horizontal_lockup_1470px copy 8.png

Proficiency Test for TVC Team Protego [Response Template]

Name of Associate: Ankit Yadav

Project Role of Associate:

Proficiency Test Score:

Proficiency Test Status:

**1.** Describe the contents and structure of an Android Package (APK).

Android package contains all the necessary elements that are required for installation and running of an application.

The contents and structure of an Android Package(APK) are described as follows :-

* **AndroidManifest.xml –**

It is the most important file of an apk .It is like heart of an application which tells necessary information about an application.

It contains the information of package name,version,permission, and component used in an application like activity, services, broadcastReceiver and contentProvider.

From the reverse engineering perspective, this file helps to guess and decide the potential threat that can caused by going through permission asked and components used.

* **lib folder –**

This is the folder that contains native libraries like c/c++ code in the sub-folder like armabi, armeabi-v7a, arm64-v8a, x86 and x86\_64.

In these folder native file is named as lib followed by name of file with .so extension.

If there is any file with .so extension then it is suspicious for security reasons as it can work with android hardware level and change the architecture or security mechanisms and also it can load any api key or encrypted text in the apk.

* **Classes.dex**

It contains all compiled code of java/Kotlin.

It helps to know the code functionality and behaviour.

For a reverse engineer it helps to know app behavior and network communication it is initiating or any file modification.

* **Assets**

It contains raw file like sound,font which is directly used in application.

It also contains data of some framework like cordova or reactnative.

From reverse engineering point of view, this file is important to check if it is using any framework or using webview for showing malicious webpage and also this file can contains encrypted text or configured file or obfuscated code which can be directly used in application for malicious purposes.

* **res**

This folder contains resources like layout, drawables, mipmap,strings,color which is used in application.

This folder can be used for malicious purpose like creating fake login page for phishing activity,creating icon for impersonating any other application.

* **META-INF**

This folder contains verification information and is generated when signing the application.

This is used by reverse engineer to verify that any apk is tampered or not by checking the digital signature of suspicious apk with any trusted source or real apk.

This also contains **MANIFEST.MF** which verify the validity of each file in the zip or apk ensuring that altering any file will revoke digital signature and make the apk invalid .

* **Resources.arsc**

This file contains precompiled table of resources or xml file

It contains information that links the code (classes.dex) to the resources(res).

**2**. Describe the following malware categories and discuss at least three (3) signals that you could use to detect each of them: Spyware, Ransomware, Backdoor, and Rooting application.

**Spyware –**

A spyware is a malicious code or software which collects the user information (PII data) like contacts, sms,call log, photo, file etc and send to third party or server without user consent.

**Signals to detect spyware –**

* **User permission –**

If the application is asking for dangerous permission like READ\_CONTACT, READ\_SMS, RECEIVE\_SMS or CAMERA etc then it is a signal for spyware.

* **Covert communication –**

If the application is making a communication to any server then this a flag as spyware connects to remote server to exfiltrate the stolen data and hence it is a for spyware.

* **Background service or alarm manager –**

Spyware may use background service like job scheduler or work manager and also can use alarm manager to periodically sync data and send to remote server.

* **Fast battery drain and network activity –**

Since spyware can periodically send data to remote server there may be chance of battery drain and high network activity.

**Ransomware –**

Ransomware is a type of malware that encrypts the user data or file and blocked the access unless the ransom is paid.

**Signals to detect ransomware –**

* **External storage permission –**

Ransomware can access file storage and encrypts the data and made it inaccessible.

* **Encrypted text or string –**

In the code there may be encrypted text or code which is used by ransomware to encrypt the file and decrypt the file when ransom is paid.

* **Communication with remote server –**

Some ransomware establish connections with remote controlled command server and send command to encrypt the file and also to decrypt the file after ransom is paid.

* **File extension –**

Some Ransomware contains encryption code in a file which file extension is different from the other file so on this basis we can check for ransomware.

**Backdoor –**

Backdoor is such type of malware that lets attackers bypass security measures and allows access to mobile devices remotely.

**Signals to detect backdoor –**

* **Unusual permission –**

If the application is asking for permission like INTERNET,READ\_CONTACT, READ\_SMS etc then there is a chance of backdoor.

* **Dynamic code loading –**

If the application is dynamically loading the code by calling method such as **class.forName( ) or dex.classLoader( )** then there may be chance of loading the code to bypass the analysis.

* **Unusual URL or ip address –**

In application there may be hardcoded url string or ip address which may be suspicious.

* **Background service –**

Application can use background service to make connection with remote server to control or gain access to user data remotely.

**Rooting**

A malware which gains root access (super user access ) of a device and bypass the security the mechanisms.

**Signals to detect rooting –**

* **Native libraries –**

Malware can use c/c++ code which works on the android OS level to bypass the security mechanisms.

* **SU binary file –**

Application may contain su (super user) file which grants access to root level.

* **Abnormal behaviour during execution –**

While doing dynamic analysis app may behave abnormal since the security of android OS is tampered.

* **Alter file extension –**

Rooting application changes the file extension of some file or inserted file which has different extension than other file.

**3.** Describe two (2) permissions in Android.

Two permissions in android are describes as follows –

* **READ\_CONTACT –**

This permission is asked to read contact list from device.

Permission is defined in androidmanifest.xml and also asked during runtime.

Code snippet –

<uses-permission android:name="android.permission.READ\_CONTACTS" />

Entry point for read\_contact while doing reverse engineering is **ContactContract.**

* **READ\_SMS –**

This permission is asked to read list of sms from device.

Permission is defined in androidmanifest.xml and also asked during runtime.

Code snippet –

<uses-permission android:name="android.permission.READ\_SMS" />

Entry point for read\_sms while doing reverse engineering is **content://sms.**

**4.** Describe two (2) security features in Android.

Two security features in android are described as follows:-

* **Runtime Permissions –**

The feature of runtime permission from android 6.0 has provided more security to user data as it is asked during the app run time execution rather than only during installation which is used to be before android 6.0.

* **Google Play Protect –**

Android provide the google play protect service which scan the app while installing and also it scan periodically to ensure that the harmful or PHA categories application is not active in the mobile.

It uses machine learning algorithms to scan the application.

**5.** When an application is suspected of cloaking, is it best to rely on static or

dynamic analysis and why? What are the pros and cons of static vs dynamic

analysis in malware detection?

Cloaking is a technique of **hiding malicious code** in the application. So both static and dynamic analysis is helpful but both have their own pros and cons.

**Static analysis –**

**Pros –**

* **Better understanding of code structure**

Since in static analysis we scan the code and understand the code behaviour then through this analysis we can better understand the code structure and flow.

* **Hard coded string or obfuscated code**

In cloaking technique,malware developer can use hard coded string which can be identified while doing static analysis.There are some tools which can identify hard coded string, url or obfuscated code.

**Cons –**

* **Dynamic code loading**

if the application is loading code dynamically then it is difficult to identify it through static analysis.

* **Dead code injection**

If the unused code is inserted and obfuscated then it is difficult to understand and get the right path of execution.

**Dynamic analysis –**

**Pros –**

* **Runtime behaviour –**

In dynamic behaviour we execute the application on tool like emulator httprequest and track the log and check the behaviour during runtime.

So in this technique we can track the httprequest made by the application.

* **Dynamic code loading**

if the app is loading code dynamically then it can be tracked during dynamic analysis.

**Cons –**

* **Anti – analysis technique –**

Some malware application has conditional check and is coded like that if is running in virtual environment or in testing method then malicious code is not executed.In that scenario, dynamic analysis is not helpful.

**6.** Name two ways an application can hide an API call from static analysis.

The two ways an application can hide an API call from static analysis are as follows:-

* **Dynamic code loading**

Loading the malicious code or API call dynamically while application is running can be helpful to hide an API call from static analysis.

* **Anti-analysis techniques and obfuscation**

Application can use conditional check to ensure that if application is running on static analysis tool then malicious code is encrypted to other form and also code can be obfuscated and change code structure to bypass static analysis.

**7.** What code is executed when an Android application is launched when a user selects the application's icon? What are two other entry points for an application?

If the app contains application subclass then **attachBaseContext**( ) method will be executed.

Otherwise **onCreate**( ) method of launcher Activity will be executed.

Two other entry points for an application –

* Static block
* attachBaseContext

**8.** How would you ensure that your application achieves persistence, restarting every time the device is powered on?

Application achieves persistence restarting every time the device is powered on can be done by using **BOOT\_COMPLETED** broadcastReceiver .

For that we have to include permission

<uses-permission android:name="android.permission**.RECEIVE\_BOOT\_COMPLETED**"/>

Then define receiver in androidmanifest.xml file

<receiver android:name="com.prac.test.AppStarter">

<intent-filter>

<action android:name="android.intent.action.BOOT\_COMPLETED"/>

</intent-filter>

</receiver>

Then define class

public class AppStarter extends BroadcastReceiver {

@Override

public void onReceive(Context context, Intent intent) {

if(intent != null) {

String action = intent.getAction();

if(action != null) {

if(action.equals("android.intent.action.BOOT\_COMPLETED")) {

// to do code

} else if (action.equals("ANOTHER ACTION")) {

// to do code

}

}

}

}

}

**9.** If you were a malware developer and knew that an endpoint solution was using a

 YARA rule to detect your malware, how would you modify the new variant to get around this YARA rule?

To bypass the YARA rule we can try following methods –

* **Dynamic code loading**

If the app loads malicious code dynamically then it would be difficult to get detect by YARA rule.

* **Code obfuscation**

If the code of the app is heavily obfuscated and code structure is changed then we can bypass the YARA rule.

* **Encrypted text**

If we use encrypted text and decrypt while app is running then it is difficult to get detect by YARA rule.

* **Anti-analysis techniques**

In the app we can use anti-analysis techniques or environment aware execution method in which we will do that if the app is running on YARA rule then malicious code will not be executed.

* **Polymorhic and metamorphic techniques**

In the polymorphic technique code will be generated dynamically and pattern of code is changed each time using metamorphic technique.So it will be not easy to get detected by YARA rule.

**10.** How would you go about unpacking a packed application?

Packed applications are those type of application which is encrypted to some form by using advanced algorithm like **UPX packer, android protectors** or can be done manually.

Such type of applications are very complicated to do reverse engineering.

Steps for unpacking a packed application are as follows –

* **Identify the packing technique –**

First we need to identify the packing technique, by which algorithm the application is packed either it has used prebuilt packer or done manually.

We can use **apkid** tool to know which type of packer the apk has used.

* **Use packer tool –**

If the packing is done by using tool then we can unpacked it by using unpacker like

* UPX unpacker
* Ollydbg
* PEid
* If the packing is done manually then we can check for decryption routine in the code or reflection method and from there we can unpack.

After that we can decompile the application using JADX for further RE process.

**Section - 2**

**2.** Explain the code below. How could it be used maliciously?

A screenshot of a computer

Description automatically generated

In the above code we can see that ContactsContract.CommonDataKinds.Phone.CONTENT\_URI is being passed as uri

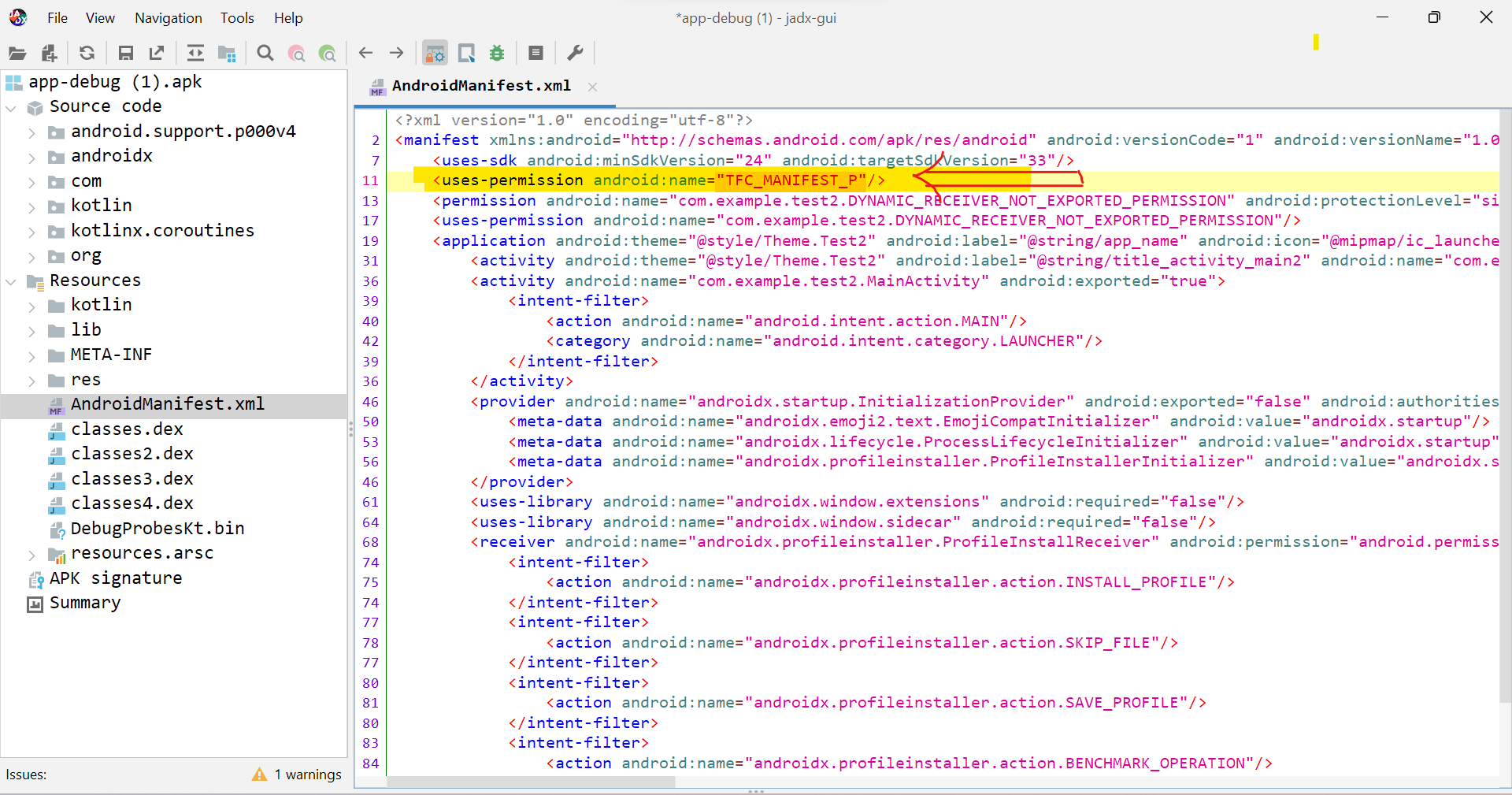
which is used to get contact list from the mobile device,

and that is storing in an arrayList.

By using the method **m8428a** to get the contact list and passing it to any http library like retrofit or okhttp, the whole contact list can be sent to remote sever.

So, in this way the above code can be used maliciously.

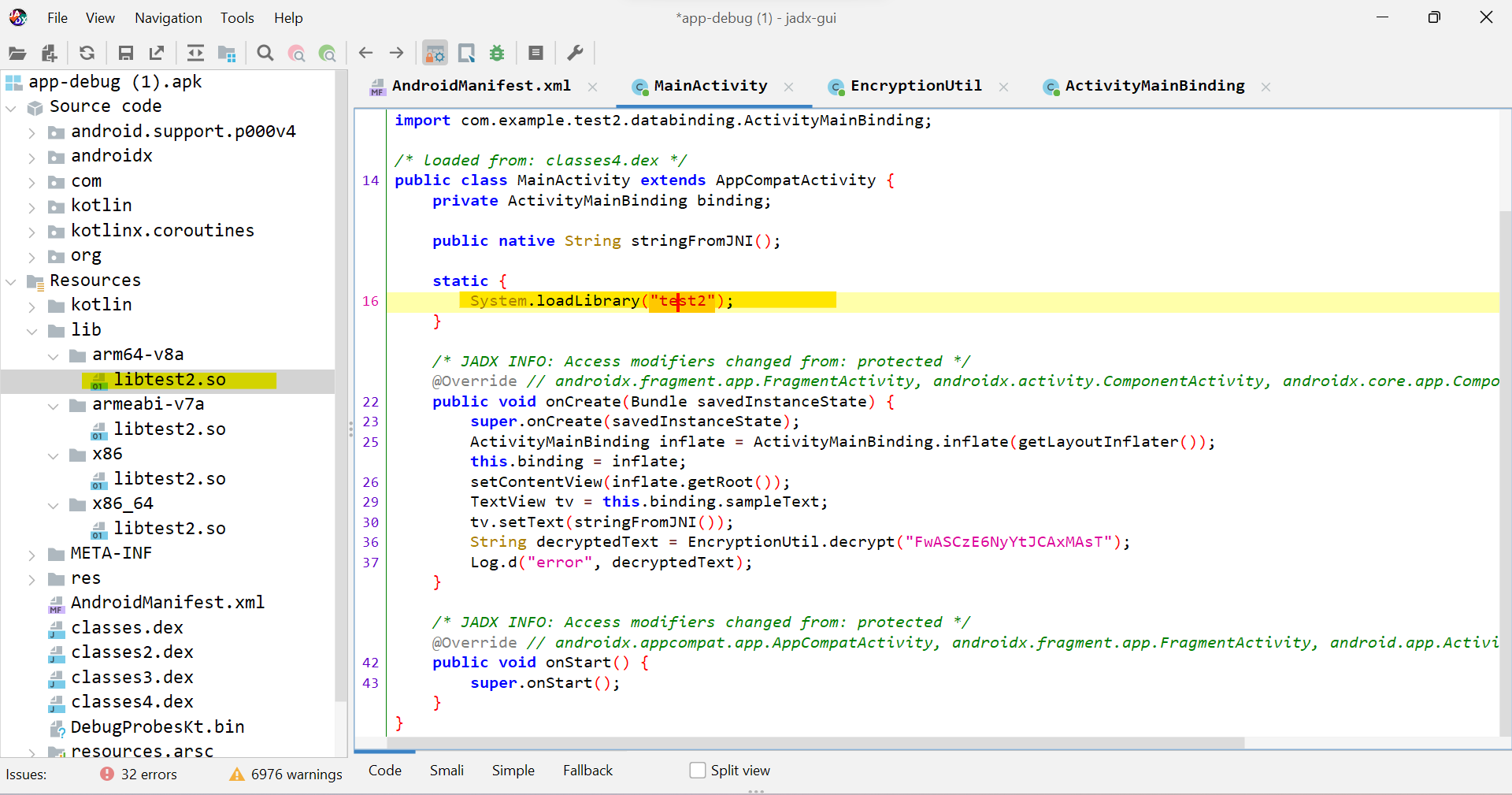
**3**.Have a look at the following App (App.apk, or . There are at least 6 “CTFs” that may contain in any order “CTF\_”, “CFT\_”, .. , apk link will be shared separately - Explain briefly with a screenshot where and how you found them



1. **TFC\_MANIFEST\_P is one of the CTFs**

* It is inside androidmanifest.xml file.
* Since we starts RE from androidmanifest.xml file there we can see user-defined permission.

1. **CTF\_NATIVE\_O is one of the CTF**



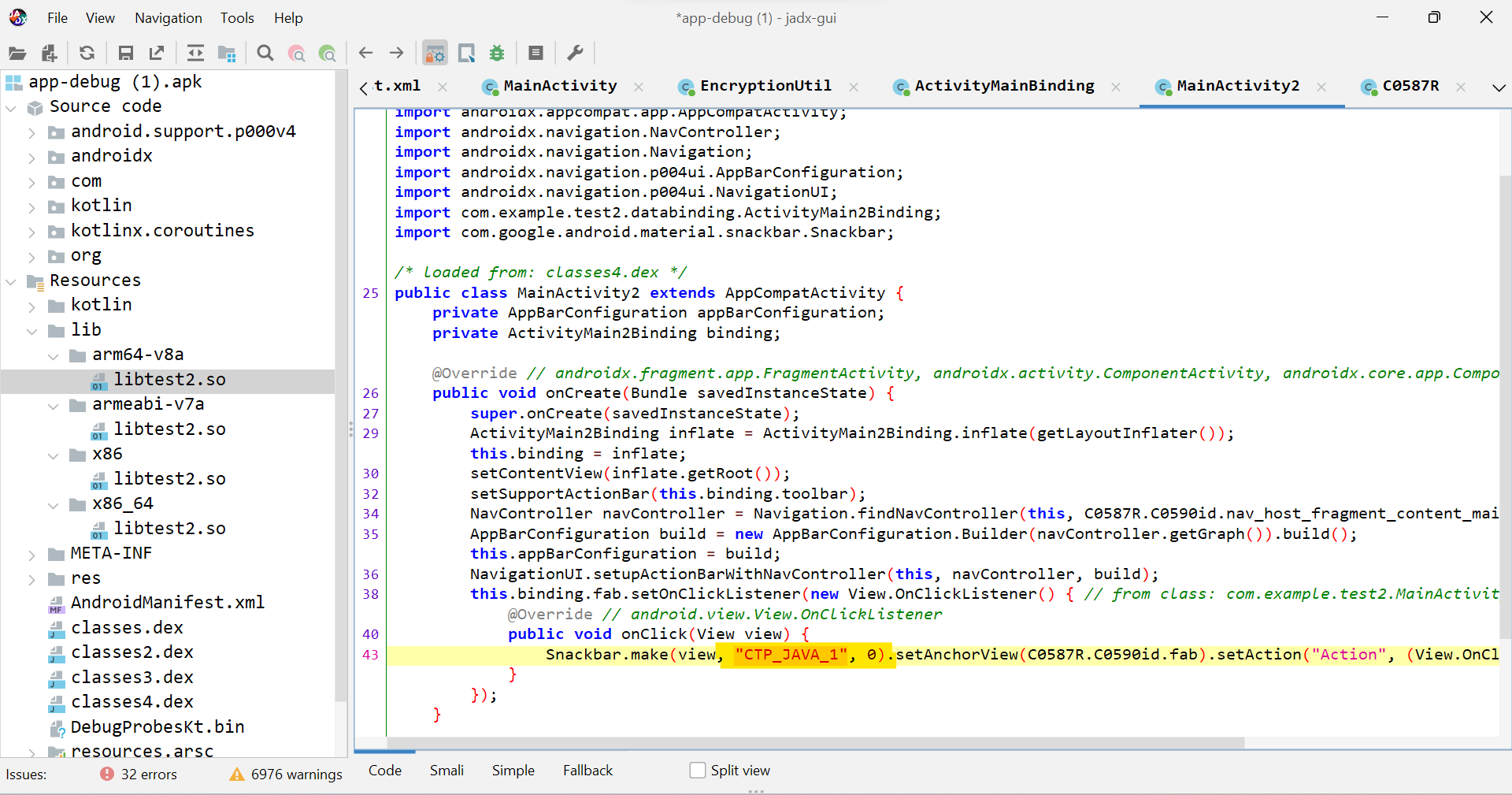
* Since the app is asking for user defined permission
* So we will start from launcher activity which MainActivity.
* There we can see that System.loadLibrary(“test2”) method is called ,this method indicates that any native library is being called
* So we will find that native library in lib folder as shown in above screenshot.
* Since this .so file we cannot open in jadx , so we will put libtest2.so file in cyberchef and search for ctf string.

Here we will get one result which is CTF variable.

A screenshot of a computer

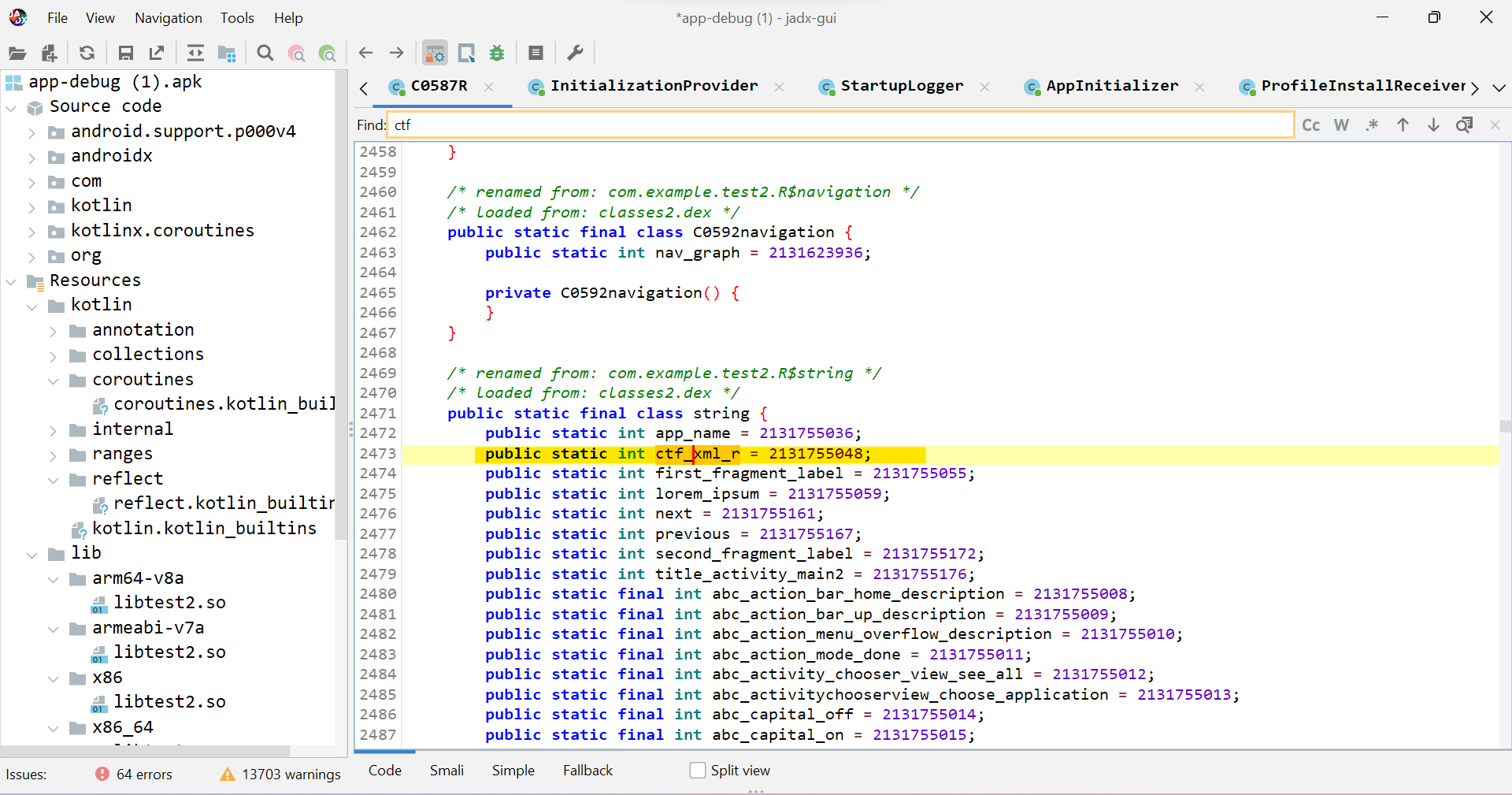
Description automatically generated

1. **CTP\_JAVA\_1 is one of the CTF**



* From the mainActivity, we are not getting anyother result and not going to anyother class
* Now we will check MainActivity2 and there we can see that “**CTP\_JAVA\_1**” is defined.

1. **ctf\_xml\_r is one of the CTF**



We will search if there is any ctf defined in string so there we can see that one ctf is defined.

1. **CFT\_XML\_R is one of the CTF**

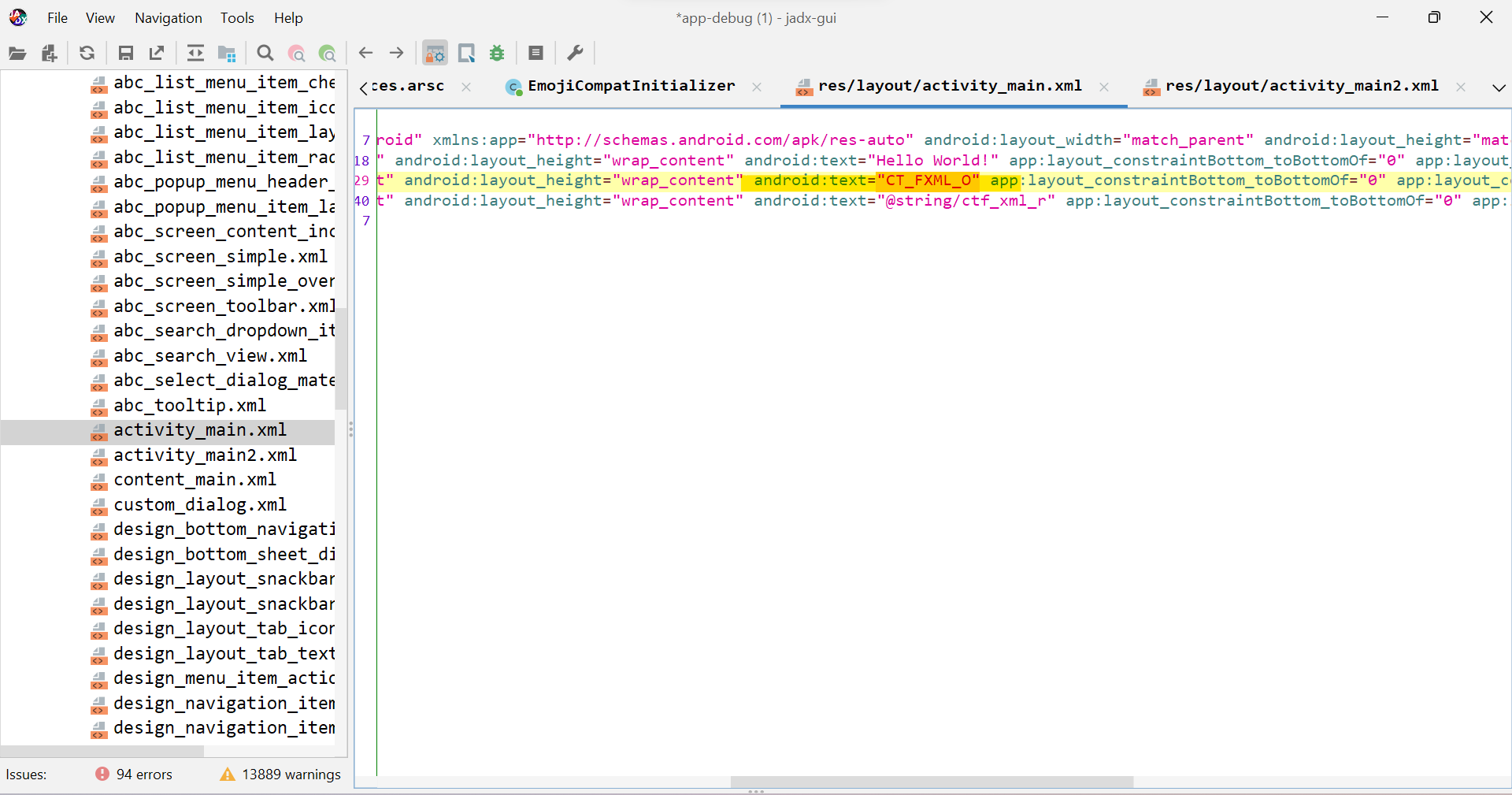
A screenshot of a computer

Description automatically generated

* We will search in strings.xml also

There we can see CFT\_XML\_R string.

1. **CT\_FXML\_O is one of the ctf**



We will see this inside activity\_main.xml