3-Reverse Enginerring Android App

what is the Reverse Enginerring in Pentest

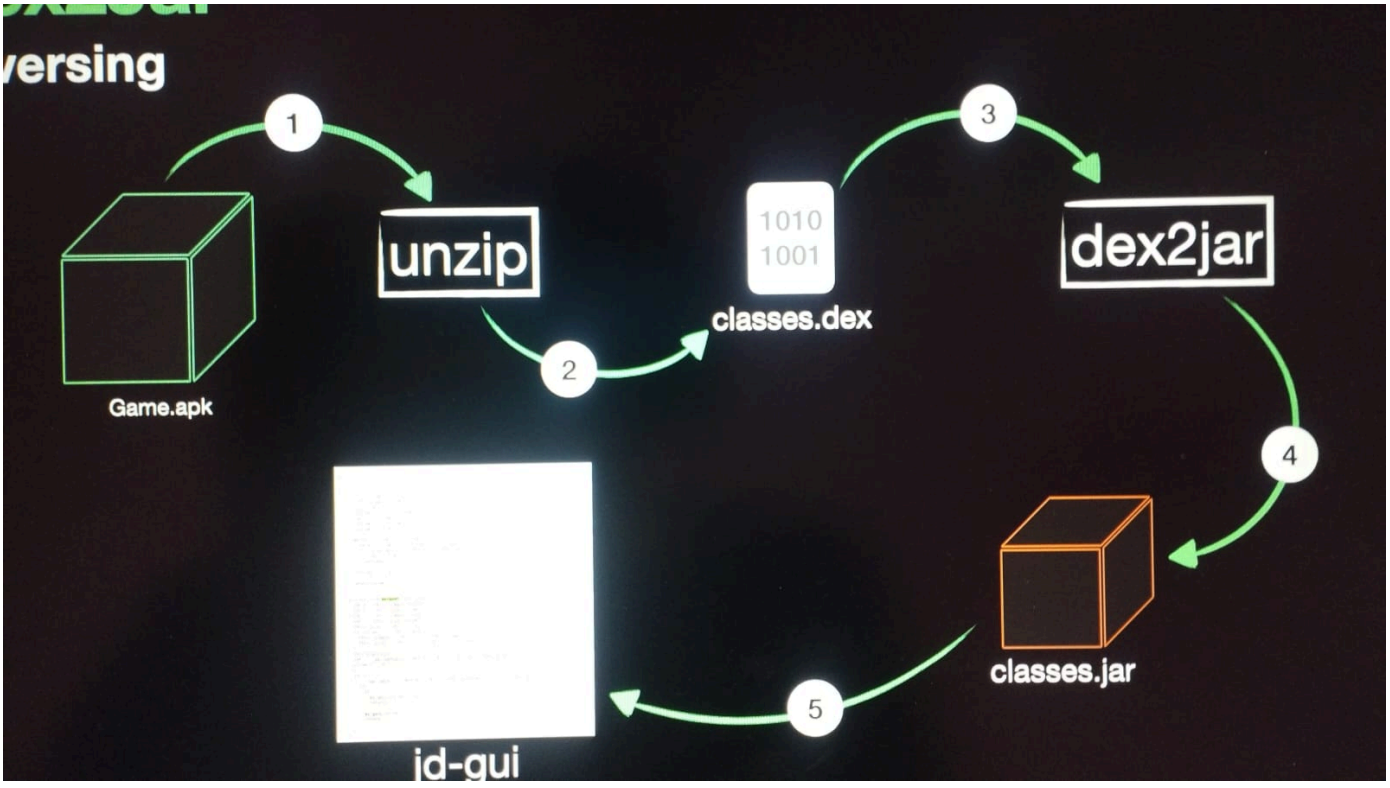
Reverse engineering is a technique used to **analyze a system, software, or hardware to understand its inner workings without access to its source code, design, or original documentation**. In penetration testing, reverse engineering is a **critical skill for discovering vulnerabilities, analyzing malware, or understanding proprietary protocols.**

Reverse engineering Android apps involves analyzing and deconstructing APK (Android Package) files to understand their inner workings. This is a crucial skill in penetration testing for **identifying security weaknesses, verifying security controls, or analyzing potentially malicious apps.**

38-Dex2jar

دلوقتي احنا عارفين لو عاوزين نفكك APK file اما باستخدام unzip او apktool ولو فككناه باستخدام unzip هيكون classes.dex فايل وده عبارة عن code binary for ولو فككناه باستخدام apktool هيكون مجموعة ملفات مكتوبة بلغة smali

فدلوقتي احنا لما نستخدم الطريقة اللي هي unzip فكده هيكون classes.dex احنا بقي عاوزين نحول الملف ده لكود java --> classes.jar هيبقي في زي كود ملخص للتطبيق نقدر نحلل من apk app



intsall tool :

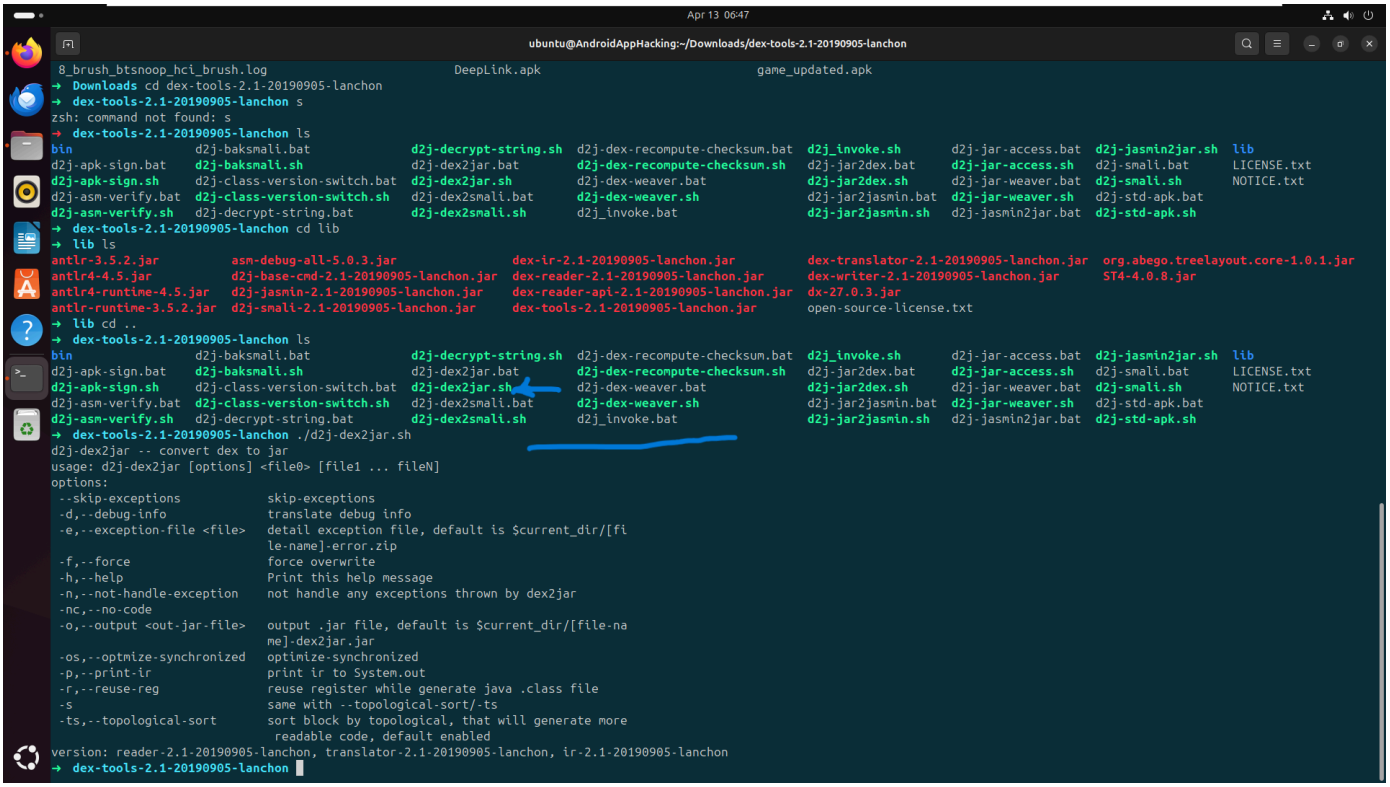
https://github.com/DexPatcher/dex2jar/releases

from this install dex-tools-2.1-20190905-lanchon.zip and unzip it

after unzip will contain this files

d2j-dex2jar.sh for linux

d2j-dexjar.bat for windows



convert from classes.dex to classes.jar, and this will create file classes.jar, and to work with it you should to install java compiler

```
dex2jar classes.dex
```

after install java decompile run it

```
java -jar jd-gui-1.6.6.jar
```

and open file classes.jar on java decompile or create alias name on ~/.zshrc -->

```
nano ~/.zshrc
alias jd-gui="java -jar jd-gui-1.6.6.jar"
source ~/.zshrc
jd-gui classes.jar
```

39- jadx , jadx-gui

---

to install jadx from github

```
➜  tools git clone https://github.com/skylot/jadx.git
Cloning into 'jadx'...
remote: Enumerating objects: 53457, done.
remote: Counting objects: 100% (2810/2810), done.
remote: Compressing objects: 100% (214/214), done.
remote: Total 53457 (delta 2665), reused 2596 (delta 2596), pack-reused
50647 (from 2)
Receiving objects: 100% (53457/53457), 22.95 MiB | 3.39 MiB/s, done.
Resolving deltas: 100% (29375/29375), done.
```

```
➜  jadx git:(master) ./gradlew dist
Downloading https://services.gradle.org/distributions/gradle-8.13-bin.zip
.............10%.............20%.............30%.............40%.............
.50%.............60%.............70%.............80%.............90%........
.....100%

Welcome to Gradle 8.13!

Here are the highlights of this release:
 - Daemon JVM auto-provisioning
 - Enhancements for Scala plugin and JUnit testing
 - Improvements for build authors and plugin developers

For more details see https://docs.gradle.org/8.13/release-notes.html

Starting a Gradle Daemon (subsequent builds will be faster)

> Configure project :
jadx version: dev

> Configure project :jadx-gui
Launch4J classpath: [%EXEDIR%/lib/jadx-gui-dev-all.jar]

> Task :jadx-core:compileJava
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

> Task :jadx-cli:compileJava
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
```

```
> Task :jadx-plugins:jadx-java-input:compileJava
Note: /home/ubuntu/tools/jadx/jadx-plugins/jadx-java-
input/src/main/java/jadx/plugins/input/java/JavaInputPlugin.java uses or
overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

> Task :jadx-gui:compileJava
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

[Incubating] Problems report is available at:
file:///home/ubuntu/tools/jadx/build/reports/problems/problems-report.html

BUILD SUCCESSFUL in 5m 51s
74 actionable tasks: 74 executed
➜  jadx git:(master) ls
build             buildSrc            config   CONTRIBUTING.md
gradle.properties  gradlew.bat  jadx-commons  jadx-gui      jadx-plugins-
tools  NOTICE     SECURITY.md
build.gradle.kts  CODE_OF_CONDUCT.md  contrib  gradle           gradlew
jadx-cli     jadx-core     jadx-plugins  LICENSE             README.md
settings.gradle.kts
➜  jadx git:(master) cd build
➜  build git:(master) ls
jadx  jadx-dev.zip  reports
➜  build git:(master) cd jadx
➜  jadx git:(master) ls
bin  lib  LICENSE  README.md
➜  jadx git:(master) cd bin
➜  bin git:(master) ls
jadx  jadx.bat  jadx-gui  jadx-gui.bat
➜  bin git:(master) pwd
/home/ubuntu/tools/jadx/build/jadx/bin
➜  bin git:(master) /home/ubuntu/tools/jadx/build/jadx/bin/jadx-gui
```

jadx help

```
➜  bin git:(master) jadx -h

jadx - dex to java decompiler, version: dev
```

```
usage: jadx [command] [options] <input files> (.apk, .dex, .jar, .class,
.smali, .zip, .aar, .arsc, .aab, .xapk, .apkm, .jadx.kts)
commands (use '<command> --help' for command options):
  plugins          - manage jadx plugins

options:
  -d, --output-dir                         - output directory
  -ds, --output-dir-src                    - output directory for sources
  -dr, --output-dir-res                    - output directory for resources
  -r, --no-res                             - do not decode resources
  -s, --no-src                             - do not decompile source code
  --single-class                           - decompile a single class, full
name, raw or alias
```

اهم options اللي هنستخدمها مع jadx

1- --log-level

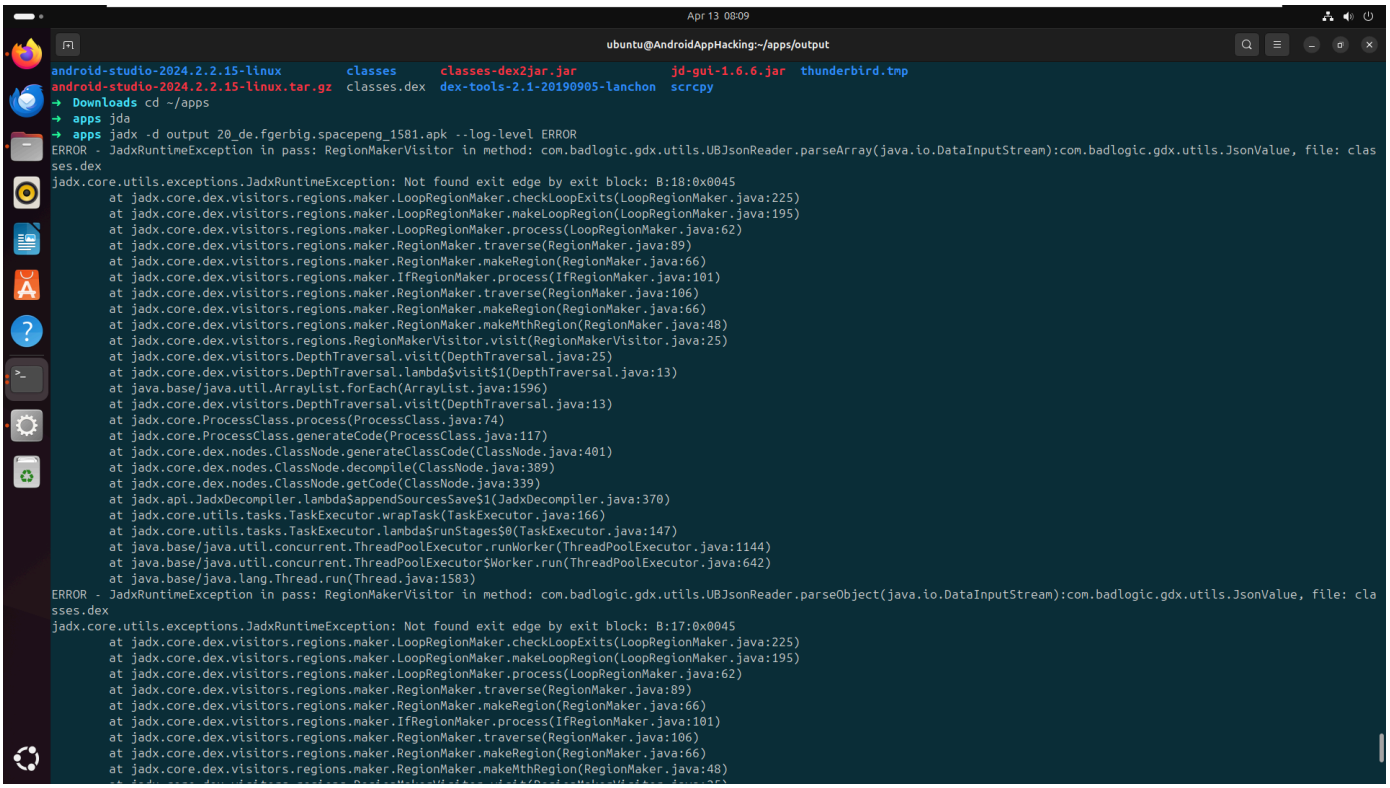
الخيار `log-level--` في **Jadx** يُستخدم لتحديد مستوى السجلات (logs) التي يتم عرضها أثناء تشغيل الأداة. هذه السجلات تُظهر المعلومات التي تُنتجها الأداة أثناء عملية فك التشفير (decompilation) أو تحليل التطبيق.

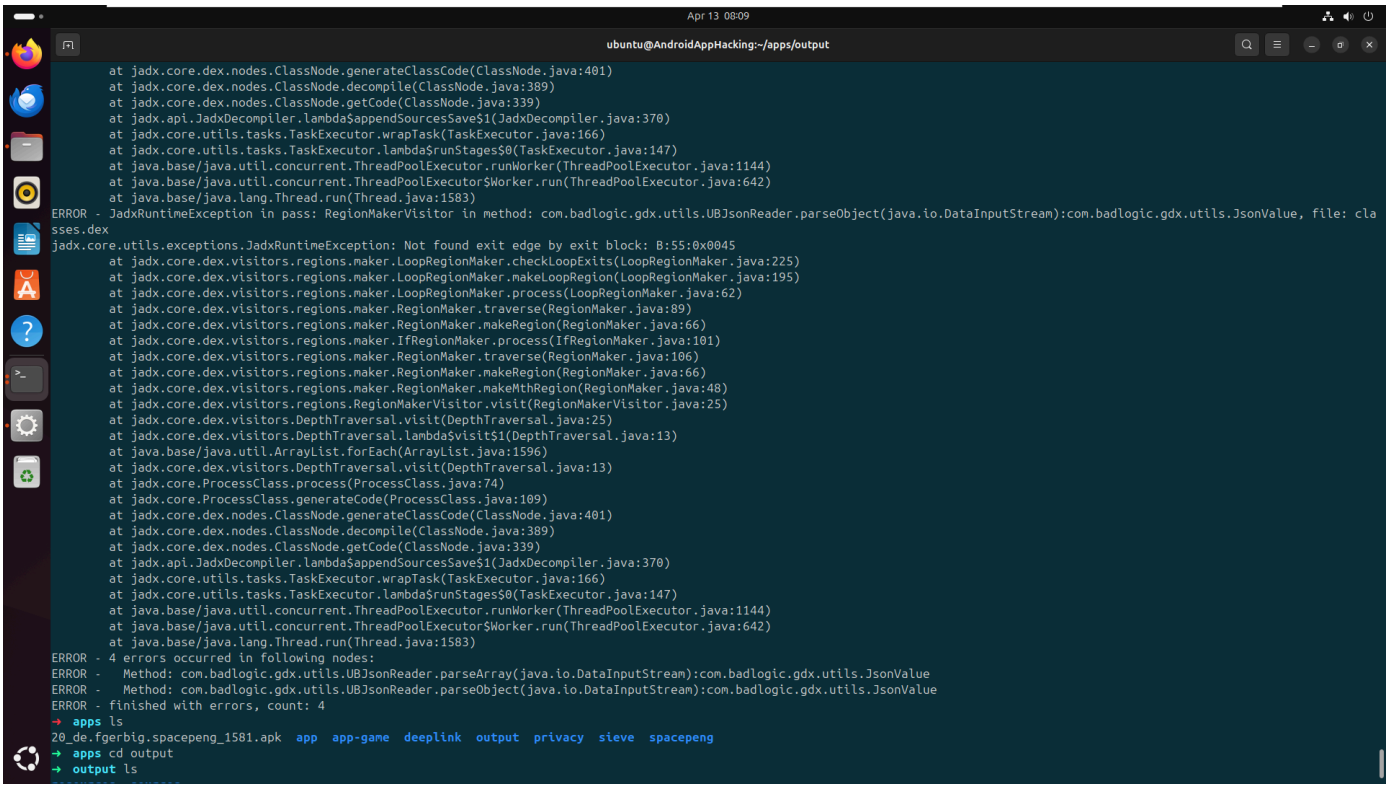
ما هو `log-level--`؟

- يُحدد كمية التفاصيل التي تريد رؤيتها أثناء تشغيل الأداة.
- المستويات الشائعة تشمل:
  - **ERROR:** عرض الأخطاء فقط.
  - **WARN:** عرض التحذيرات والأخطاء.
  - **INFO:** (الوضع الافتراضي) عرض المعلومات العامة.
  - **DEBUG:** عرض تفاصيل أكثر للتحليل والتصحيح.
  - **QUIET:** تقليل السجلات إلى الحد الأدنى.

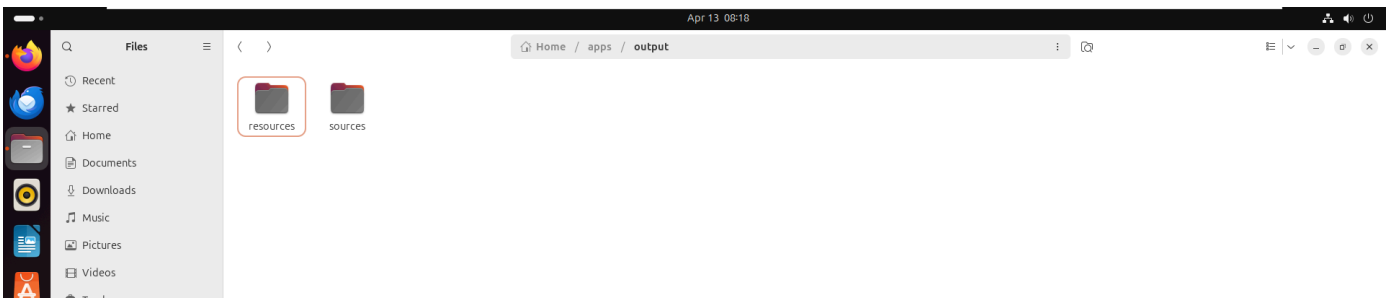
```
jadx --log-level LEVEL app.apk
```

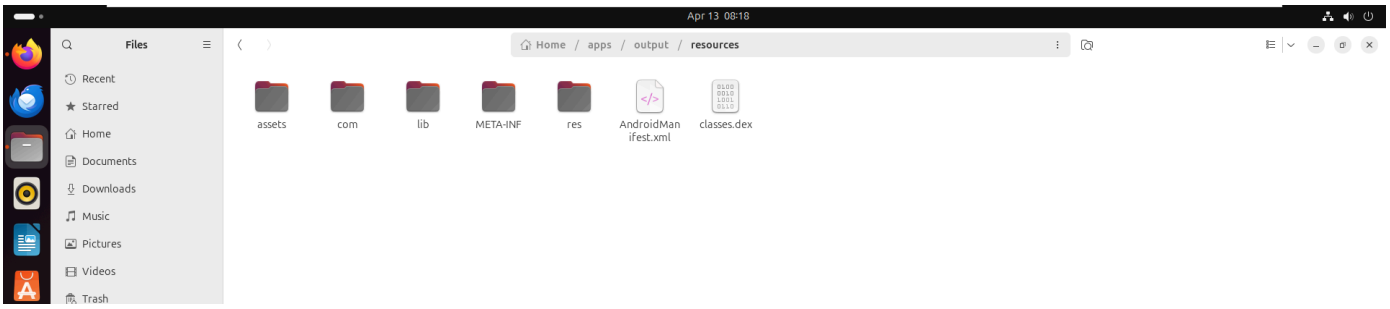
```
jadx --log-level ERROR app.apk
```

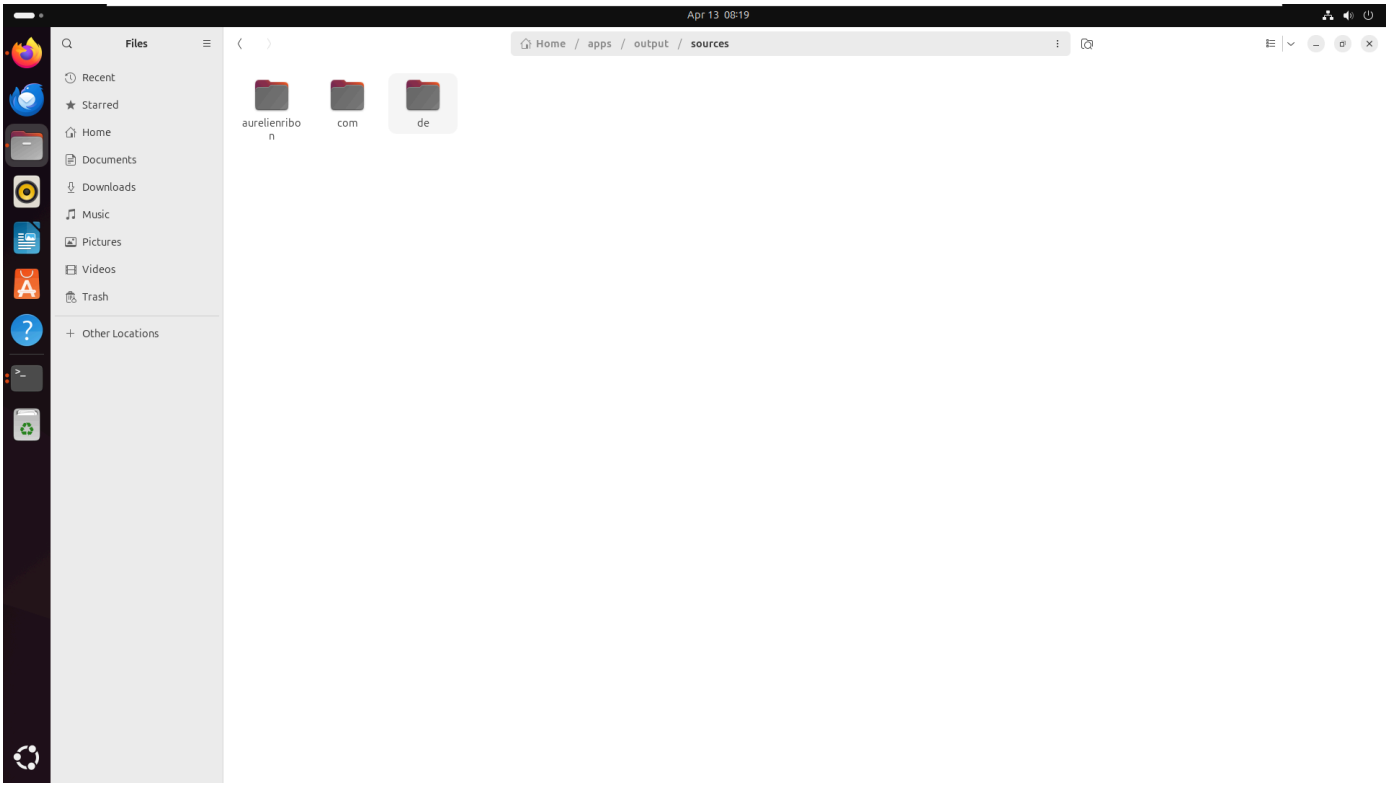




لما نفككه بقي هيكون 2 folder واحد ل resources وده اللي هو زي الملفات لما نعمل unzip or apktool وواحد لل source اللي هو بقي اكواد java







لو عاوز تفك اكتر من تطبيق في نفس folder

```
find ./ -iname "*.apk" -exec jadx {} -d out/{} \;
```

2- --deobf

يُستخدم أثناء عملية فك التشفير (decompilation) لتحليل التطبيقات التي تحتوي على **كود مُبهم** (obfuscated code). يُشير **"beobf"** إلى "best-effort obfuscation handling"، مما يعني أن الأداة ستبذل قصارى جهدها لتجاوز التشويش الذي تم تطبيقه على الكود

ما هو التشويش (Obfuscation)؟

التشويش هو عملية تُستخدم لإخفاء الكود البرمجي وجعله أكثر صعوبة للفهم أو التحليل. تُستخدم تقنيات التشويش بشكل شائع لحماية التطبيقات من الهندسة العكسية، مثل:

- تغيير أسماء المتغيرات والأساليب إلى أسماء غير مفهومة (مثل `a`, `b`, `c`).
- تقنيات الإخفاء المتقدمة مثل إضافة دوال وهمية أو إعادة ترتيب التعليمات البرمجية.

```
jadx -d output_dir -beobf app.apk
```

3- --cfg

يُستخدم لتوليد **Graph Control Flow** أو "مخطط تدفق التحكم"، وهو تمثيل رسومي يُظهر كيفية تدفق التعليمات البرمجية داخل التطبيق.

ما هو **Control Flow Graph (CFG)**؟

- هو مخطط يوضح كيف يتم تنفيذ التعليمات البرمجية في التطبيق **Control Flow Graph**:
  - يُظهر العقد (nodes) التي تمثل التعليمات أو الكتل البرمجية.
  - يُظهر الحواف (edges) التي تمثل التدفق بين هذه الكتل (مثل الحلقات، الشروط، والقفزات).
- يساعد هذا المخطط في فهم منطق التطبيق، خاصة في حالة التطبيقات المشوّشة أو التي تحتوي على تعقيد برمجي.

استخدام خيار `cfg--`

عند استخدام هذا الخيار مع أداة Jadx، تقوم الأداة بإنشاء ملفات رسومية تمثل تدفق التحكم. يُمكنك استخدام برامج عرض المخططات (مثل Graphviz) لفتح هذه الملفات وتحليلها.

صيغة الاستخدام:

```
jadx --cfg -d output_dir app.apk
```

- `![7ae6828710363ee35a5fe3f0d7b3ae62.png](_resources/5ca913e56c6b494cbceba76f6bcfbbfd.png)`
- `--cfg`: يُفعل توليد ملفات تدفق التحكم.
- `-d output_dir`: يُحدد المجلد الذي سيتم تخزين الملفات الناتجة فيه.
- `app.apk`: ملف APK الخاص بالتطبيق الذي يتم تحليله.

نتيجة الخيار `cfg--`

- يتم إنشاء ملفات `dot.` في المجلد الناتج.
- ملفات `dot.` تحتوي على مخططات تدفق التحكم، ويمكنك فتحها باستخدام أدوات مثل **Graphviz** لتحليلها.

عرض المخططات باستخدام Graphviz:

1. قم بتثبيت Graphviz (إذا لم يكن مثبتًا):
   `sudo apt install graphviz`
2. لعرض ملف `dot.`:
   `dot -Tpng input.dot -o output.png`
   - يتم تحويل المخطط إلى صورة بصيغة PNG.

40- AndroGuard tool

Androguard is a full python tool to play with Android files.

- DEX, ODEX
- APK
- Android's binary xml
- Android resources
- Disassemble DEX/ODEX bytecodes
- Basic Decompiler for DEX/ODEX files
- Frida support for easy dynamic analysis
- SQLite database to save the session

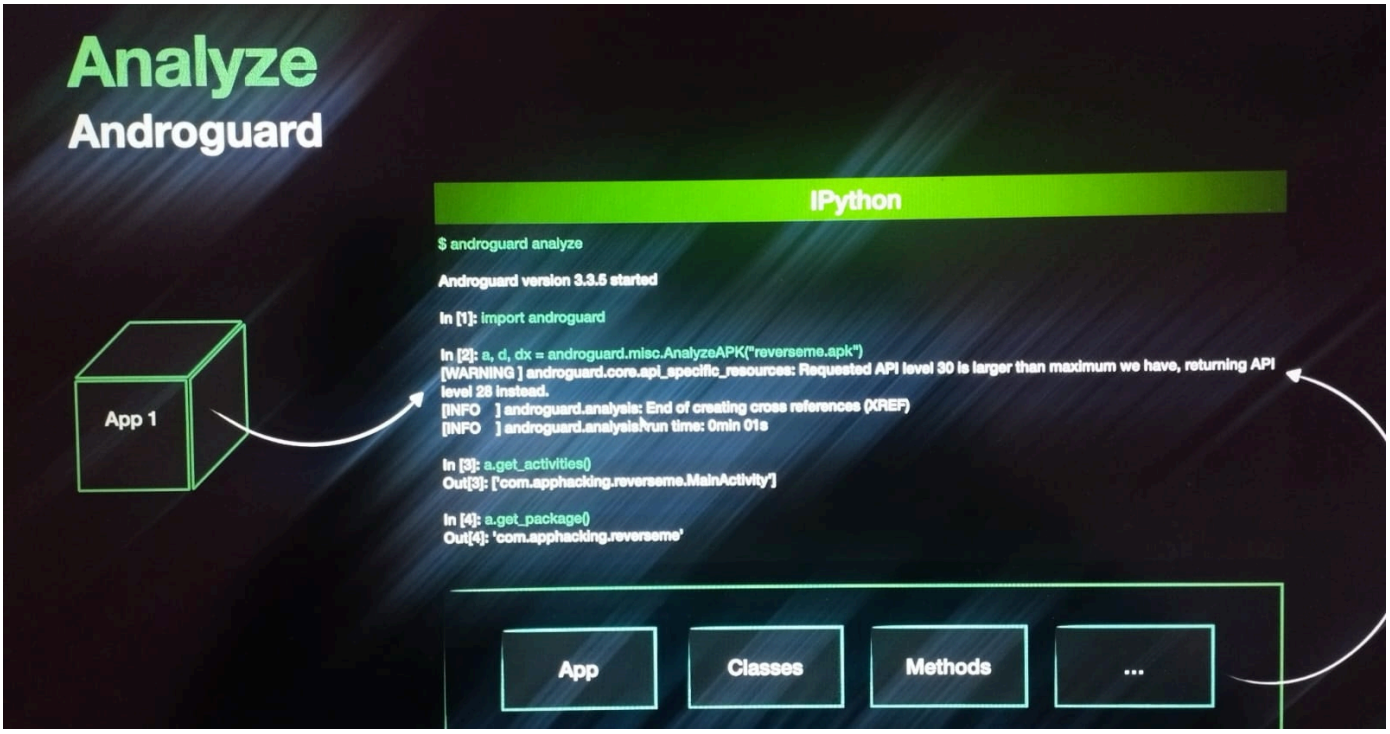
بتحتوي الاداة دي علي modules

1- analyze module

- **Analyze Module** للحصول على معلومات مفصلة حول DEX أو APK هو جزء من الأداة يُستخدم لتحليل ملفات:
  1. **الكود البرمجي** (Java أو Smali).
  2. **الأذونات** (Permissions) المطلوبة من التطبيق.
  3. **العلاقات بين الفئات (Classes) والأساليب (Methods)**.
  4. **الموارد (Resources)** مثل الملفات النصية والصور.

ماذا يحدث أثناء التحليل؟

1. **تحميل ملف APK:** يتم فك الحزمة (unpack) وتحليل محتوياتها، بما في ذلك ملفي **classes.dex** (الكود البرمجي) و **AndroidManifest.xml** (بيانات التكوين).
2. **توليد بيانات تحليلية:** يتم تحليل المكونات التالية:
   - **Manifest Analysis**: استخراج الأذونات والنشاطات (Activities) والخدمات (Services) التي يستخدمها التطبيق.
   - **Code Analysis**: استرجاع الكود البرمجي (Decompiled Code) وتحليل الأساليب والدوال.
3. **عرض النتائج:** يتم عرض بيانات تحليلية مفصلة، مثل أسماء الفئات والأساليب المستخدمة وأي روابط مشبوهة قد تشير إلى سلوك ضار.



1. a - APK Object

- هذا الكائن يمثل التطبيق نفسه كملف APK.
- يمكن من خلاله الوصول إلى المعلومات الوصفية للتطبيق مثل:
  - البيانات الموجودة في **AndroidManifest.xml**.
  - قائمة الملفات والموارد داخل التطبيق.
  - التوقيعات الرقمية (digital signatures) للتطبيق.

مثال:

```
print(a.get_package()) # للحصول على اسم الحزمة (Package Name)
```

---

2. d - DEX Object

- هذا الكائن يمثل ملفات **Dalvik Executable (DEX)** داخل التطبيق.
- ملفات DEX تحتوي على الكود القابل للتنفيذ (الـ Bytecode) الذي يتم تشغيله على أجهزة أندرويد.
- يسمح لك بفحص وتحليل الـ Bytecode مباشرة.

مثال:

```
for dex in d:
    print(dex.get_methods())  # للحصول على جميع الميثودز الموجودة في ملفات DEX
```

---

3. dx - Analysis Object

- هذا الكائن يتم إنشاؤه بعد القيام بعملية تحليل عميقة للكود باستخدام Androguard.
- يوفر أدوات متقدمة لتحليل الـ Bytecode، مثل:

- تحليل التدفق (Control Flow Analysis).
- استخراج العلاقات بين الفئات (Classes) والميثودز.
- البحث عن استدعاءات مشبوهة أو ثغرات أمنية محتملة.

مثال:

```
print(dx.get_classes()) # للحصول على جميع الكلاسات مع التحليل
```

خلاصة العلاقة بين الثلاثة:

- **a**: لتحليل البنية العامة للتطبيق (ملف APK).
- **d**: للوصول إلى كود الـ Bytecode الموجود في ملفات DEX.
- **dx**: لتحليل الكود وعلاقاته بشكل متعمق.

استخدام Analyze Module

لتفعيل التحليل، يمكن استخدام الأمر التالي:

```
androguard analyze -i app.apk
```

ex: analyze apk file and get function on this app

هنا هو بيكون 3 classes واحد ل (apk ---> a) والتاني (dex --> d) والتالت (dx)وعلشان تتعمل مع function اللي في كل واحد بتحط class_name (a |d | dx) .tap

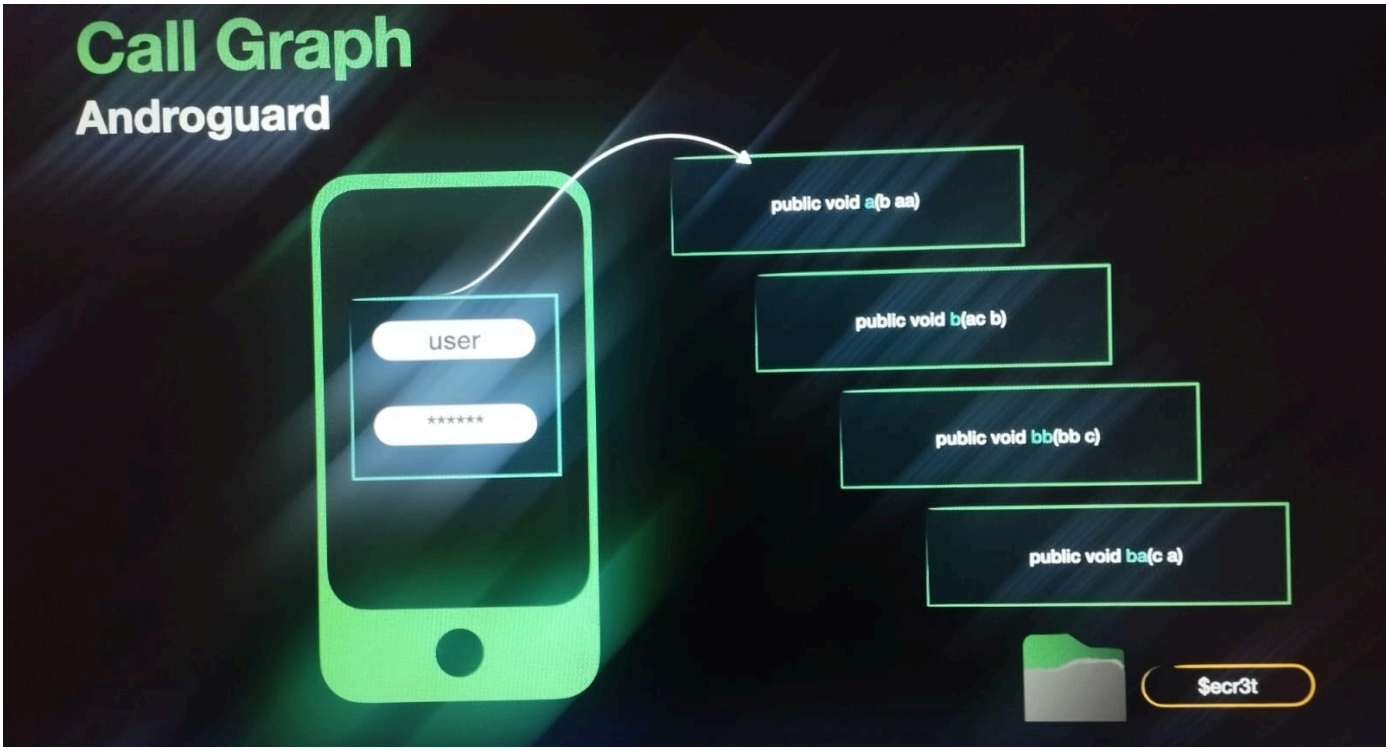
```
➜  Androgurd androguard analyze  ~/apps/20_de.fgerbig.spacepeng_1581.apk
>>> filename
/home/ubuntu/apps/20_de.fgerbig.spacepeng_1581.apk
>>> a
<androguard.core.apk.APK object at 0x7b95d3b18b30>
>>> d
[<androguard.core.dex.DEX object at 0x7b95d3b1bc50>]
>>> dx
<analysis.Analysis VMs: 1, Classes: 1551, Methods: 15329, Strings: 15672>

Androguard version 4.1.3 startedTip: `?` alone on a line will brings up
IPython's help

In [1]: a.get_activities()
Out[1]: ['de.fgerbig.spacepeng.android.AndroidLauncher']
In [2]: d.clear
Out[2]: <function list.clear()>
In [3]: dx.get_android_api_usage
Out[3]: <bound method Analysis.get_android_api_usage of <analysis.Analysis
```

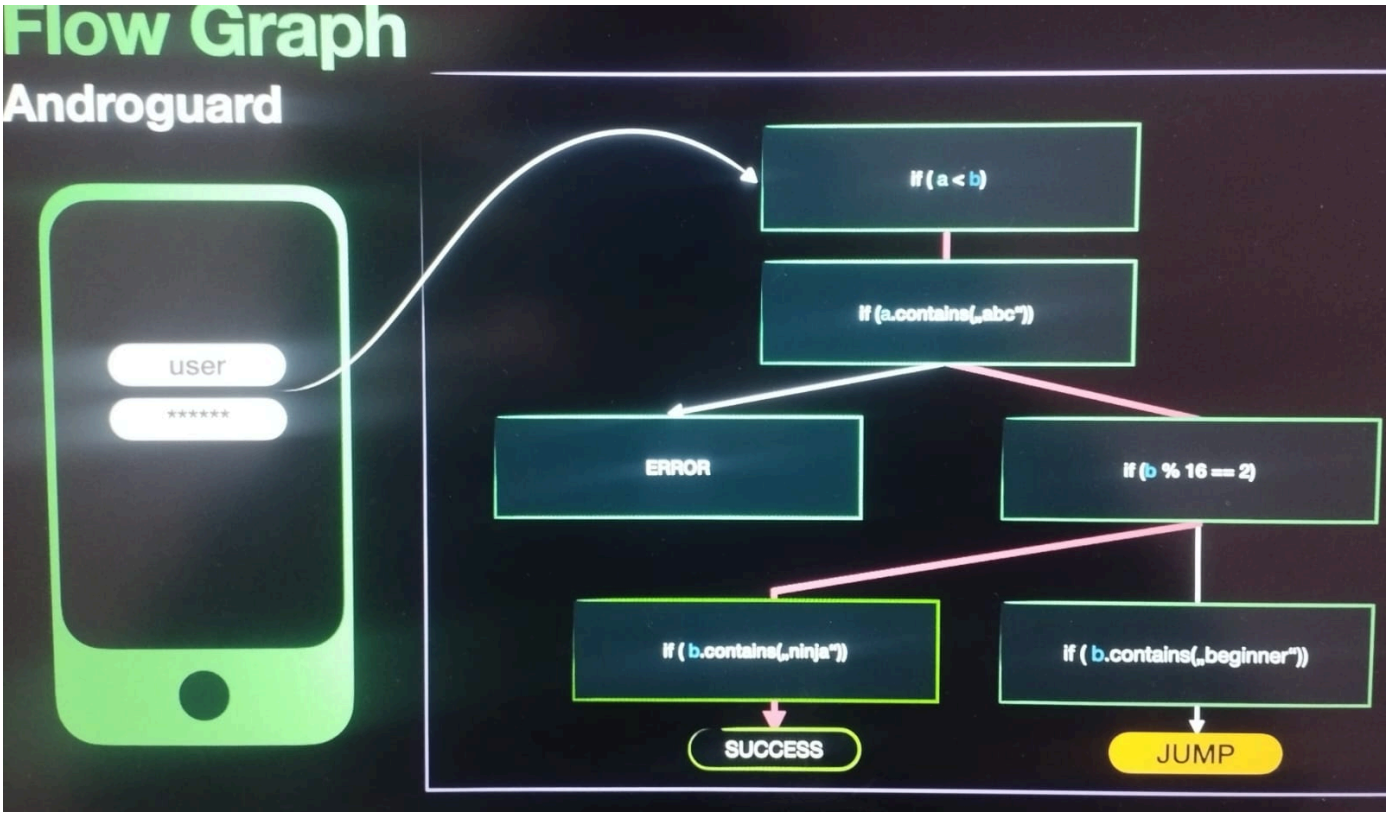
```
VMs: 1, Classes: 1551, Methods: 15329, Strings: 15672>>
```

2- CG mode

A Call Graph:

- Represents **nodes** (functions/methods) and **edges** (calls between functions).
- Helps to visualize the flow of function calls in an application.
- Is useful for analyzing:
  - Code structure.
  - Potential vulnerabilities or misconfigurations.
  - Complex interactions between methods.





Generate the Call Graph: Use the `cg` command to activate Call Graph mode:

```
androguard cg -i app.apk -o output.dot
```

Visualize the Call Graph: The `.dot` file can be visualized using tools like **Graphviz**:

- Install Graphviz:
  `sudo apt install graphviz`
- Generate an image (e.g., PNG):
  `dot -Tpng output.dot -o call_graph.png`
- Open the image to explore the Call Graph.

42- Call Graph

هنشرح دلوقتي ازاي نتعامل مع apk file من خلال ان احنا هنحوله ل graph

1-convert apk to graph by using androguard

```
➜  apps androguard cg -o callgraph.gml reverseme.apk
```

callgraph.gml is write with smali code

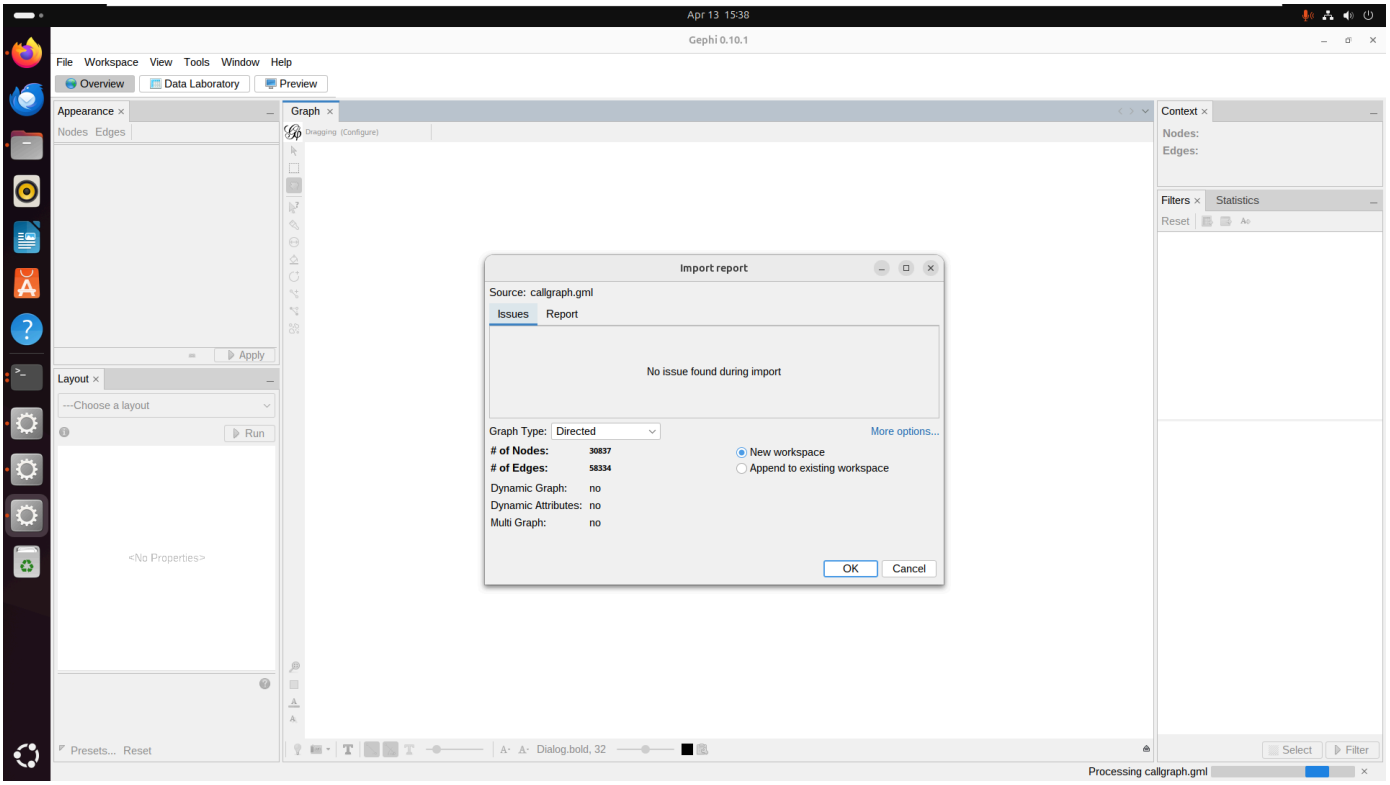
```
graph [
  directed 1
  node [
    id 0
```

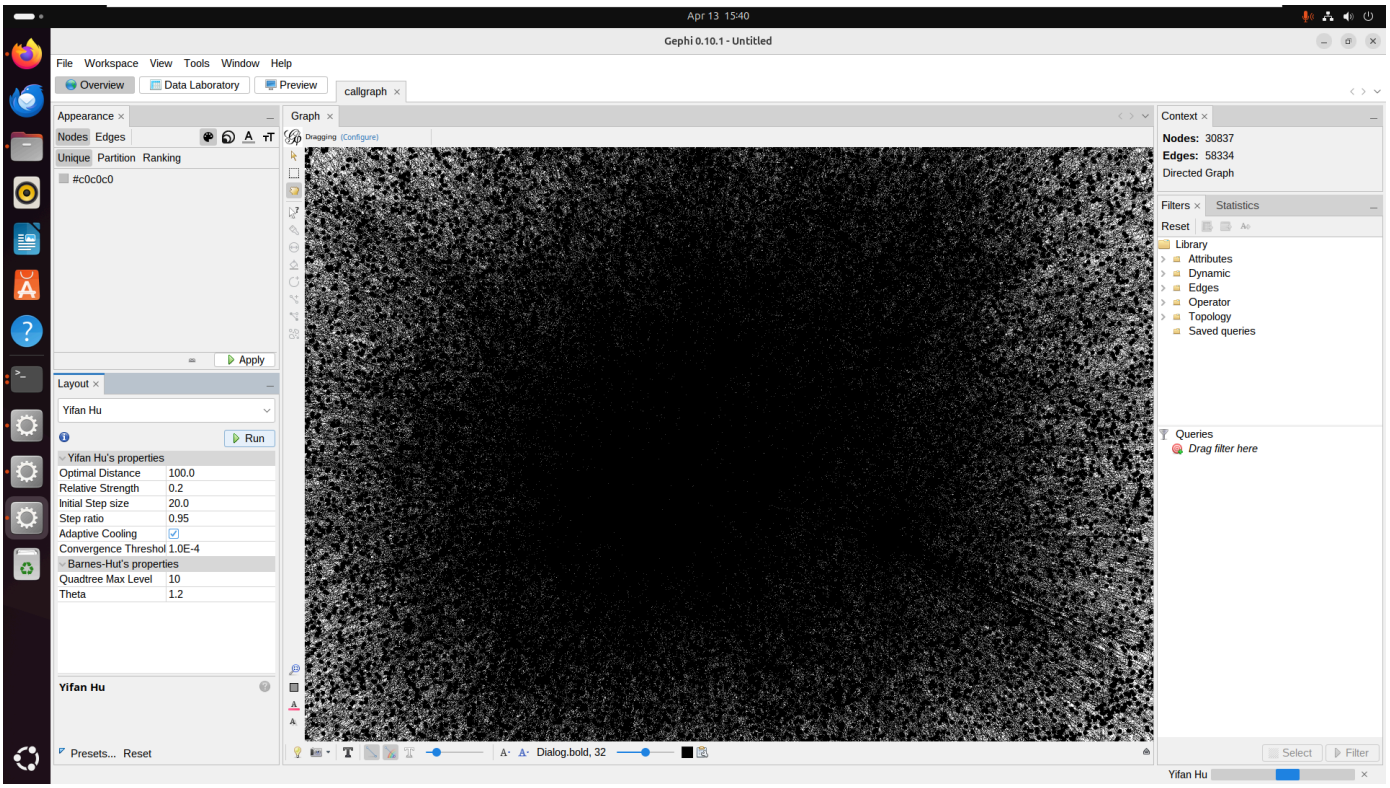
```
    label "Landroid/support/v4/app/INotificationSideChannel;-
>cancel(Ljava/lang/String; I Ljava/lang/String;)V [access_flags=public
abstract] @ 0x0"
    external 0
    entrypoint 0
    methodname "cancel"
    descriptor "(Ljava/lang/String; I Ljava/lang/String;)V"
    accessflags "public abstract"
    classname "Landroid/support/v4/app/INotificationSideChannel;"
  ]
```

2- install tool work with this graph form https://gephi.org/users/download/

after install tool run tool, and open it, and select callgraph.gml on website

he create 30837 nodes and 58334 edges

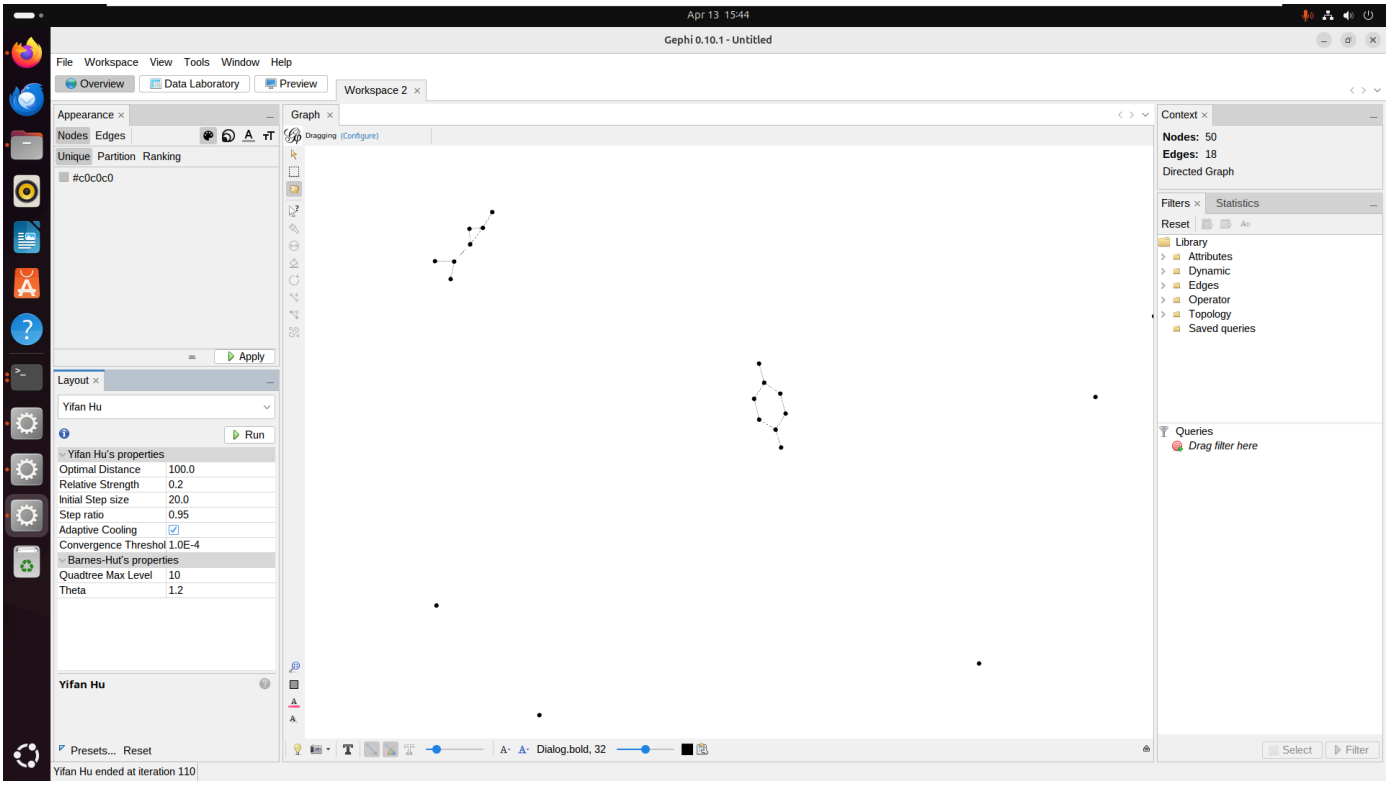




go to Data Laboratory --> select label --> write app name (com.apphacking.reverseme)

select all and click right mouse and move to new workspace

after filter for app only



43-Creating a FlowGraph (CTF)

https://challs.reyammer.io/challenges

هنا اللي challenge اللي هنحله دلوقتي هو upos ننزل بقه

1-unzip upos apk

```
➜  upos_challange unzip upos.apk
Archive:  upos.apk
  inflating: AndroidManifest.xml
➜  upos_challange ls
AndroidManifest.xml  assets  classes2.dex  classes.dex  META-INF  res
resources.arsc  upos.apk
```

2-convert from dex to jar and open .jar with jd-gui

```
➜  upos_challange dex2jar classes2.dex
dex2jar classes2.dex -> ./classes2-dex2jar.jar
➜  upos_challange jd-gui classes2-dex2jar.jar
```

3- use jadx with upos.apk

بس مش هيرضي يفتحه علشان بيتتطلب java version اقل من اللي موجودة

```
➜  upos_challange jadx-gui upos.apk
Error: A JNI error has occurred, please check your installation and try
again
Exception in thread "main" java.lang.UnsupportedClassVersionError:
jadx/gui/JadxGUI has been compiled by a more recent version of the Java
Runtime (class file version 55.0), this version of the Java Runtime only
recognizes class file versions up to 52.0
    at java.lang.ClassLoader.defineClass1(Native Method)
    at java.lang.ClassLoader.defineClass(ClassLoader.java:756)
    at
java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
    at java.net.URLClassLoader.defineClass(URLClassLoader.java:473)
    at java.net.URLClassLoader.access$100(URLClassLoader.java:74)
    at java.net.URLClassLoader$1.run(URLClassLoader.java:369)
    at java.net.URLClassLoader$1.run(URLClassLoader.java:363)
    at java.security.AccessController.doPrivileged(Native Method)
```

```
    at java.net.URLClassLoader.findClass(URLClassLoader.java:362)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:418)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:352)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:351)
    at sun.launcher.LauncherHelper.checkAndLoadMain(LauncherHelper.java:621)
```

4- convert to java version select version 11 for java and java compiler

```
 1554  jadx-gui upos.apk
 1555  sudo update-alternatives --config java
 1556  sudo update-alternatives --config javac
```

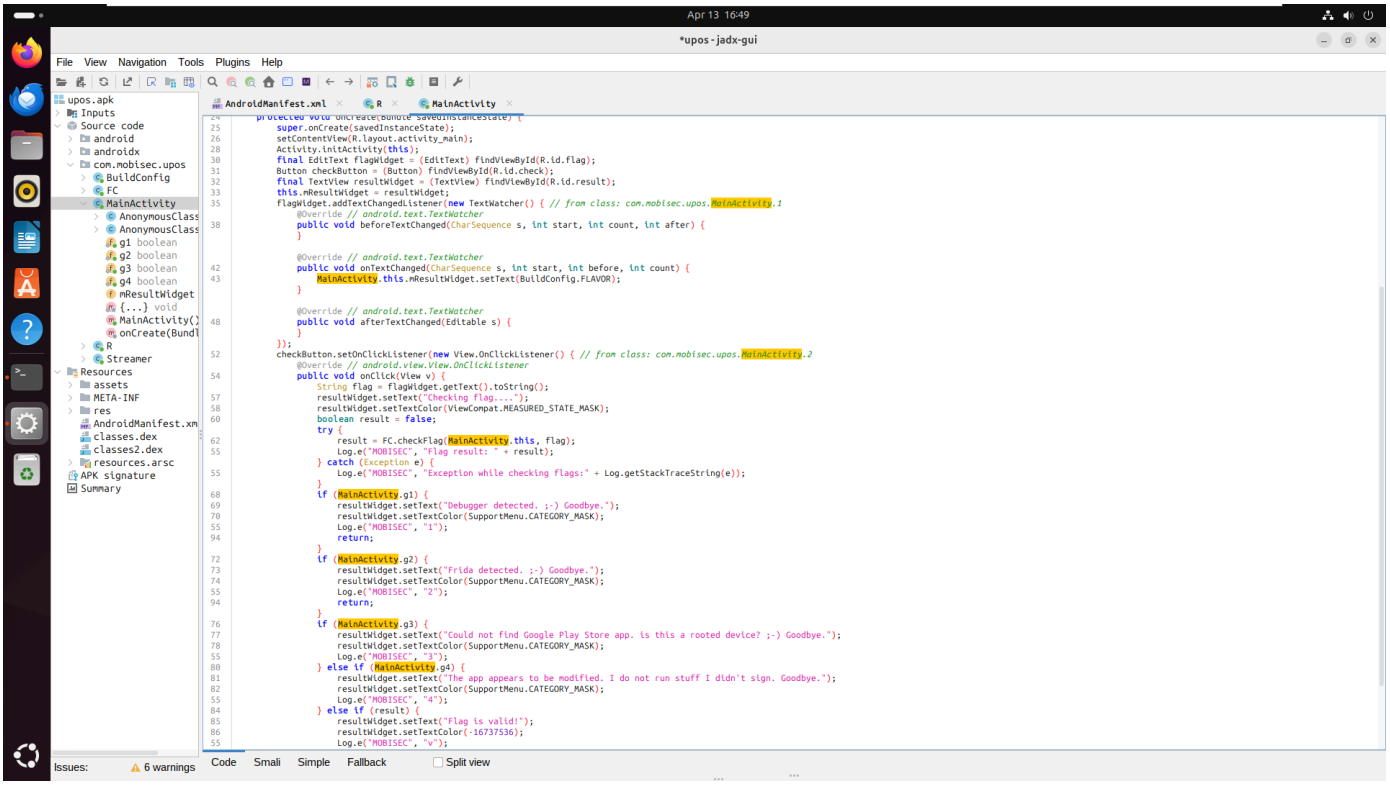
```
  ➜  upos_challange sudo update-alternatives --config java
[sudo] password for ubuntu:
There are 2 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                            Priority
Status
------------------------------------------------------------
  0            /usr/lib/jvm/java-21-openjdk-amd64/bin/java      2111
auto mode
  1            /usr/lib/jvm/java-21-openjdk-amd64/bin/java      2111
manual mode
* 2            /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java   1081
manual mode

Press <enter> to keep the current choice[*], or type selection number: 0
update-alternatives: using /usr/lib/jvm/java-21-openjdk-amd64/bin/java to
provide /usr/bin/java (java) in auto mode
➜  upos_challange sudo update-alternatives --config javac
There are 2 choices for the alternative javac (providing /usr/bin/javac).

  Selection    Path                                          Priority
Status
------------------------------------------------------------
  0            /usr/lib/jvm/java-21-openjdk-amd64/bin/javac   2111      auto
mode
  1            /usr/lib/jvm/java-21-openjdk-amd64/bin/javac   2111
manual mode
* 2            /usr/lib/jvm/java-8-openjdk-amd64/bin/javac    1081
manual mode

Press <enter> to keep the current choice[*], or type selection number: 0
```

```
update-alternatives: using /usr/lib/jvm/java-21-openjdk-amd64/bin/javac to provide /usr/bin/javac (javac) in auto mode
```

use jadx

```
jadx-gui upos.apk
```

analyze the MainActivity

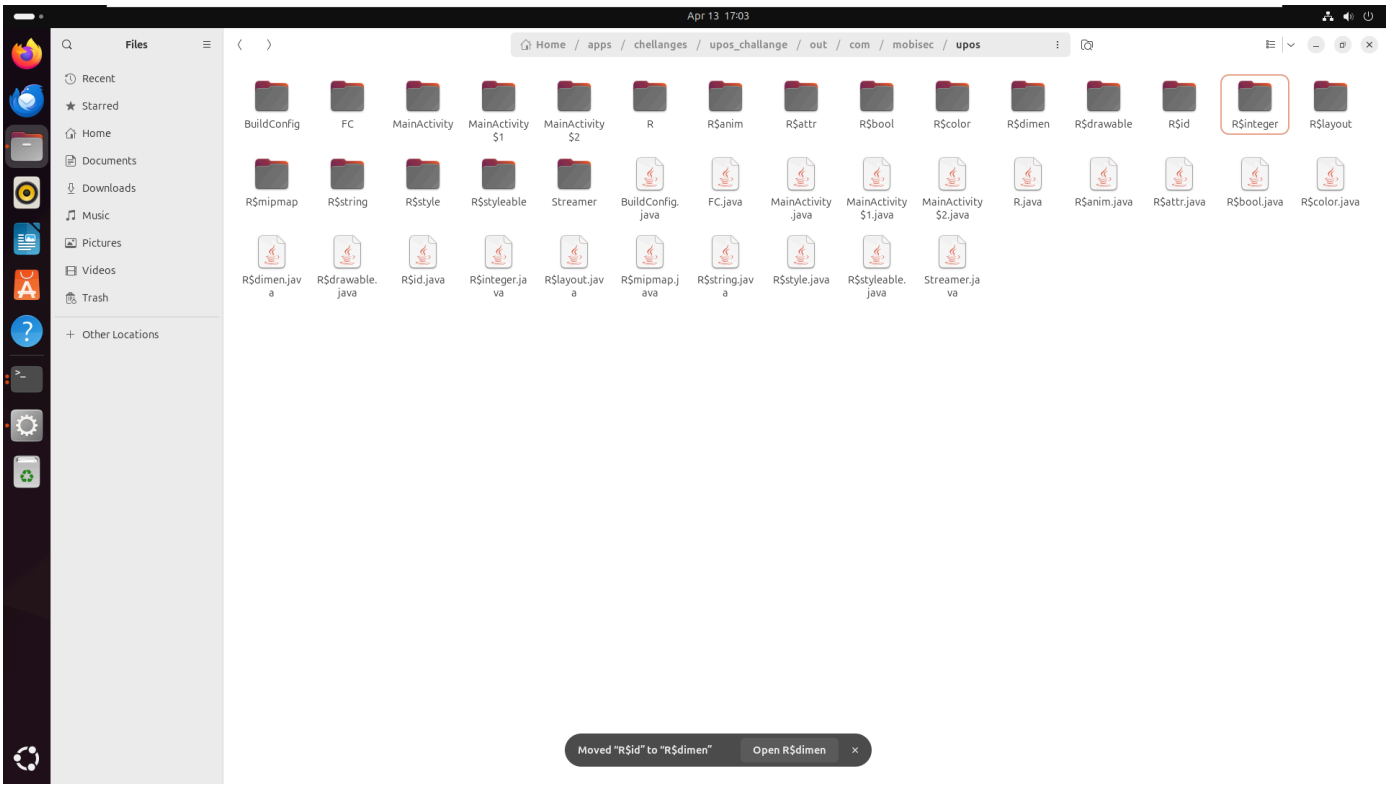


5- use androguard with decompile mode

decompile --> Decompile an APK and create Control Flow Graphs.

```
➜  upos_challange androguard decompile -o out/ upos.apk
```

after decompile it open FC.java file on com folder



6- create FlowGraph by using androguard decompile -f (fromat) file.png

To get all CFG in png format and limit the processing only to a certain namespace, the following command can be used:

```
androguard decompile -o outputfolder -f png -i someapp.apk --limit
"^Lcom/elite/.*"
```

Please make sure that graphviz and pydot are installed.

```
$ sudo apt-get install graphviz
$ pip install -U pydot
```

This will decompile the app someapp.apk into the folder outputfolder and limit the processing to all methods, where the classname starts with com.elite..

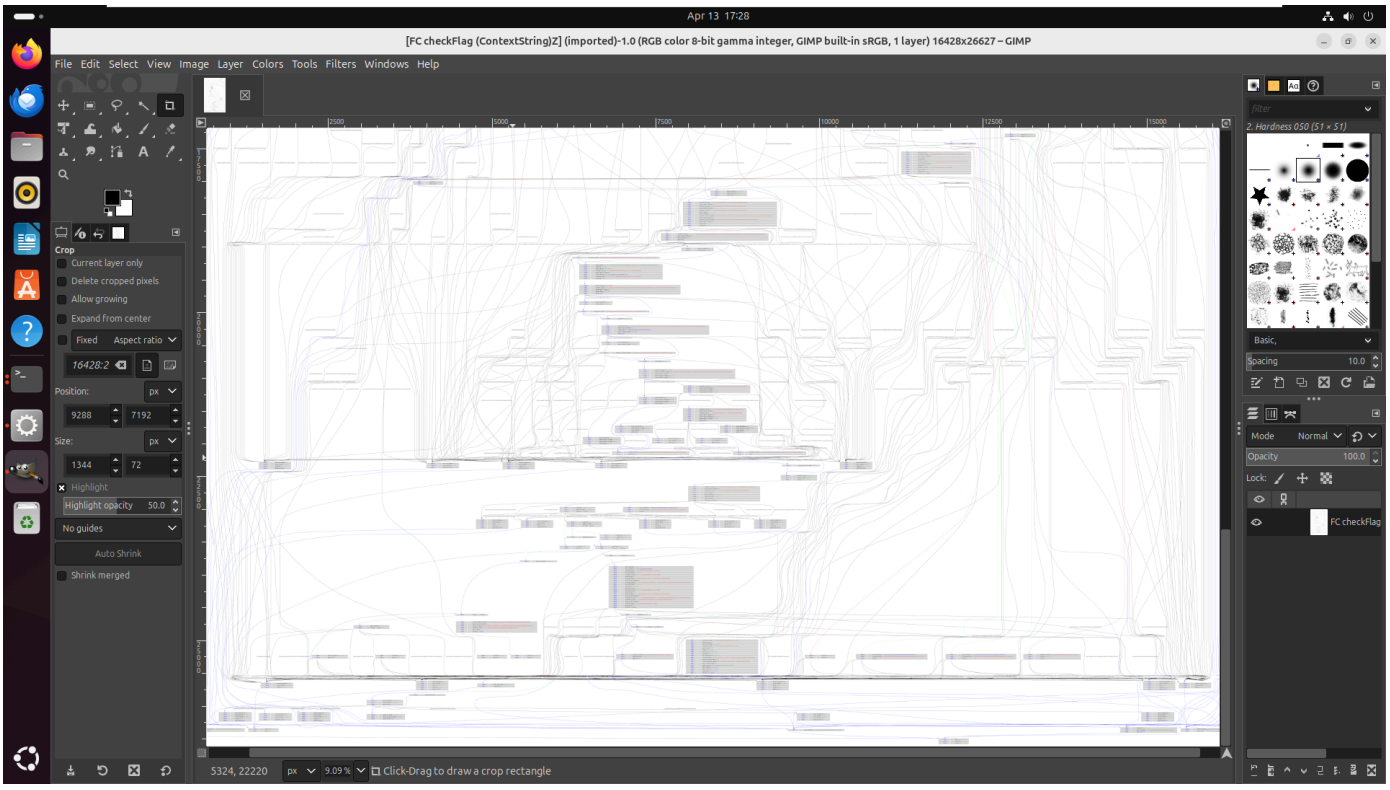
decompile the app

```
➜  upos_challange androguard decompile -o out_fg -f png --limit
"^Lcom/mobisec/" upos.apk
Dump information upos.apk in out_fg
Create directory out_fg
Decompilation ... End
Decompilation ... End
Dump Lcom/mobisec/upos/BuildConfig; <clinit> ()V ... png ... source codes
... bytecodes ...
```

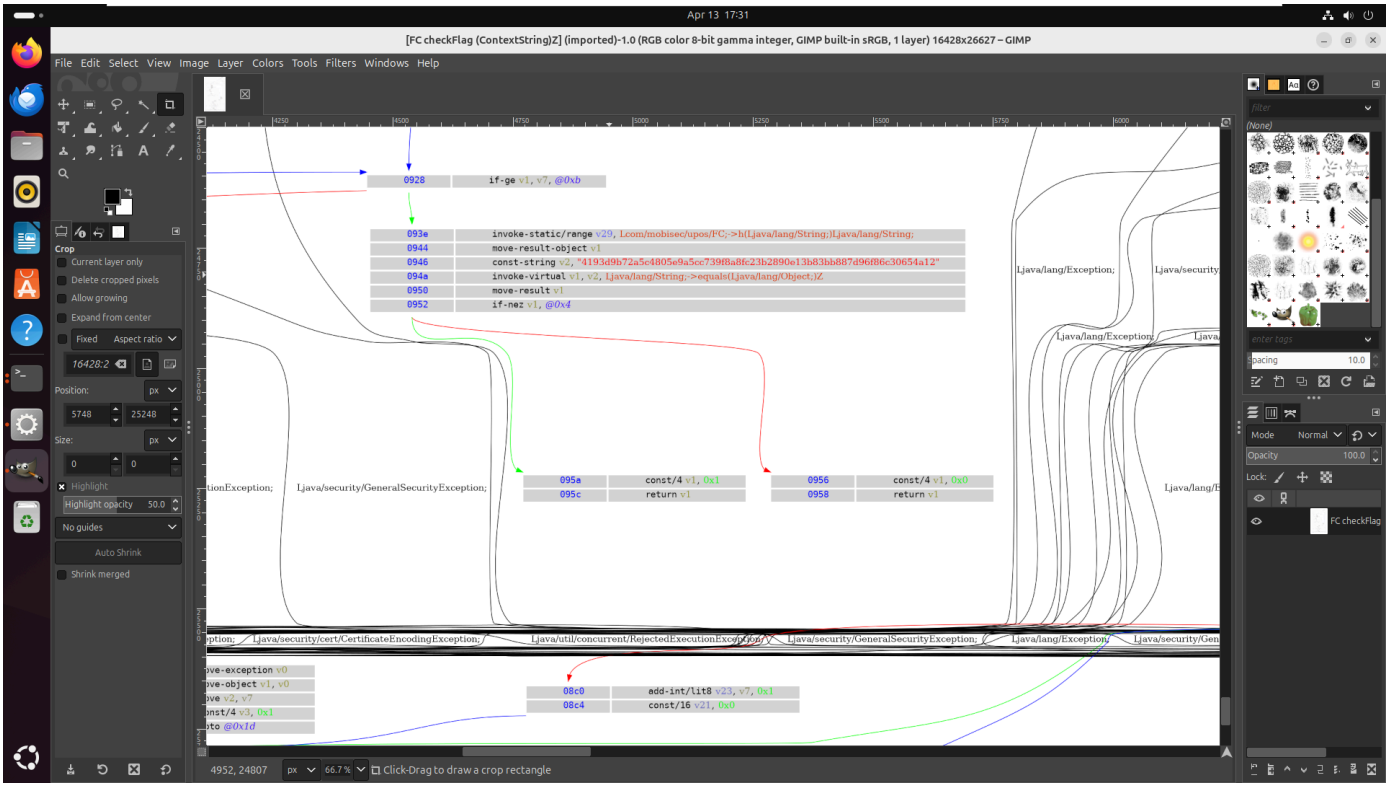
open the image

```
gimp FC\ checkFlag\ \(ContextString\)Z.png
```

create big FlowGraph for app



is writing with smali code



بس علشان نكمل حله لازم نكن فاهمين smali علشان كده وصلنا لحد و هنكمل حله لما نفهم smali language

44- challenge -Intro

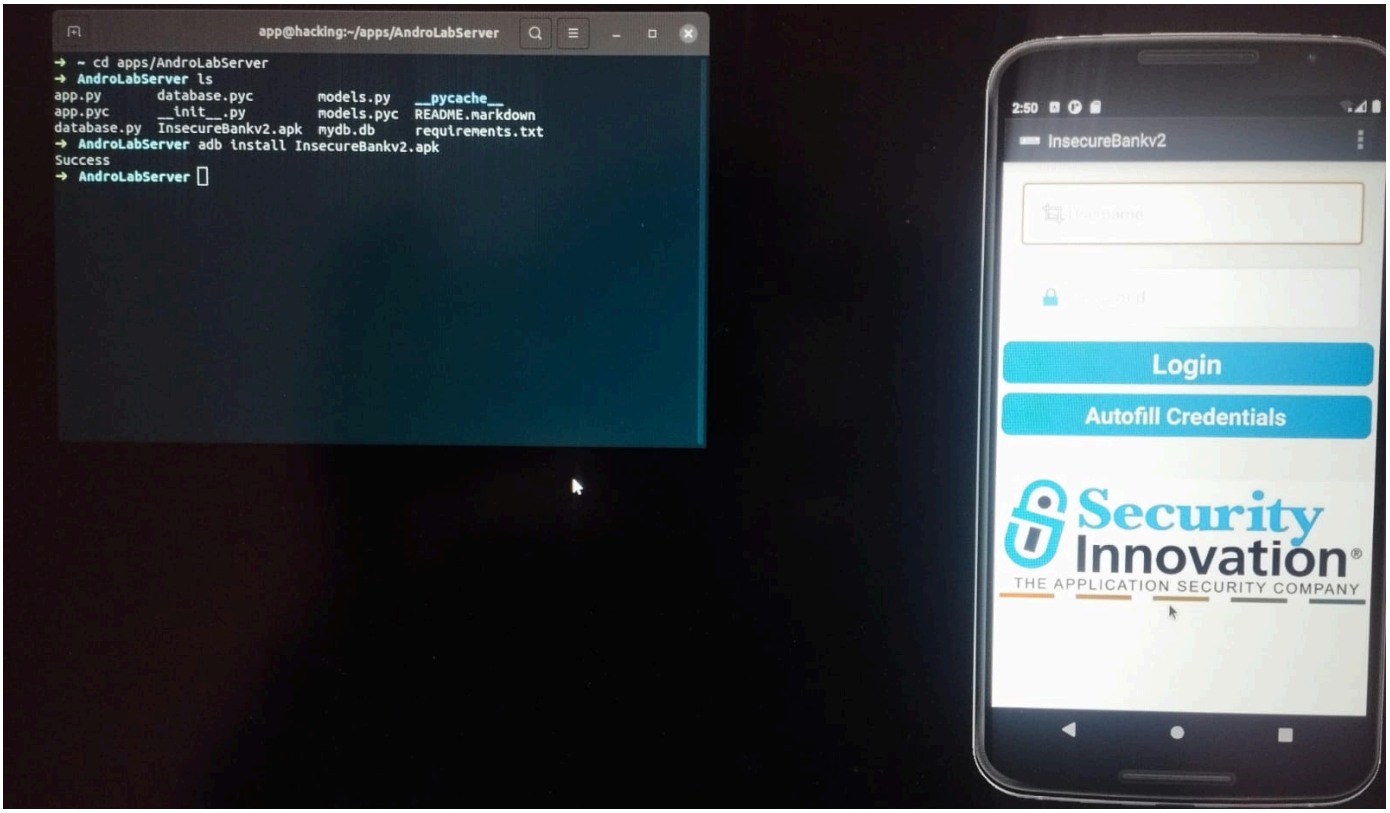
دلوقتي هنشوف challenge في كل حاجة من اول ما خدناه لحد هنا وهو عبارة عن apk app مربوط ب web server باستخدام python

```
→ my_udemy_challange cd AndroLabServer
→ AndroLabServer ls
app.py    database.py    __init__.py          models.py    mydb.db       README.markdown
app.pyc   database.pyc   InsecureBankv2.apk   models.pyc   __pycache__   requirements.txt
→ AndroLabServer
```

```
pip3 install -r requerment.txt
```

run the app.py

```
➜  AndroLabServer python3 app.py
The server is hosted on port: 8888
```

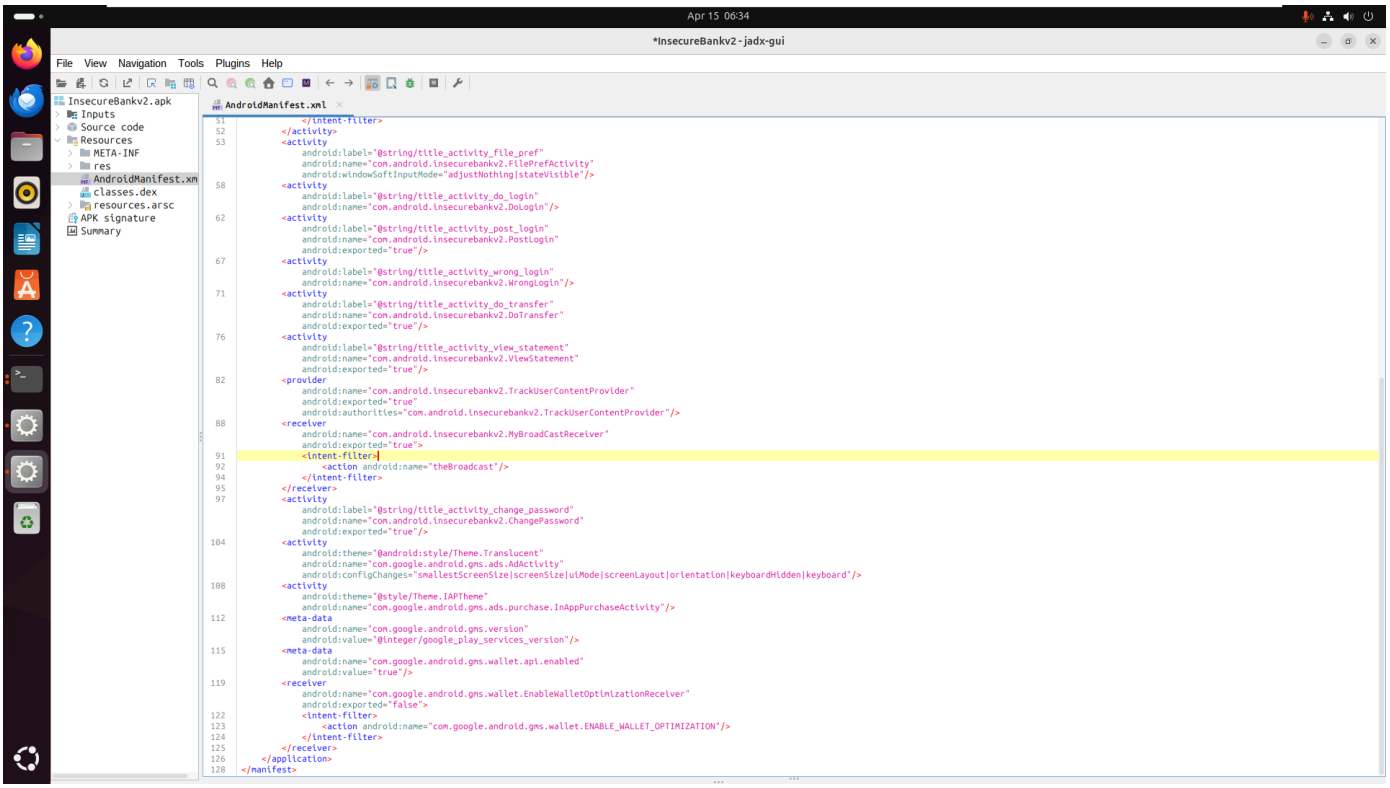


45- Hacking Activities

لو روحنا بقي استخدمنا jdax-gui علشان نحلل apk

```
jadx-gui InsecureBank.apk
```

ونشوف بقي ملف Manifast.xml



هلاقي ان في activities واهم اللي هو change-password علشان معمول "exported="true وده معناه ان احنا مثلا لو واقفين في login activity ممكن نخش علي change-password من غير ما نعمل logi

```
<activity
    android:label="@string/title_activity_change_password"
    android:name="com.android.insecurebankv2.ChangePassword"
    android:exported="true"/>
<activity
```

واسيطا في شوية activities مهمة معمول ليه exported

```
<activity
    android:label="@string/title_activity_post_login"
    android:name="com.android.insecurebankv2.PostLogin"
    android:exported="true"/>
<activity
    android:label="@string/title_activity_do_transfer"
    android:name="com.android.insecurebankv2.DoTransfer"
    android:exported="true"/>
<activity
    android:label="@string/title_activity_view_statement"
    android:name="com.android.insecurebankv2.ViewStatement"
    android:exported="true"/>
```

1-analyze code java for login activity

اهم الحاجات اللي لازم نعرفها

1- M7PREFS="mySharedPreferences"

contain username and encrypted password

هنا بيشرح ان البيانات بتكون محمية في دليل مخفي وعلشان تشوفها لازم تبقي معاك صلاحية الروت Root Privilage

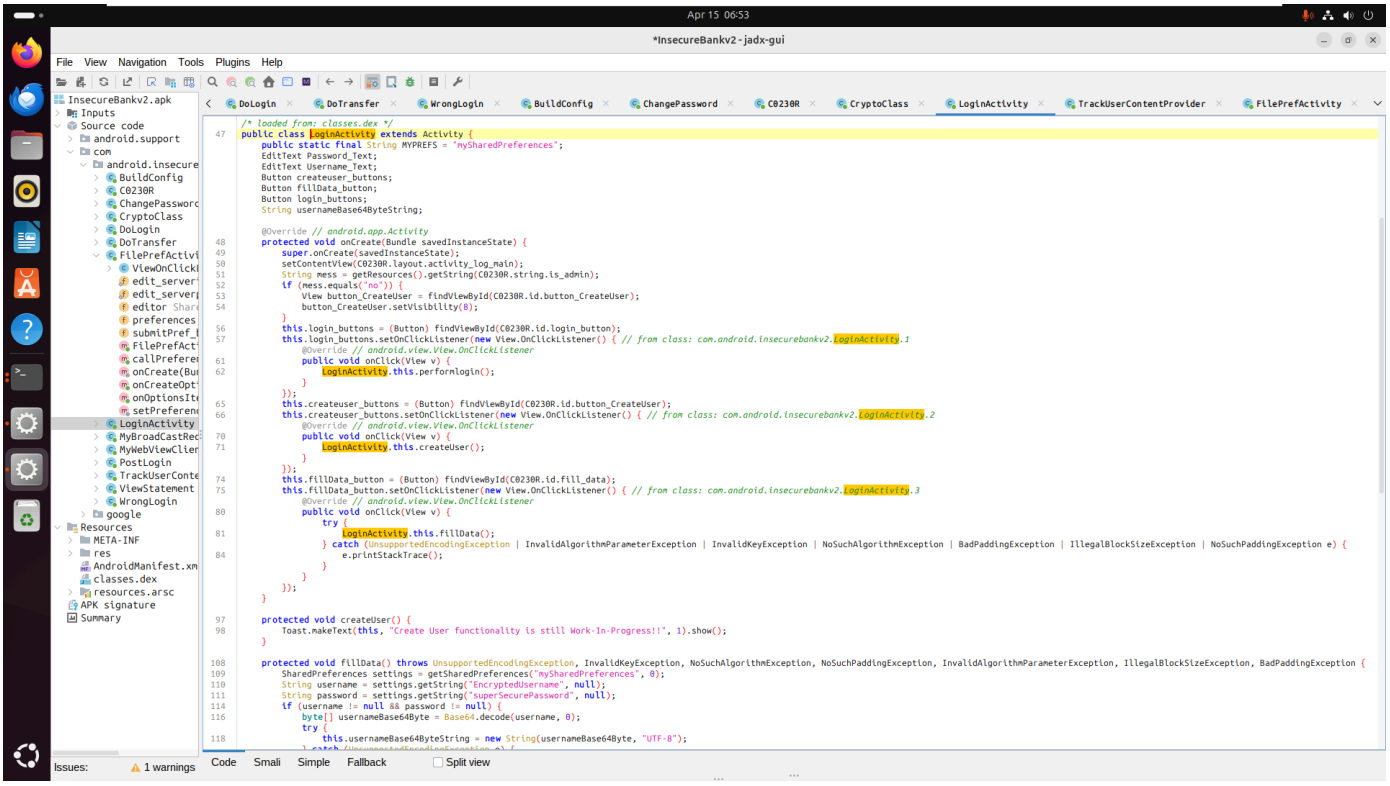
2- fillData() function

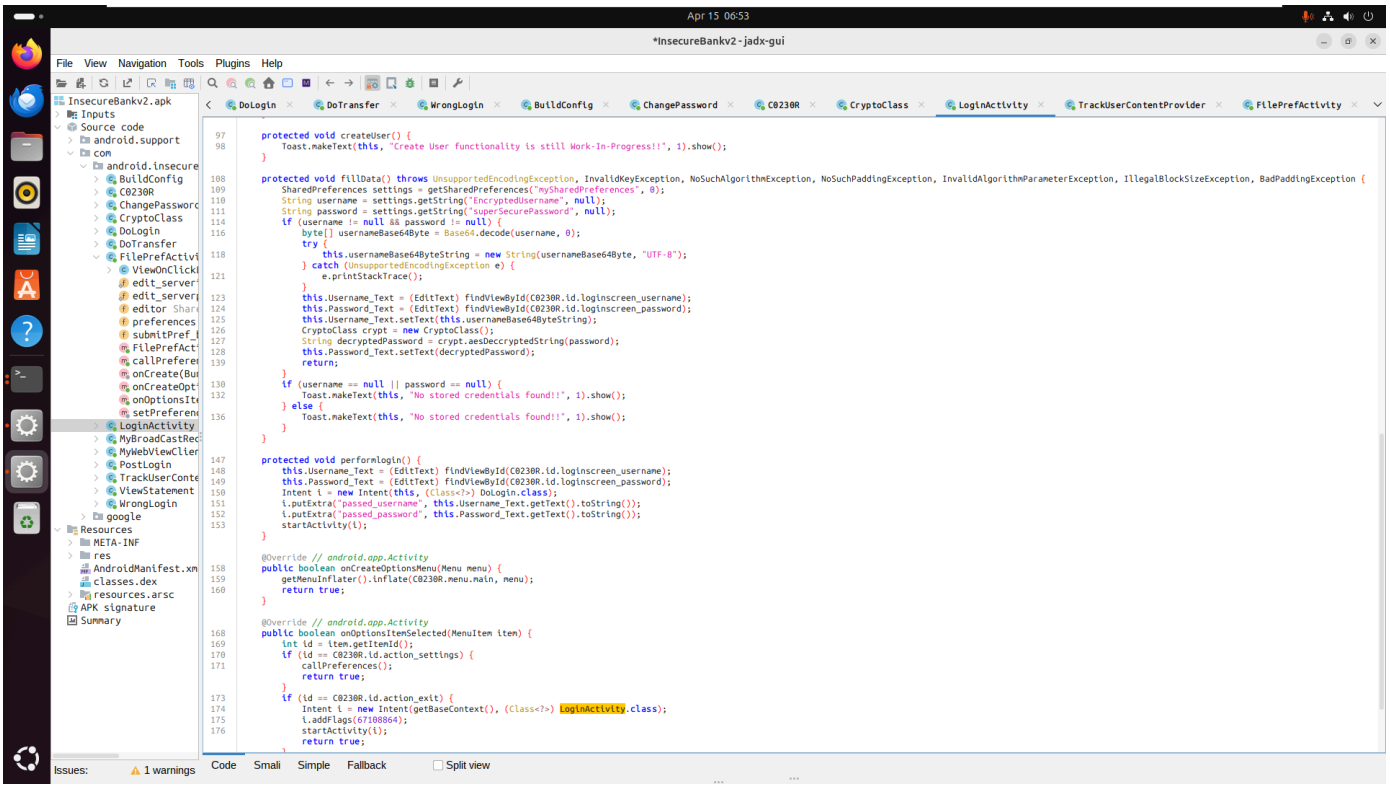
ودي تحتوي علي getstring("key",value) ودي بتروت تجيب من internel file قيمة key ولو مش موجودة بنحط null وي هنا كده :
هو هيروح يساوي username,password ب key الموجودة

username --> key is : EncryptedUsername , value : null

password --> key is : superSecurePassword , value : null

```
String username = settings.getString("EncryptedUsername", null);
String password = settings.getString("superSecurePassword", null);
```

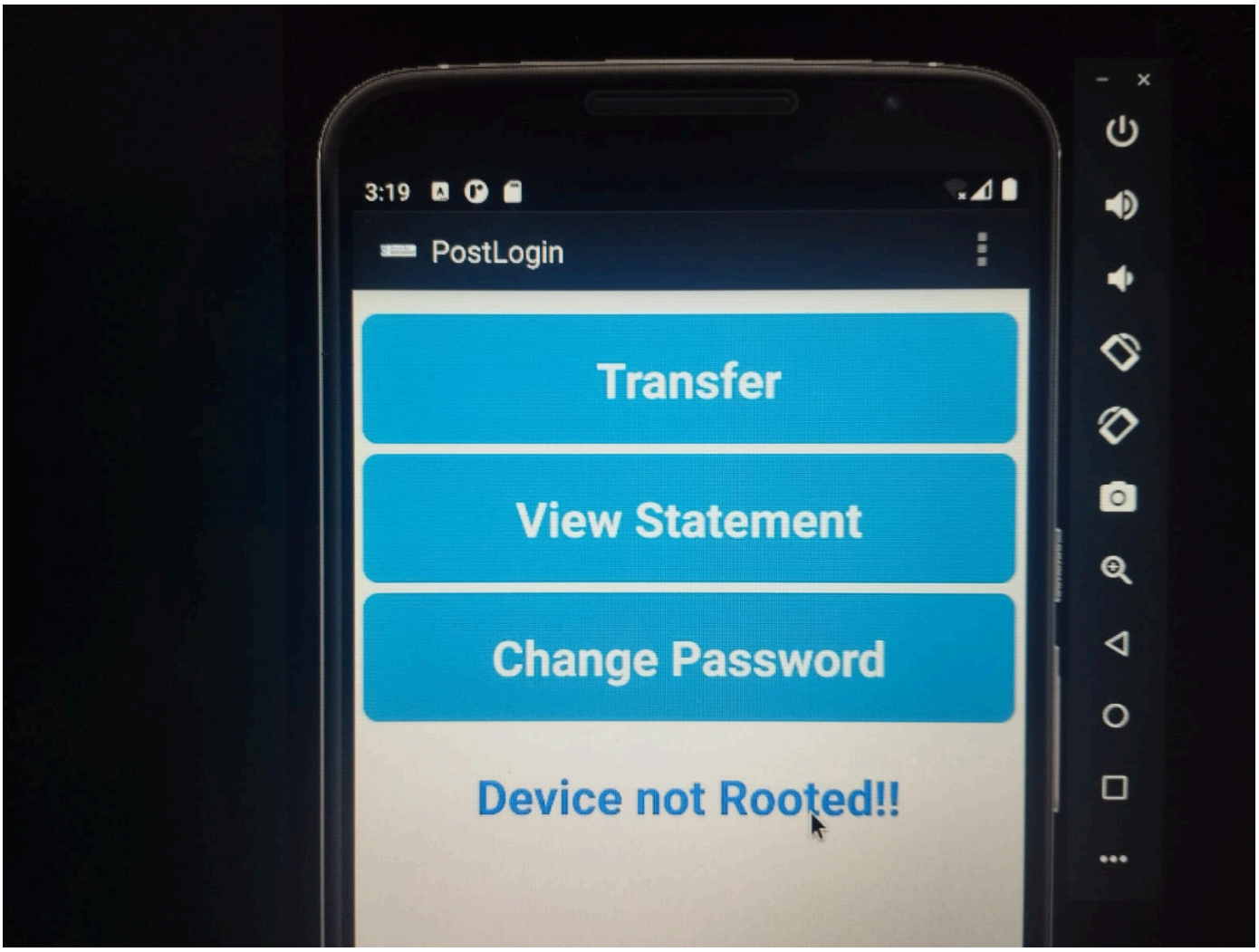




after know the activities are exported, how to start-activites

1- start activites for post-login

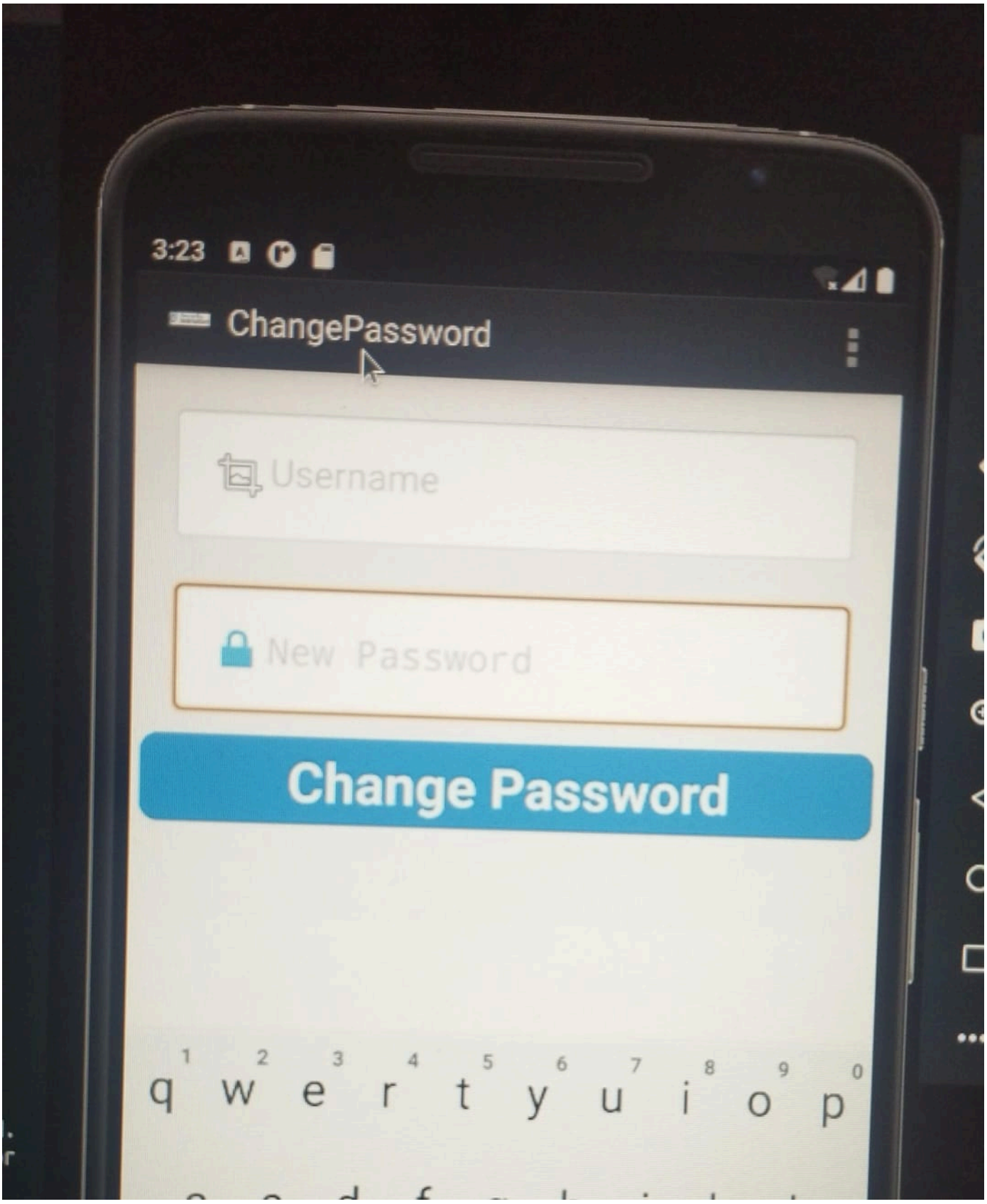
```
adb shell
am start-activity -n
com.android.insecurebankv2/com.android.insecurebankv2.PostLogin
```



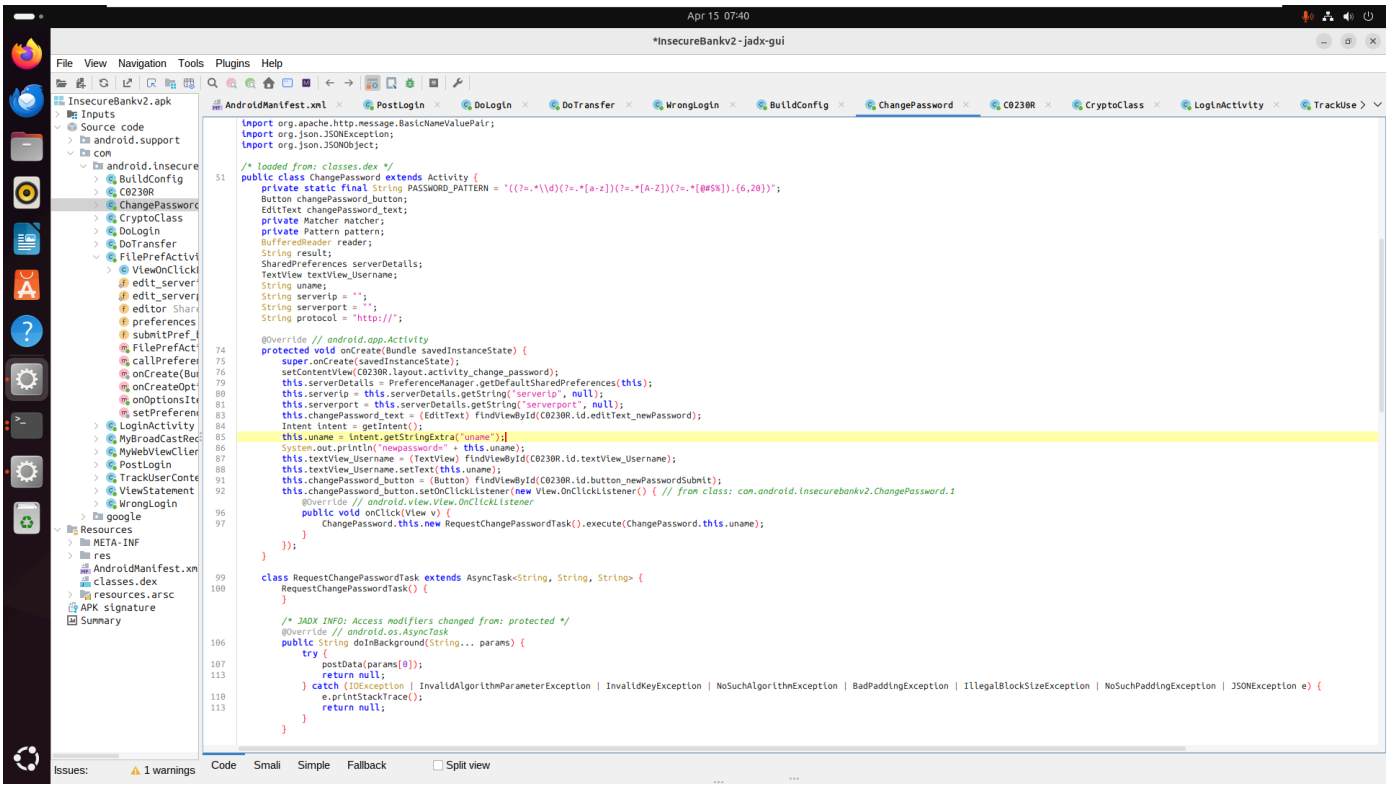
وهنا بيقول ان device not root فاحنا ممكن نبقي root باستخدام su

2- change password activites لو دخلنا بقي علي change password

هلناقي ان هو بيطلب new-password بس انا مش محدد username علشان اصلا احنا معملناش login فممكن نحدد username من adb shell

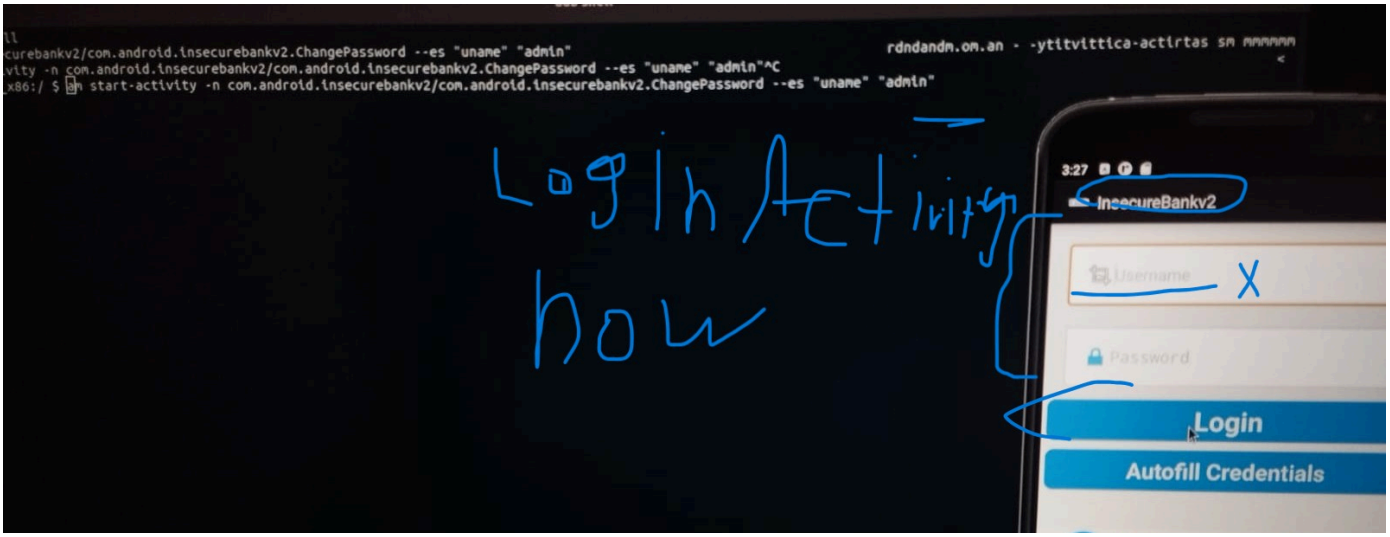


دلوقتي هنشوف chagne-password code هنلاقي ان هو في intent بيغير password لل username اللي بدخله وقولنا لو هو <intent> فاخنا ممكن نستخدمه علشان كده هنستخدم adb shell علشان نغير password لمستخدم احنا هنحدده

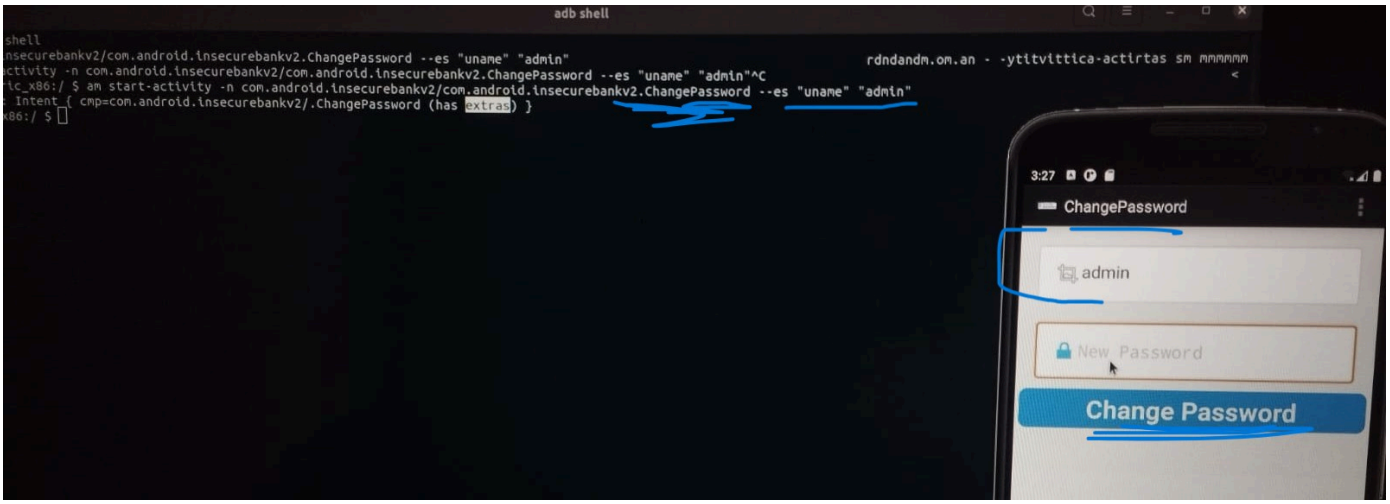



```
am start-activity -n com.android.insecurebankv2/com.android.insecurebankv2.ChangePassword --es "uname" "admin"
```

before start activity of change password for admin user



after start activity we access the change password of admin user



كده شرحنا ازاي من reverse enginerring قدرنا ان احنا نعمل hacking activities عن طريق ان احنا بنشوف الكود بتاع Manifest.xml وبنشوف activities اللي معمول لها exported or intent filter وبنحاول نستخدمها ان احنا نعمل bypass ل صفحة معينة او activities

47- Hacking Content Provider

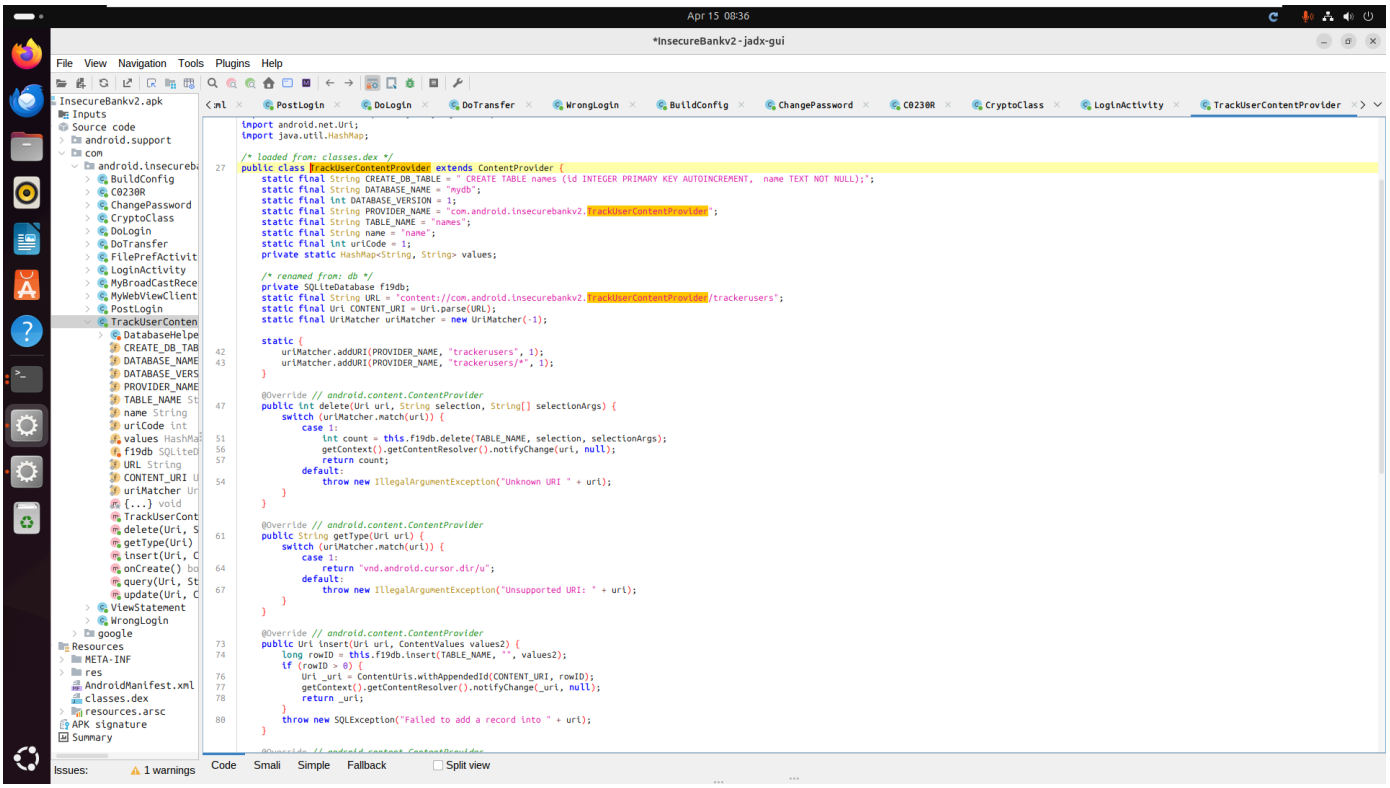
هنشرح هنا ازاي هنعمل Hacking Content Provider وده من خلال ان هتلاقي معمول له "exported ="true في الملف بتاع Manifest.xml وقولنا ان هو ملهوش <intent>فلو روحنا بقي نشوف الكود بتاع الجزء بتاع content في التبطيق في Manifest.xml

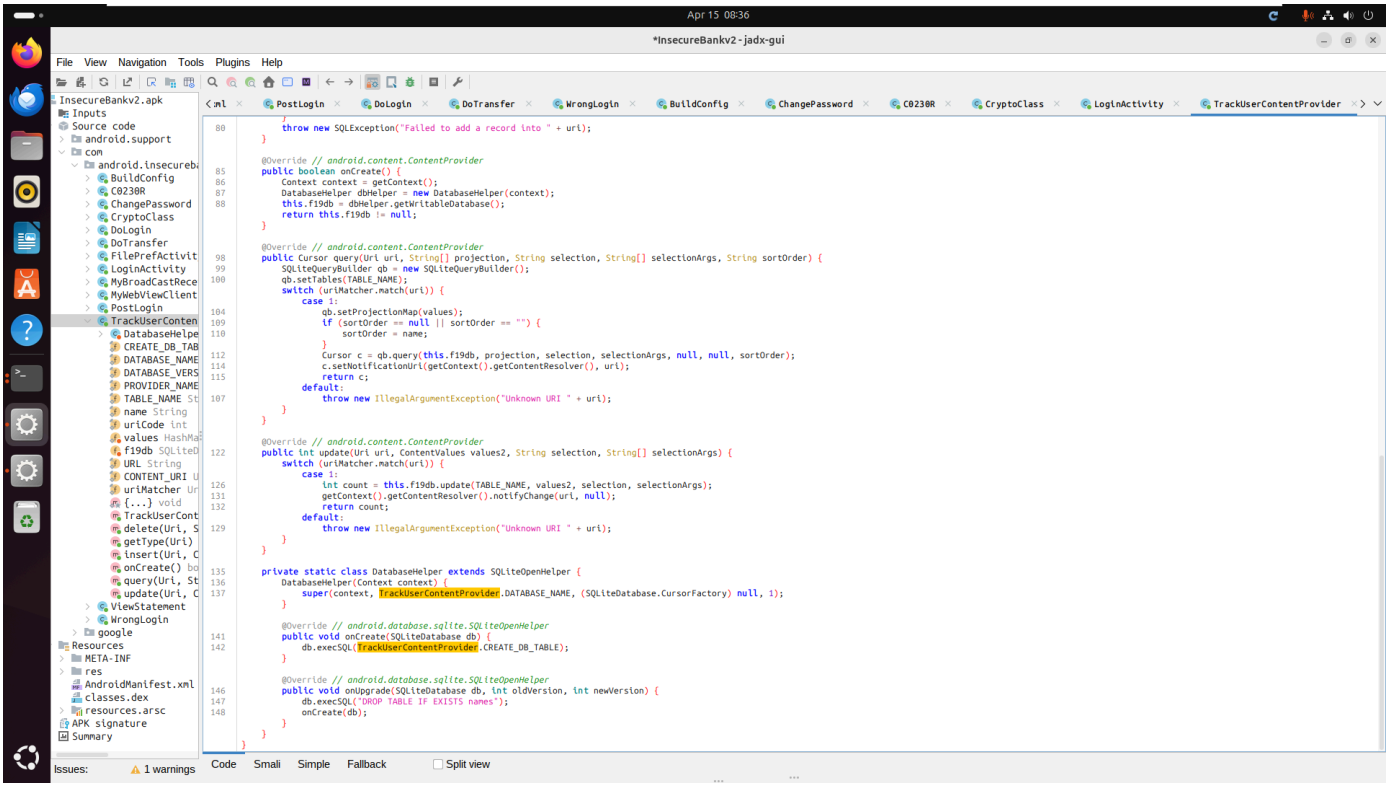
```
        <provider

android:name="com.android.insecurebankv2.TrackUserContentProvider"
            android:exported="true"

android:authorities="com.android.insecurebankv2.TrackUserContentProvider"/>
```

هنا اه في content provider معمول له "exported="true هنشوف بقي java code for TrackUserContentProvider

هنا اه database_name=mydb وان هو بيحتوي علي table_Name=names وده بيبقي في id, name

url -- > content://com.android.insecurebankv2.TrackUserContentProvider/trackerusers





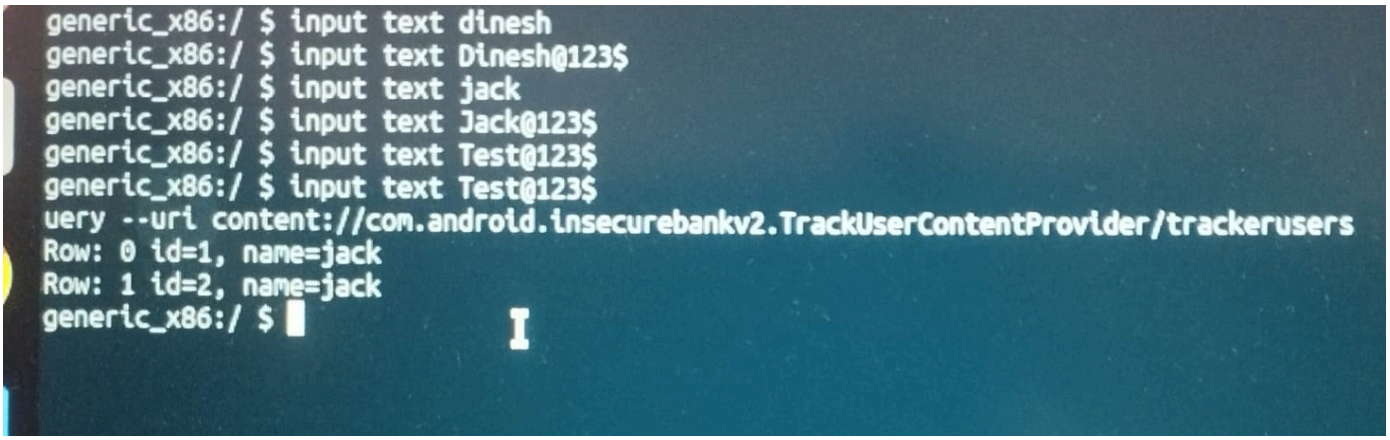
قوولنا بقي علشان نتعامل مع content هستخدم content (query | update | insert | delete | read | write)

اول حاجة بنعمل query فيها بنطلب بنشوف ايه اللي هيجلنا من خلاله بنشوف ايه users اللي موجودة وبنستخدم بقي users دي علشان نعمل start-activity change password for use got from query ه

هنا بقي هو ازاي بيجيب دول

```
select id,name from trackusers
```

يبقي كده id,name هما اللي هنعمل فيهم injectionللي هما projection بتوعي

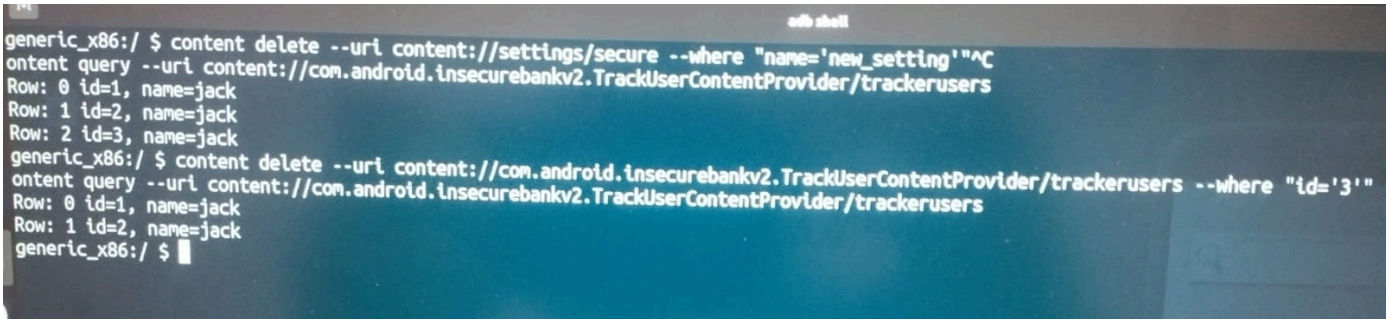


هنا اه علرفنا ان في user اسمه jack ممكن بقي نستخدم activity بتاع change password ونغيره password

```
am start-activity -n
com.android.insecurebankv2/com.android.insecurebankv2.ChangePassword --es
"uname" "jake"
```

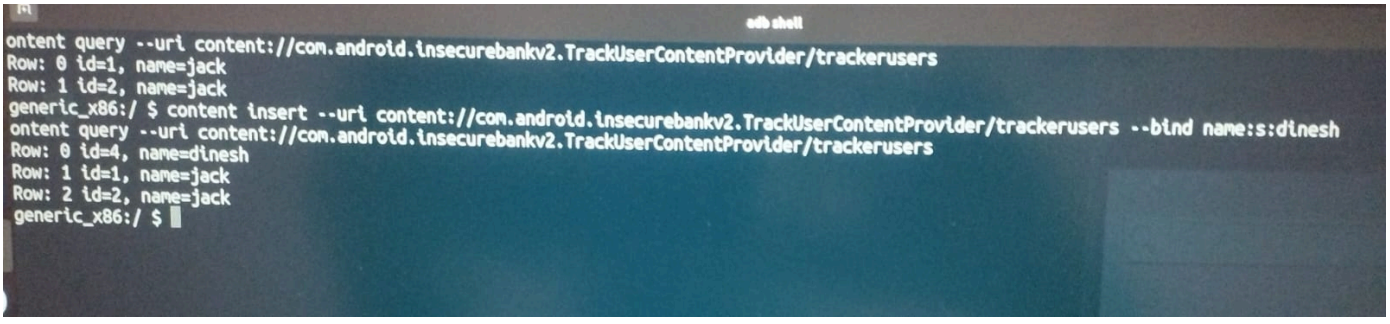
how to delete

```
content delete --uri content://....      --where "column_name='value'"
```



how to insert

```
content insert --uri content://...   --bind col_name:data_type:new_value
```



وبكده عرفنا ازاي نستغل coentet provider ان احنا نعمل query,update,insert او نتحكم في database اللي عندنا

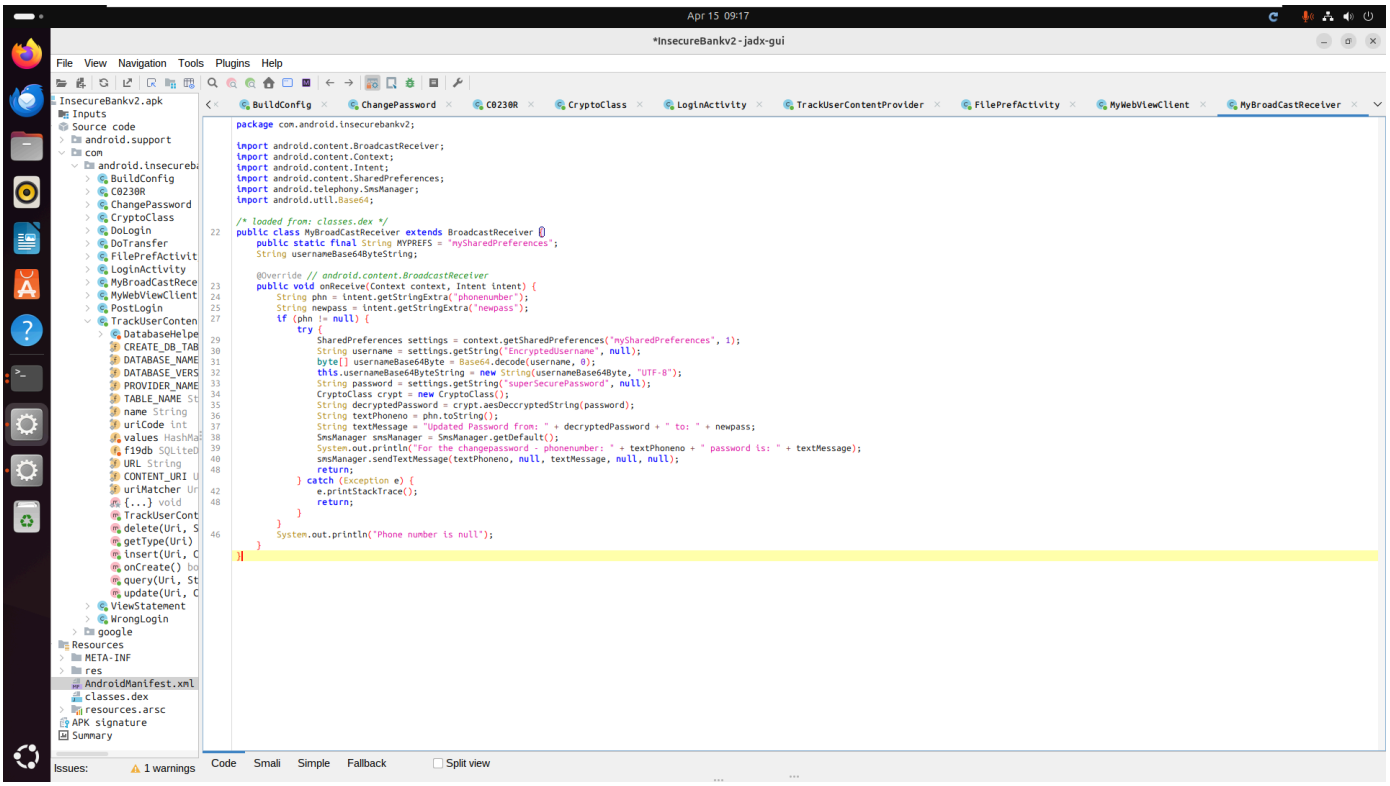
48-Hacking Broadcast Reciever

دلوقتي هنشوف ازاي هنعمل hacking broadcast receiver ودلوقتي لما نروح نشوف receiver في Manifest.xml ونشوف ايه reciever اللي معمول له "exported="true

هنا MyBroadCastReceiver معمول له exported

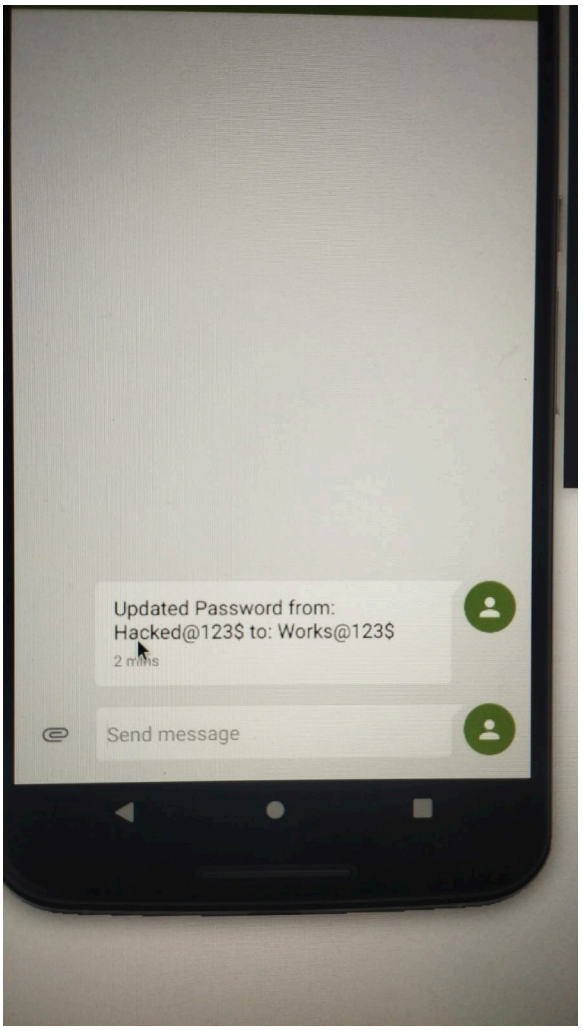
```
<receiver
            android:name="com.android.insecurebankv2.MyBroadCastReceiver"
            android:exported="true">
            <intent-filter>
                <action android:name="theBroadcast"/>
            </intent-filter>
        </receiver>
```

java code for MyBroadCastReceiver



هنا لما تعمل password هنا بتتبعت في logcat وخلي بالك هو مش هيعمل تحديث للباسورد القديمة هو بس هيعرض الباسورد القديمة والباسورد الجديدة في logcat والاستفادة منه هو اننا بنجيب الباسورد القديمة وبكده احنا عرفنا ايه هي password القديمة ونقدر نستخدمها عادي

```
am broadcast -n component_Name/broadcast_name -a "acton_name" --es
"phonenumber" "233232" --es "newpass" "new_password"
```



هنا اه هو بعتلي old password and new password

خلي بالك ان broadcast ممكن تحدد هو يتبعت لمين لان الجهاز هو اللي بيبعتوا للتطبيقات اللي عليه فلو محددناش هيتبعت لكله ولو حددنا هيتبعت للي محددين له فقط

```
      Send a broadcast Intent.  Options are:
      --user <USER_ID> | all | current: Specify which user to send to; if
not
          specified then send to all users.
```

49- Password Decryption

ازاي بقي نعمل decrypt للباسورد اللي في الملفات بتاعت التطبيق نفسه لما نعمل adb shell

any app under **/data/data/app-name**

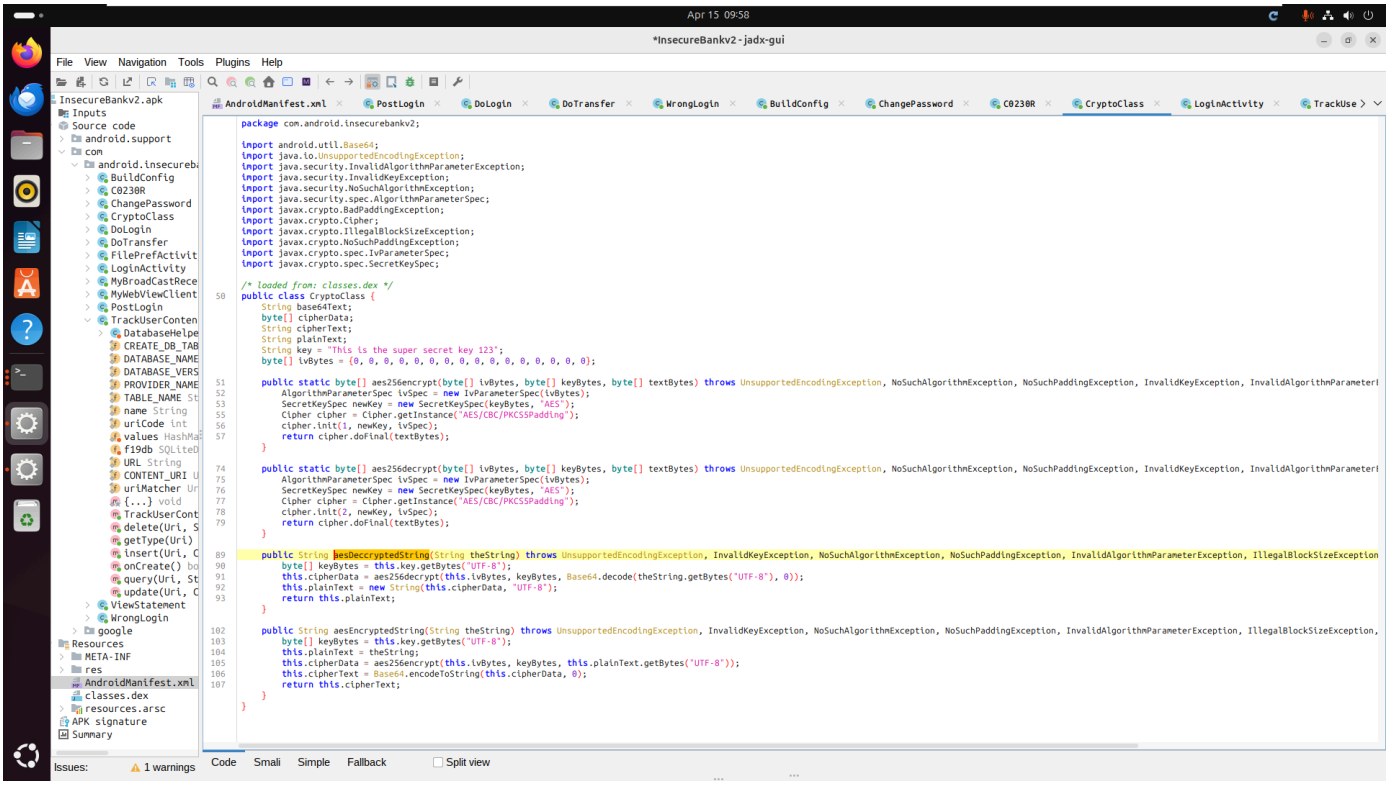
```
generic_x86:/data/data/com.android.insecurebankv2 # ls
app_textures  app_webview  cache  code_cache  databases  shared_prefs
generic_x86:/data/data/com.android.insecurebankv2 # cd
```

ولو دخلنا بقي علي الملف mySharedPreferences.xml اللي كان بيجيب منه الباسورد هنلاقي ان هو في password ولكن متشفرة base64

```
generic_x86:/data/data/com.android.insecurebankv2/shared_prefs # ls -lthsa
total 20K
131281 4.0K drwxrwx--x 2 u0_a121 u0_a121 4.0K 2022-02-06 16:22 .
131267 4.0K drwxr-x--x 8 u0_a121 u0_a121 4.0K 2022-02-06 15:22 ..
131287 4.0K -rw-rw---- 1 u0_a121 u0_a121  127 2022-02-06 15:22 WebViewChromiumPrefs.xml
131282 4.0K -rw-rw---- 1 u0_a121 u0_a121  160 2022-02-06 14:51 com.android.insecurebankv2_pr
.xml
131338 4.0K -rw-rw---- 1 u0_a121 u0_a121  221 2022-02-06 16:22 mySharedPreferences.xml
at mySharedPreferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="superSecurePassword">NZRGUJYLGZ+bBWqElpGdQ==&#10;    </string>
    <string name="EncryptedUsername">amFjaw==&#13;&#10;    </string>
</map>
generic_x86:/data/data/com.android.insecurebankv2/shared_prefs #
```

وهنا اه بقي الملف اللي في جزء تشفير الباسورد وهنا هو بيستخدم **symmetric encoding aes256**

وان هو جواه **"key_value="This is the super secert key 123"** وان **IV=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)**



base64 decoding

```
➜  Insecure_bank echo "NZRGUJYLGZ+bBWqElpGdQ==" |base64 -d |xxd -p
base64: invalid input
35944650960b199f9b056a8496919d
```

on CyberChif website decode it

set key on code and set IV is 16byte of 00