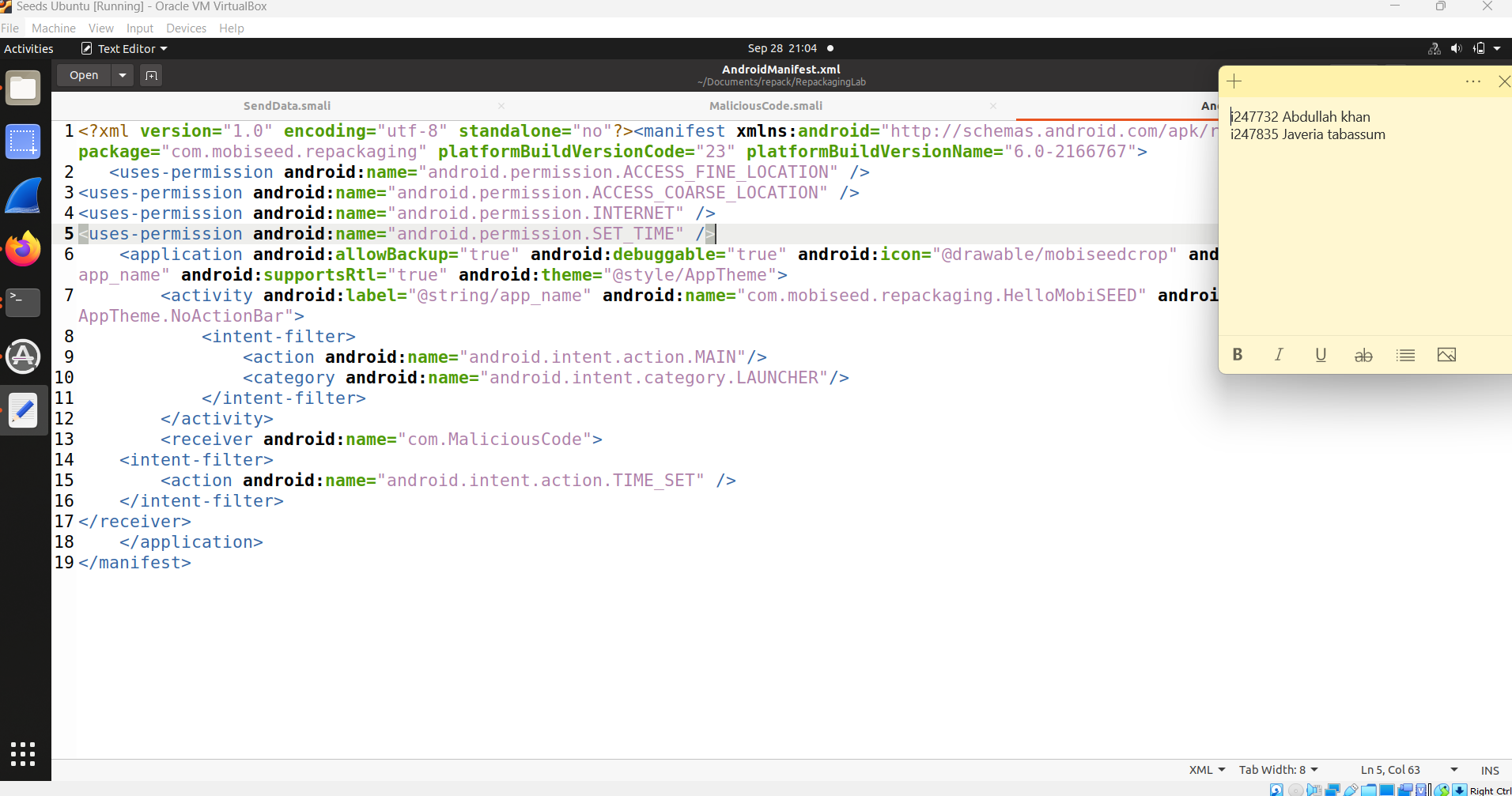
//MaliciousCode.smali file :

This file is responsible for gathering location information and transmitting it to the server and without the consent of the user, it initiates location tracking.

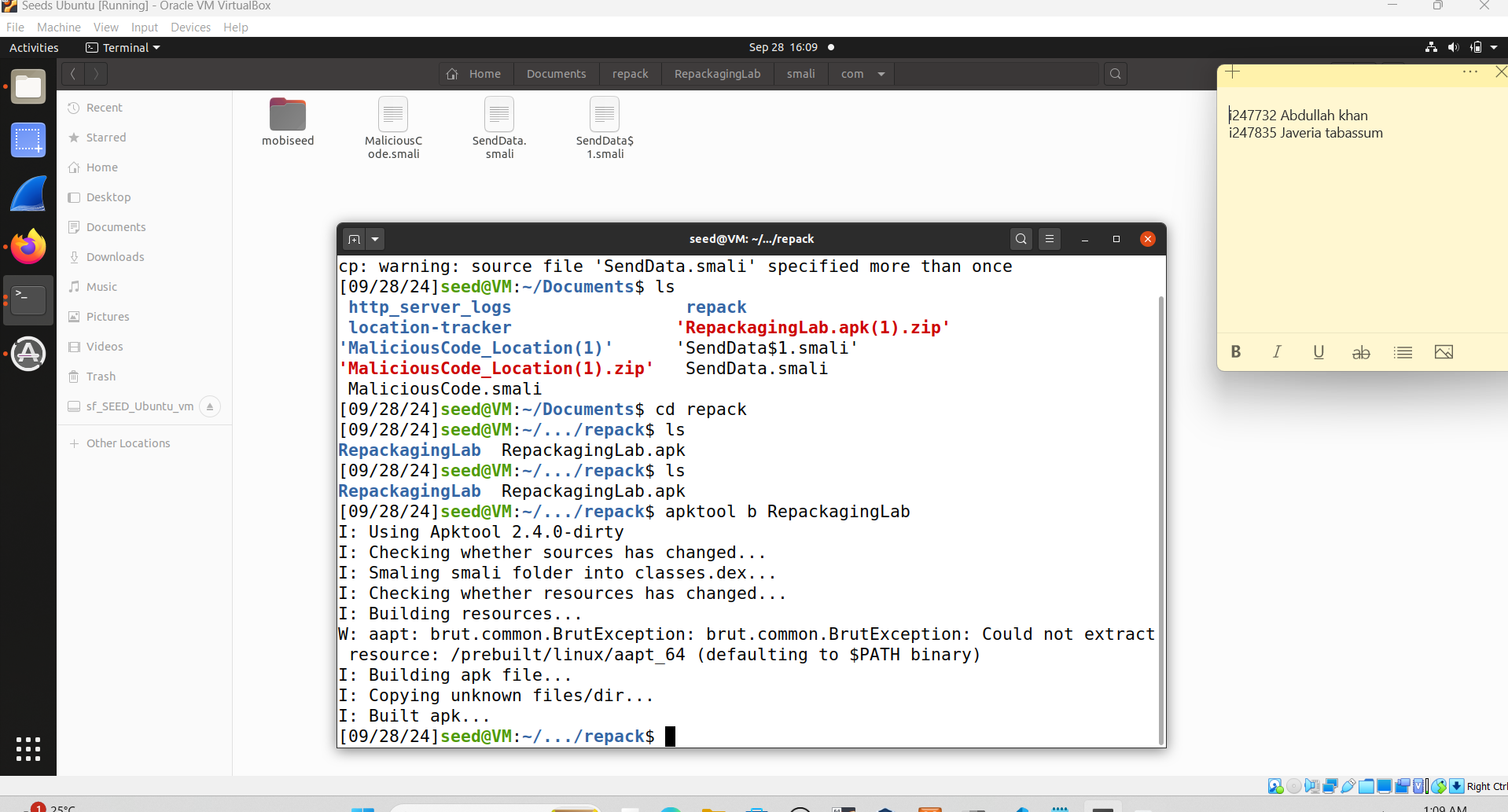


Next we will set permissions to trigger malicious code on a specified time. We are basically setting up permissions and configuring the application’s broadcast receiver which allows the app to execute malicious code injected from (com.MaliciousCode)

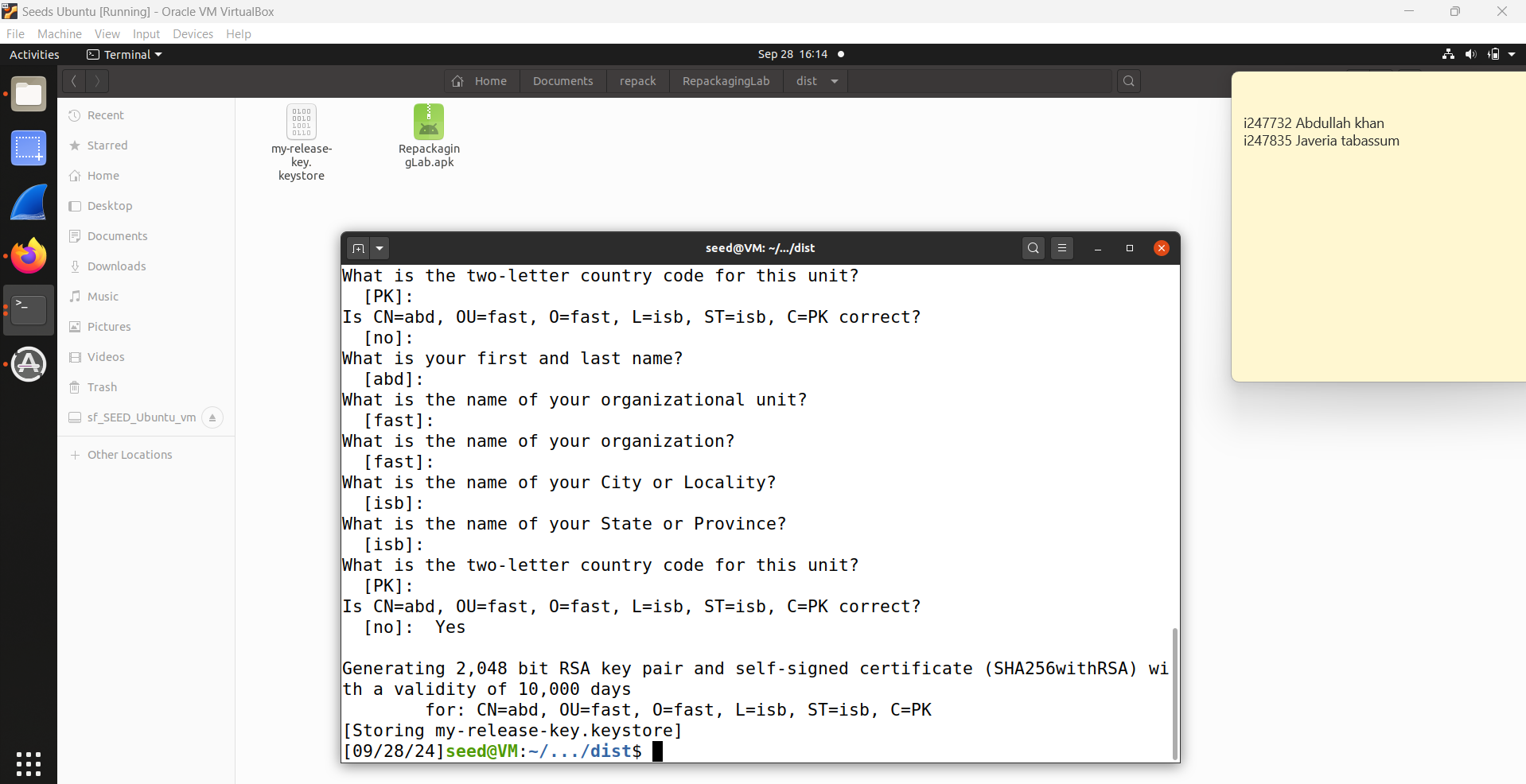
whenever the device’s time is changed. The permissions related to location and internet allow the app to track and transmit location data.



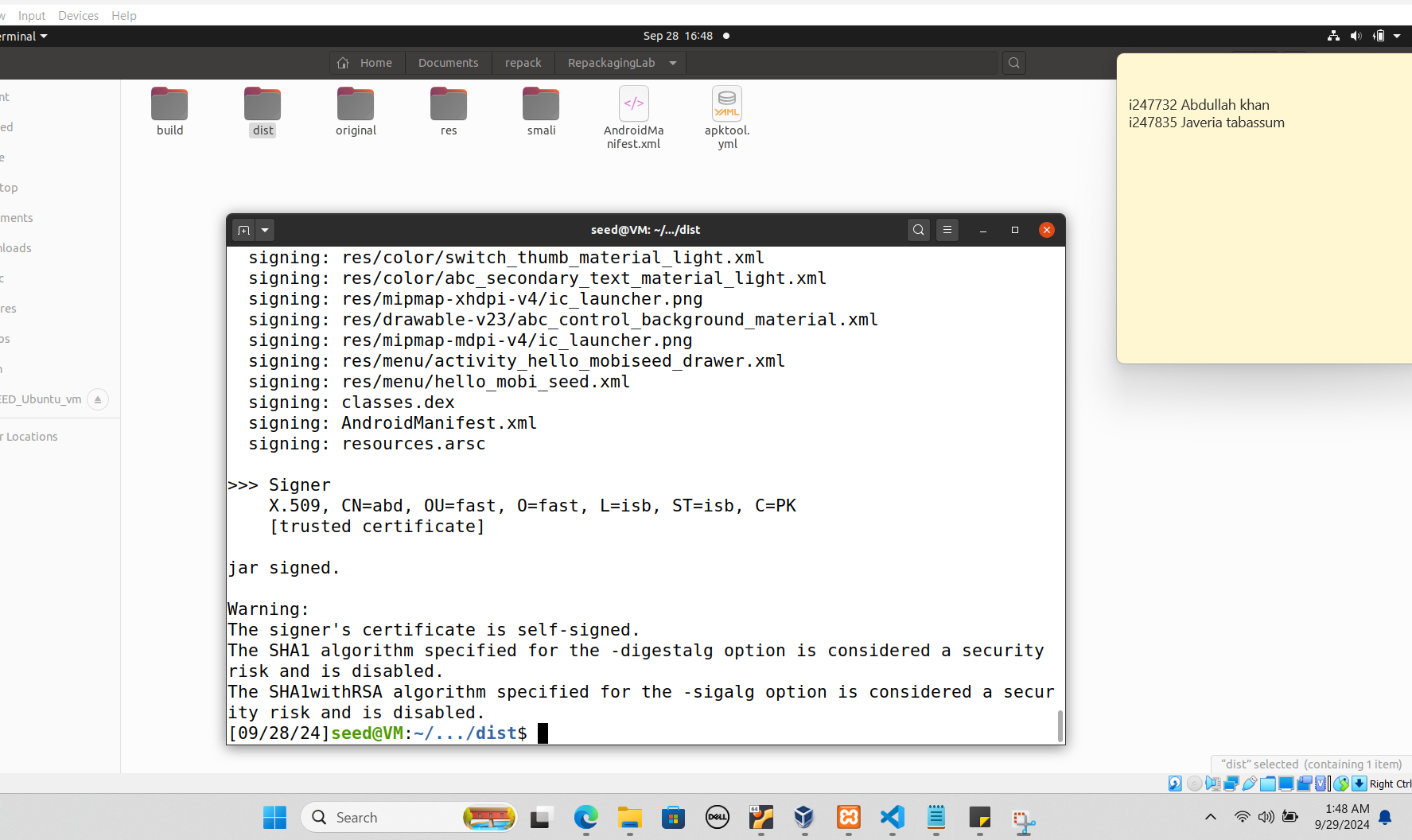
Then we will rebuild the app using the apk tool by typing the same command we used in task 04.



Finally, we will configure location services to simulate the target location by generating key

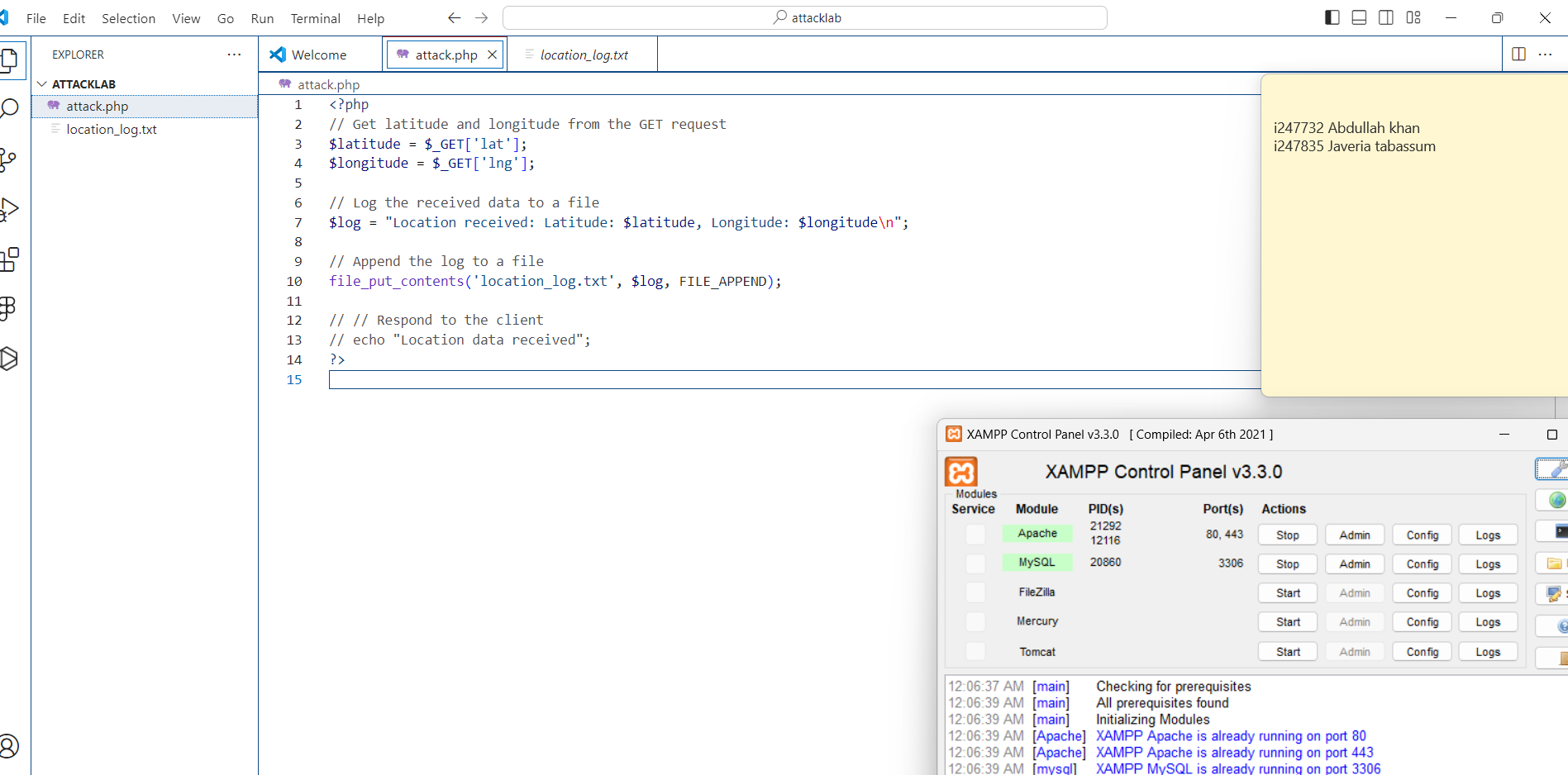


then we will sign the file and after the file is signed we will connect it to android using adb install command in the step3



**6.2. Configuring DNS**

In this step we will configure a DNS server to send and receive data from the victim's device. For that we created an apache server on xampp and we created a get api that will get longitude and latitude and will append it in file.

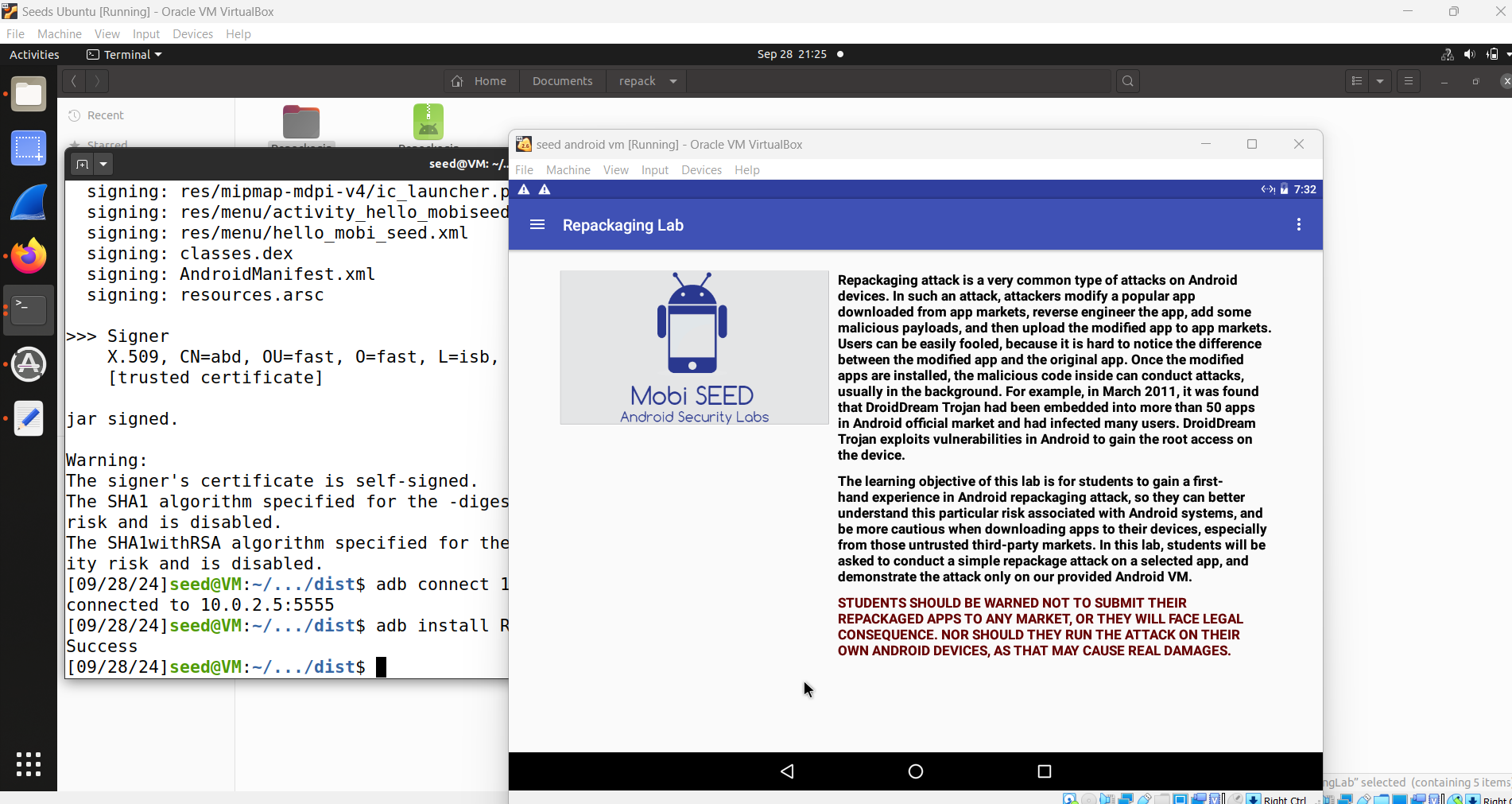


**6.3. Repackaging and Installing the Victim App**

In this step we are finally repackaging and installing the android app in which we injected the malicious code for tracking location. (these are the same steps being revised, we already performed them in above tasks)

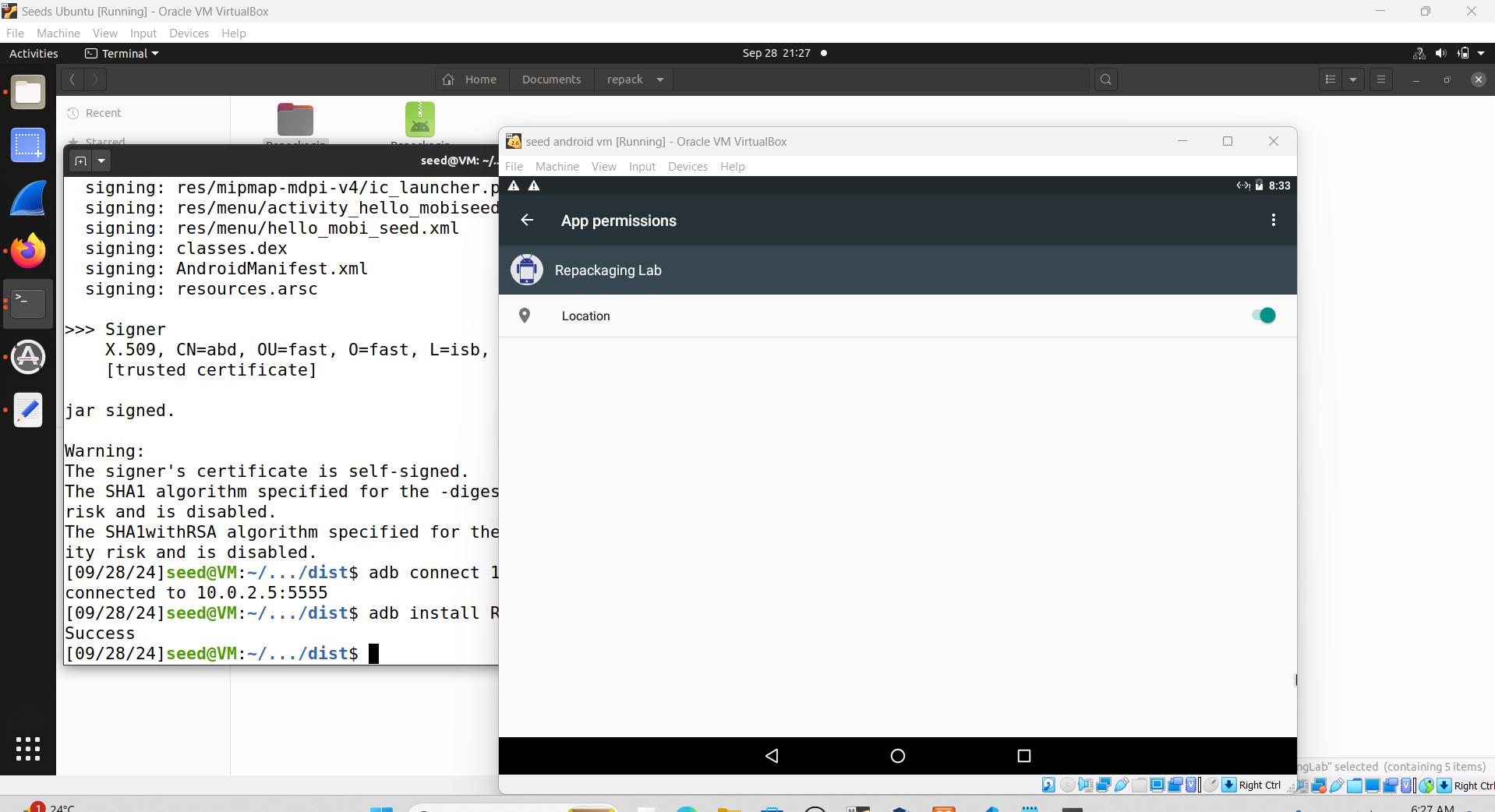


As it's clear in the below screenshot that the android app is installed in the victim’s device.



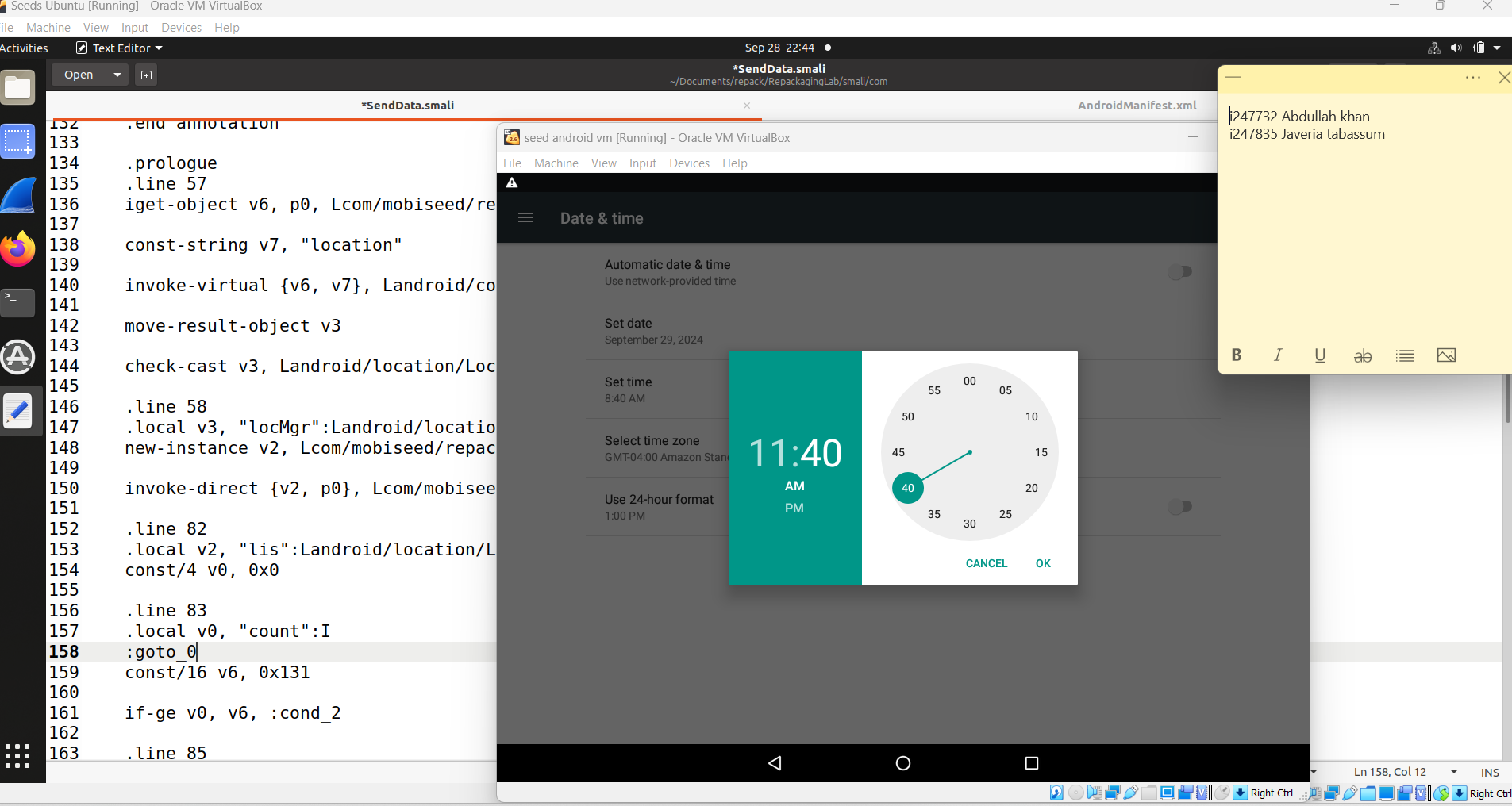
**6.4. Enabling VM Permissions**

In this step we will enable VM permissions for location access on the victim's machine.



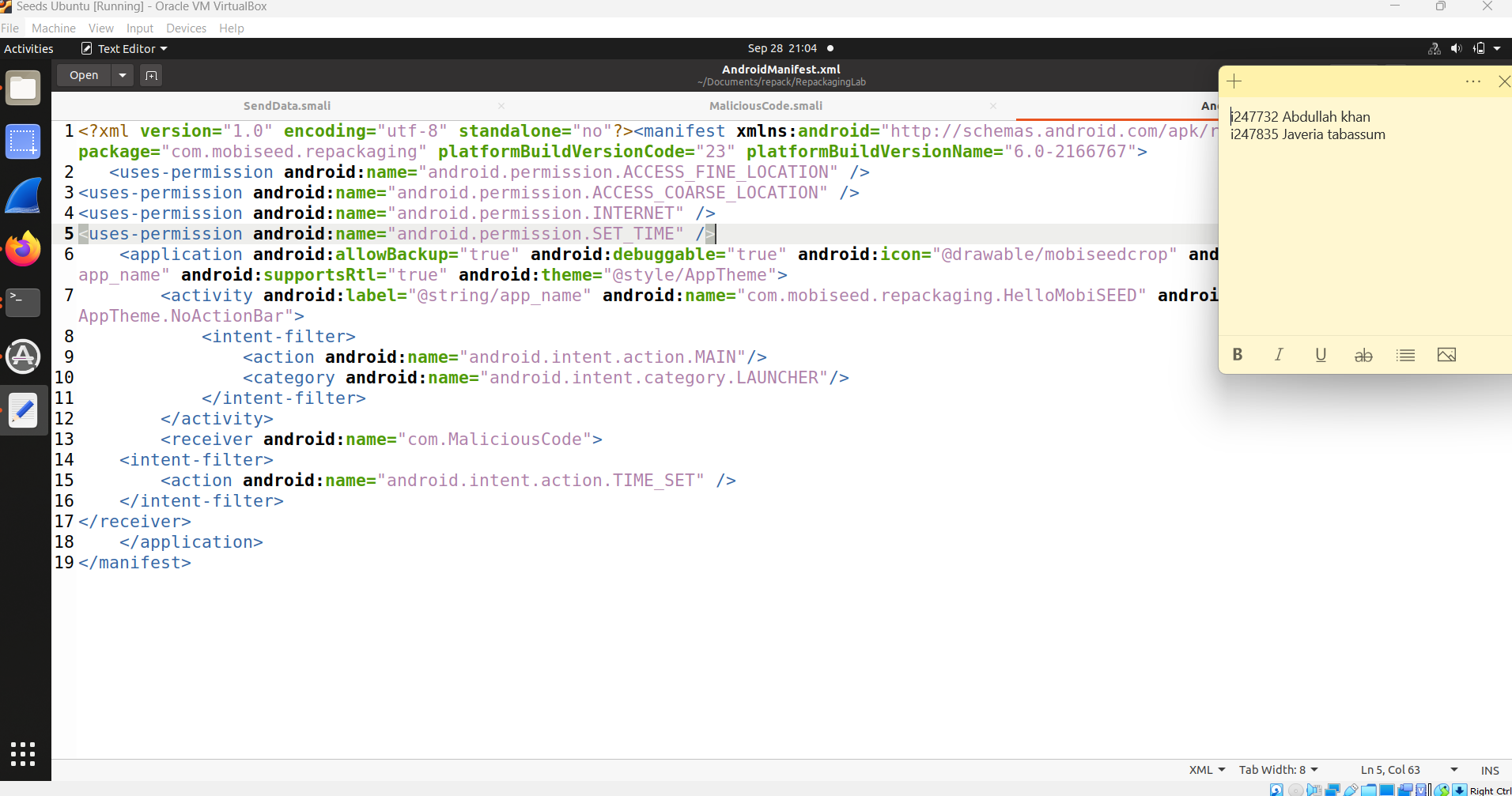
**6.5. Triggering Attack Code**

In this step, we will trigger the malicious code that tracks the victim’s location. For that we will set the time to trigger the attack.



So the malicious code is triggered when the system sends the broadcast android.intent.action.TIME\_SET, which occurs whenever the device’s time is changed.

the screenshot for reference of our query for attack:

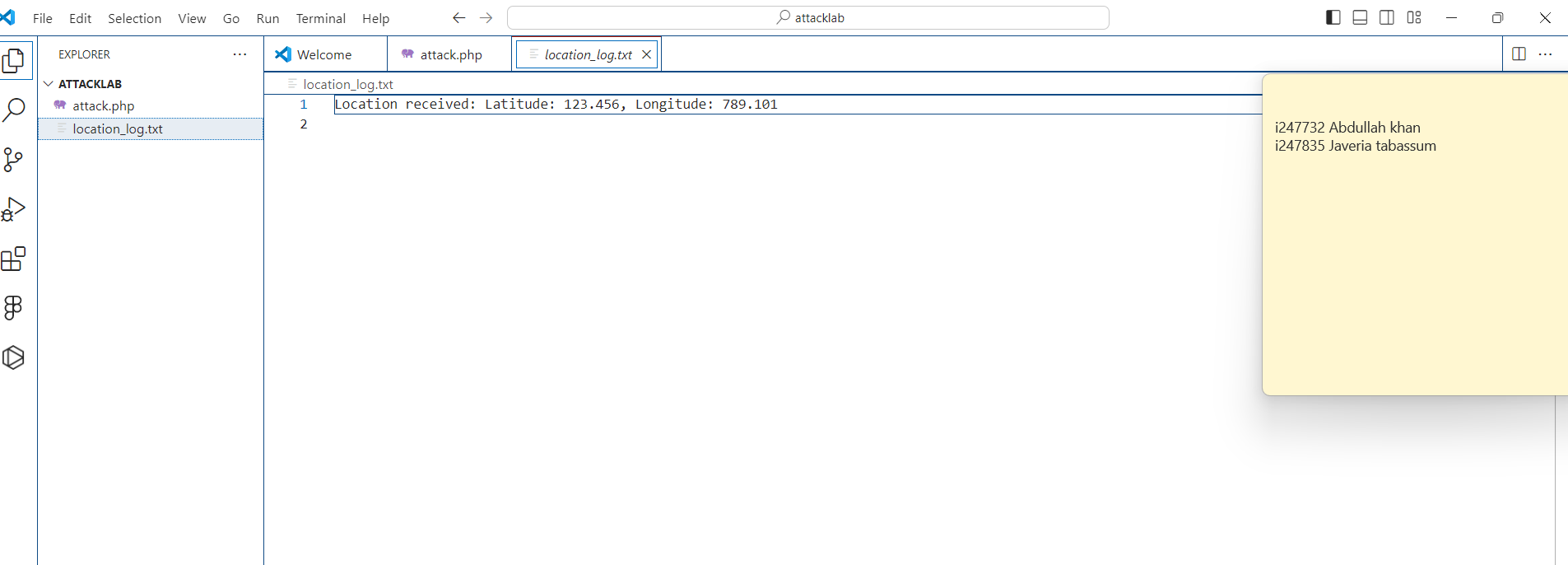


In the above queries, the app requests several critical permissions:

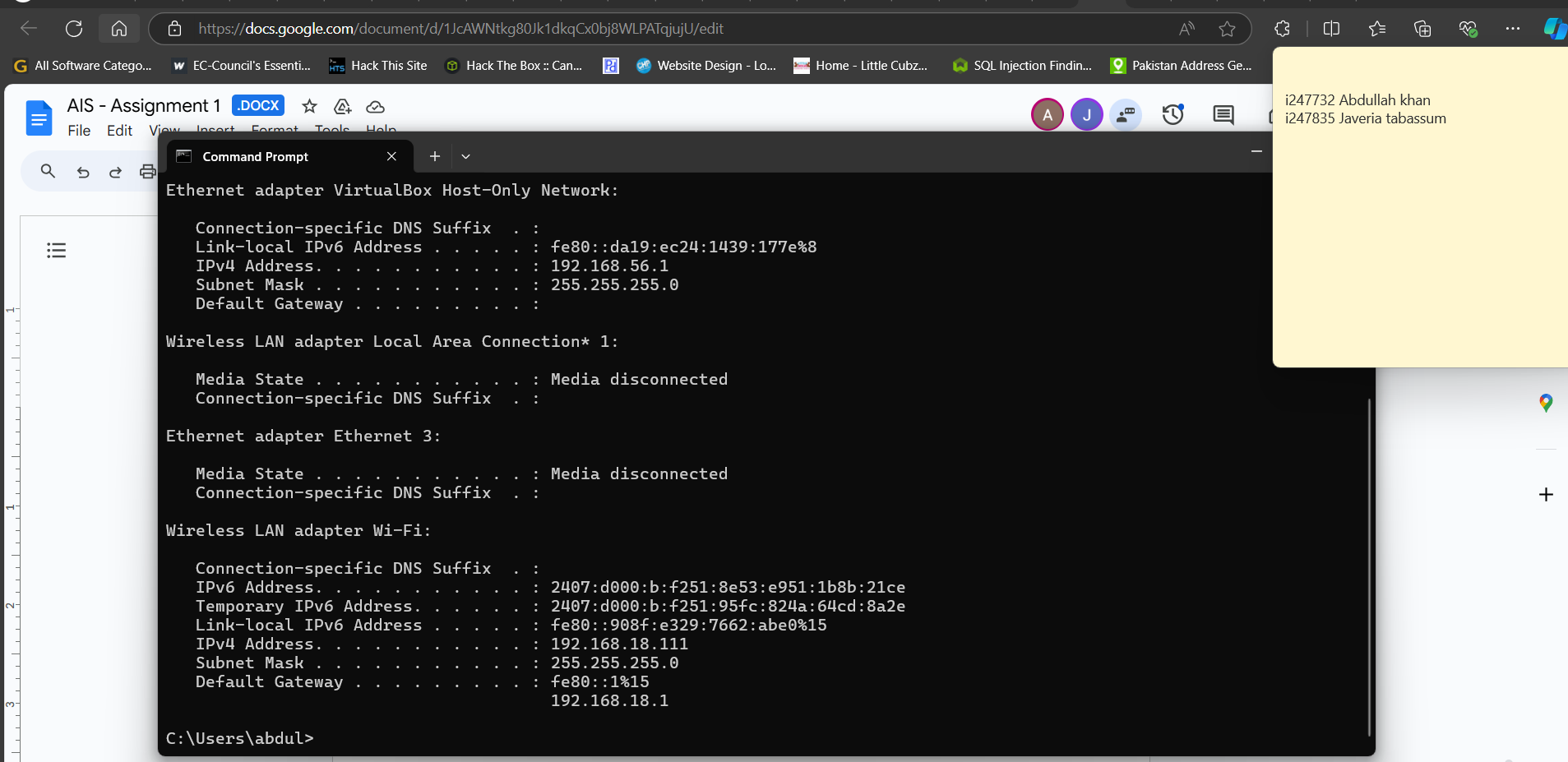
* **ACCESS\_FINE\_LOCATION**: Grants access to precise location information from GPS.
* **ACCESS\_COARSE\_LOCATION**: Grants access to approximate location information, using Wi-Fi or mobile networks.
* **INTERNET**: Allows the app to access the internet.
* **SET\_TIME**: Allows the app to set the system time.

**6.6. Tracking the victim**

The below screenshot is from the server which clearly shows the location of the victim being received based on the injected code for tracking.



The server (Xampp) was running on localhost, so we used our machine’s ip to access the server.



also we used dummy content because gps isn't available on our android VM.