

初识APK文件结构，双开，汉化，基础修改

1. APK结构

apk 全称 Android Package，它相当于一个压缩文件，只要在电脑上将apk后缀改为zip即可解压。

文件	注释
assets目录	存放APK的静态资源文件，比如视频，音频，图片等
lib 目录	armeabi-v7a基本通用所有android设备，arm64-v8a只适用于64位的 android设备，x86常见用于android模拟器，其目录下的.so文件是c或 c++编译的动态链接库文件
META-INF目录	保存应用的签名信息，签名信息可以验证APK文件的完整性，相当于 APK的身份证(验证文件是否又被修改)
res目录	res目录存放资源文件，包括图片，字符串等等，APK的脸蛋由他的 layout文件设计
AndroidManifest.xml 文件	APK的应用清单信息，它描述了应用的名字，版本，权限，引用的库文件等等信息
classes.dex文件	classes.dex是java源码编译后生成的java字节码文件，APK运行的主要逻辑
resources.arsc文件	resources.arsc是编译后的二进制资源文件，它是一个映射表，映射着 资源和id，通过R文件中的id就可以找到对应的资源

▼ APK 文件是一个压缩文件，可通过解压工具对其进行解压缩以查看其内部结构，但解压缩后的文件不能直接运行，必须重新打包并签名后才能安装到设备上。

1. 文件结构

- **/META-INF文件夹**：包含应用程序的签名信息等元数据，如MANIFEST.MF（应用的清单文件）、CERT.SF（包含签名信息，用于校验应用包的完整性）、CERT.RSA（应用的数字证书，存储了证书信息）。
- **/res文件夹**：存储应用程序的各种资源文件，如界面布局（布局文件，如 `activity_main.xml`）、图形图像（放在\drawable-[配置]文件夹中，不同的屏幕密度有不同的目录）、字符串（`strings.xml`）、样式（`styles.xml`）、颜色（`colors.xml`）、动画（如 `animated.xml`）等。

- **/assets文件夹**：存放应用程序的任意资源，如配置文件、图片、视频等，这些文件会在编译时保留在原样，不会被处理。
- **/lib文件夹**：用于存储应用程序的原生代码库（**.so** 文件），针对不同的处理器架构有不同的文件夹（如 **/lib/armeabi-v7a/**、**/lib/arm64-v8a/**、**/lib/x86/** 等）。
- **/AndroidManifest.xml**：是应用程序的配置文件，声明了应用程序的组件（如活动、服务、广播接收器、内容提供者）、权限、图标、标签、版本等重要信息。
- **/classes.dex文件**：是应用程序的 Java 或 Kotlin 编译生成的字节码文件，包含应用程序的执行逻辑。
- **/resources.arsc文件**：是 Android 资源文件，包含了应用程序中的各种资源（如字符串、样式、颜色等）的索引信息。
- **其他文件**：如证书文件（如 **/META-INF/CERT.SF** 和 **/META-INF/CERT.RSA**）、应用图标、应用启动屏文件等。

2. APK的加载过程

当用户安装 APK 文件时，Android 系统会解析 APK 文件的结构，提取其中的 `AndroidManifest.xml` 文件和 `resources.arsc` 文件，识别出应用程序的组件、资源、权限等信息。之后，系统会将 `classes.dex` 文件中的字节码加载到内存中，并解释执行，完成应用程序的启动和运行。在运行过程中，系统会根据需要动态加载并使用其他资源文件（如 `res` 文件夹中的资源文件和 `assets` 文件夹中的文件）。

2. 双开及原理

[【VirtualAPP 双开系列08】如何实现多开 - UID virtualapp 使用-CSDN博客](#)

双开：简单来说，就是手机同时运行两个或多个相同的应用，例如同时运行两个微信

原理	解释
修改包名	让手机系统认为这是2个APP，这样的话就能生成2个数据存储路径，此时的多开就等于你打开了两个互不干扰的APP
修改Framework	对于有系统修改权限的厂商，可以修改Framework来实现双开的目的，例如：小米自带多开

通过虚拟化技术实现	虚拟Framework层、虚拟文件系统、模拟Android对组件的管理、虚拟应用进程管理 等一整套虚拟技术，将APK复制一份到虚拟空间中运行，例如：平行空间
以插件机制运行	利用反射替换，动态代理，hook了系统的大部分与system—server进程通讯的函数，以此作为“欺上瞒下”的目的，欺骗系统“以为”只有一个apk在运行，瞒过插件让其“认为”自己已经安装。例如：VirtualApp

(1) 通过MT管理器修改包名实现双开

1. 使用MT管理器提取应用的安装包



2. 找到我们要提取的apk，提取定位



提示

文件已保存到 /storage/emulated/0/
MT2/apks/CTF02_1.0.apk

定位

关闭

3.通过APK共存功能修改包名实现apk双开



APK签名

XML翻译模式

APK优化

XML批量替换

APK共存

RES资源混淆

RES资源精简

RES反资源混淆

去除签名校验

DEX字符串解密

数据复用优化

DEX混淆对抗

注入日志记录

DEX重新划分

注入文件提供器

APK共存

新包名

com.example.test.ctf03

使用旧版共存方案



自动签名

取消

确定

4. 实现双开



(2) 通过NP管理器修改包名实现双开

流程与MT大致相同

工具



捐赠鼓励



安装包提取



工具箱

用户应用

系统应用



CTF02

1.0

包名

com.example.test.ctf02

版本号

1

安装包大小

1.67M

签名状态

V1 + V2

加固状态

未加固

数据目录

/data/user/0/com.example...

APK路径

/data/app/com.example.te...

首次安装时间

2025-02-25 19:58:11

上次更新时间

2025-02-25 19:58:11

UID

10054

更多

提取安装包



DEX混淆字典提取

DEX混淆对抗

替换DEX包名/类名

一键拆分DEX

一键添加崩溃日志记录

一键合并DEX

一键随机签名APK

普通一键功能

查看签名

添加签名校验

APK共存

超级混淆 4.0

APK签名

APK混淆

控制流混淆 8.0

APK VM保护

DEX字符串解密

RES资源混淆

测试签校强度

RES反资源混淆

APK伪加密

反APK伪加密

通用编辑

APK DEX2C

APK共存

新包名:

com.example.test.ctf03

开启随机签名



注意: Apk共存仅限于在允许二次修改的
Apk文件上使用。

取消 确定

打开修改报名后的apk安装



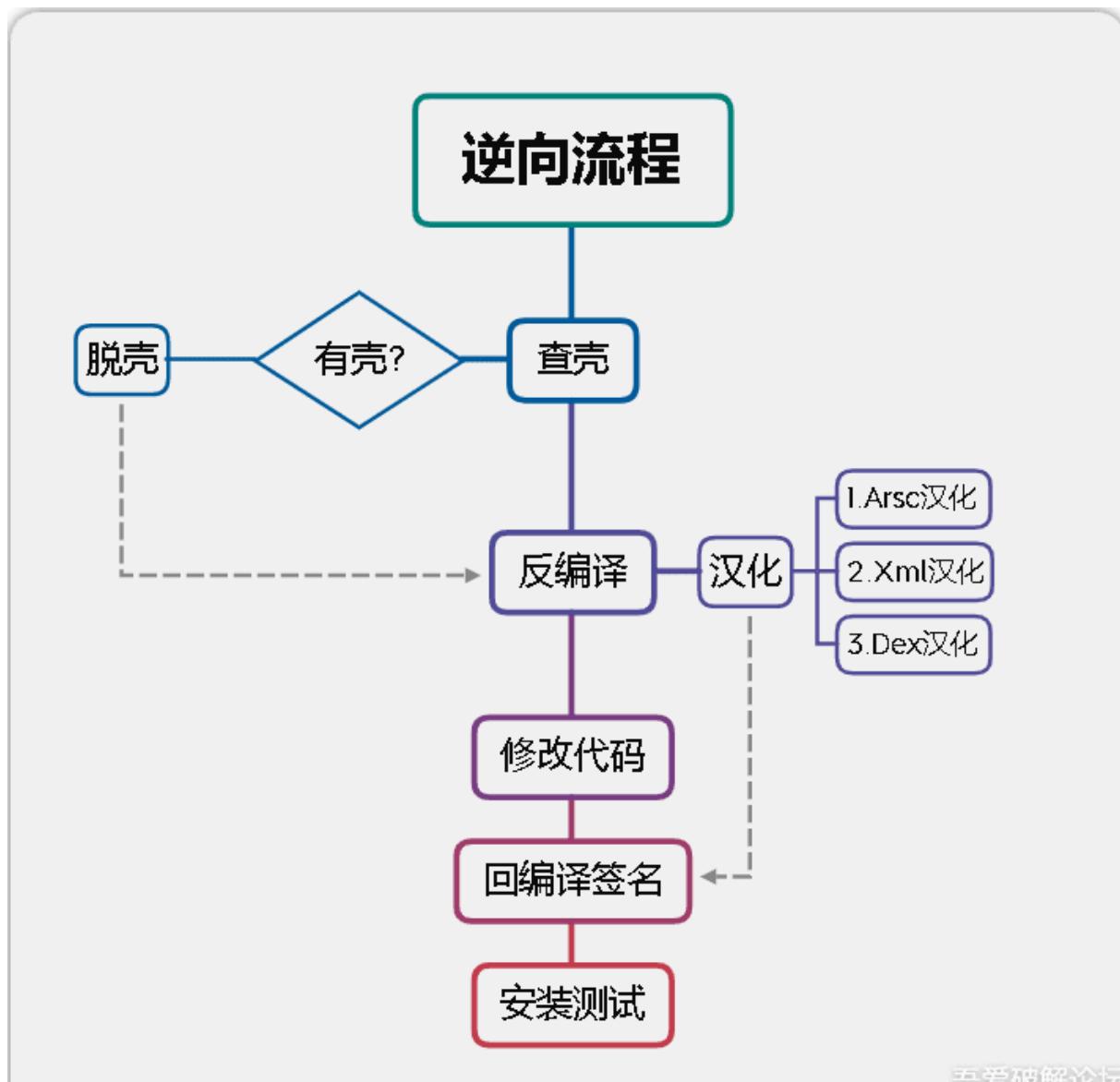
就可以在手机上完成双开。因为修改报名实现双开动了应用的签名信息，如果开发者做了签名校验可能会出现安装后闪退崩溃等情况。

3. 汉化APK

使用专门的工具对外文版的软件资源进行读取，翻译，修改，回写等一系列处理，使用软件的菜单，对话框，提示等用户界面显示为中文，而程序的内核和功能保持不变，这个过程即为软件汉化。

（需要注意，如果要直接装应用就应该先签名安装，看石否有签名校验导致的闪退。）

基本上字符串都是在arsc里，建议一键汉化，然后再润色。少量没汉化到的字符串参考视频中的方法定位去逐个汉化。



(1) 简单的英语字句修改

- ▼ 1.我们在管理器中点击查看



▼ 2.利用MT自带的搜索功能



▼ 3.点击高级搜索相当使用全局搜索（搜索应用中的全部文件）

搜索

欲搜索文件名 (支持通配符*和?)



搜索子目录



高级搜索

按文件大小过滤 无限制 - 无限制

文件中包含内容

password



区分大小写



正则表达式

取消

确定

▼ 4. 对查询到的结果进行反编译

搜索结果(3)



resources.arsc

79-11-30 00:00 203.03K

/



activity_main_1.xml

79-11-30 00:00 1.15K

/res/layout/



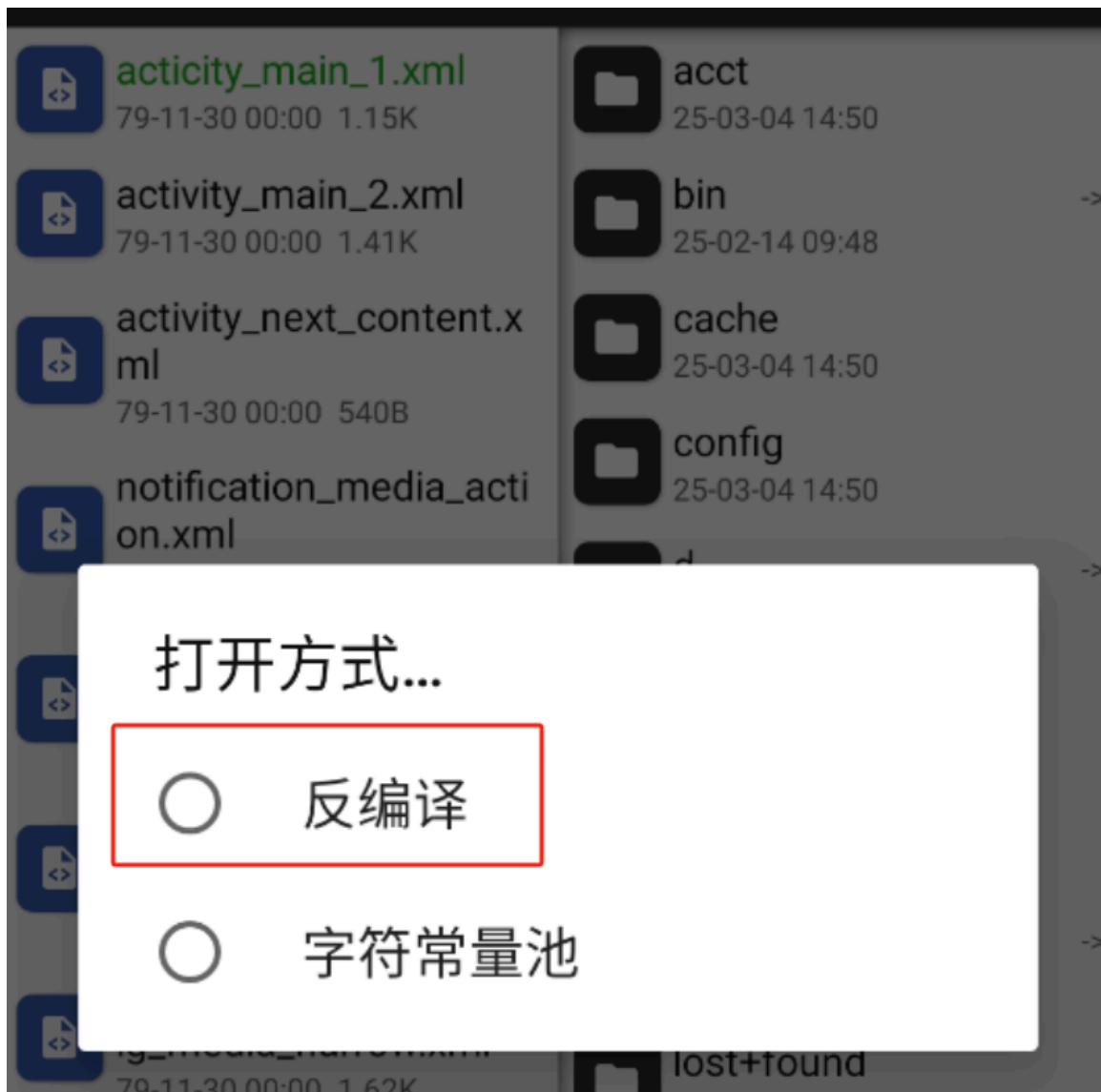
classes.dex

79-11-30 00:00 2.49M

/

清除

关闭



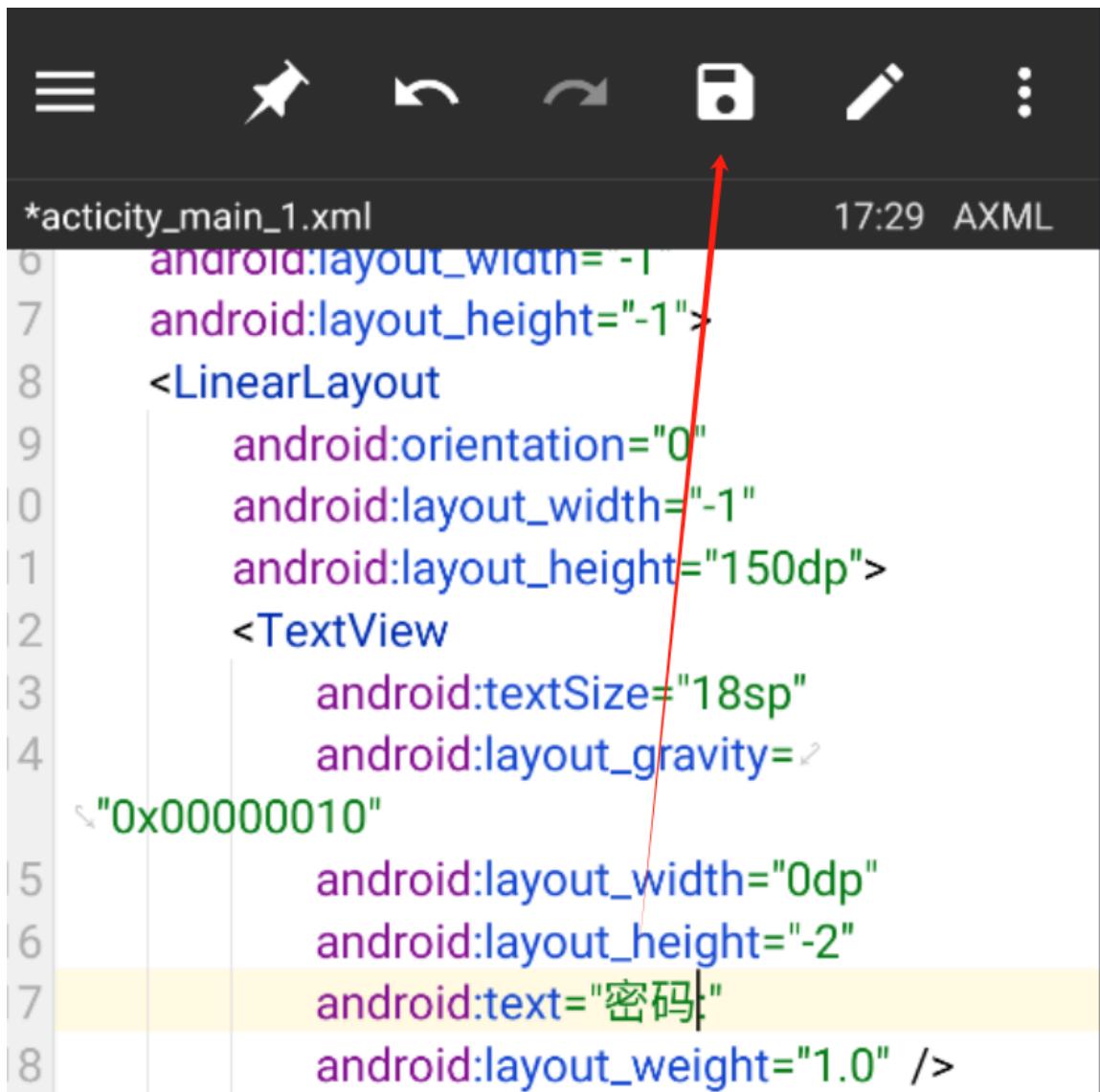
▼ 5.对内容进行修改后保

The screenshot shows an Android XML editor interface with the following details:

- Toolbar icons: three horizontal lines, a star, a refresh, a circular arrow, a save, a pencil, and a more options menu.
- File name: `activity_main_1.xml`
- Time: `17:35`
- File type: `AXML`
- Code content:

```
6     android:layout_width="-1"
7     android:layout_height="-1">
8     <LinearLayout
9         android:orientation="0"
10        android:layout_width="-1"
11        android:layout_height="150dp">
12        <TextView
13            android:textSize="18sp"
14            android:layout_gravity=<2
15            "0x00000010"
16            android:layout_width="0dp"
17            android:layout_height="-2"
18            android:text="PassWord:" />
19        <EditText
```

The line `android:text="PassWord:"` is highlighted with a red rectangle.



A screenshot of an Android XML editor showing the file `*acticity_main_1.xml`. The code displays an XML structure for a layout. A red arrow points from the top right towards the text "密码" in line 7. The XML code is as follows:

```
0     android:layout_width="-1"
1     android:layout_height="-1">
2     <LinearLayout
3         android:orientation="0"
4         android:layout_width="-1"
5         android:layout_height="150dp">
6             <TextView
7                 android:textSize="18sp"
8                 android:layout_gravity="2
9                 <"0x00000010"
10                android:layout_width="0dp"
11                android:layout_height="-2"
12                android:text="密码"
13                android:layout_weight="1.0" />
```

- ▼ 6.发现我们对内容进行了重新签名，应用与最开始安装已经是不一样的了。



- ▼ 7.对于卸载提示可以直接关闭处理，在设置里的APK安装检验证关闭就不会再提示



← 设置

显示删除警告

删除文件时如果未选择移动到回收站，将显示红色警告，可一定程度上避免您误删文件



安装

APK 安装验证

安装 APK 前验证签名和版本号



APK 安装防自动删除

防止 APK 文件在安装后被系统自动删除，在极少部分系统上会导致安装器异常，如有遇到请关闭该功能



安装 APK 前二次确认

在使用 Shizuku 或 Root 安装 APK 前需二次确认



使用 Shizuku 安装 apk/apks/xapk

重要安装并激活 Shizuku



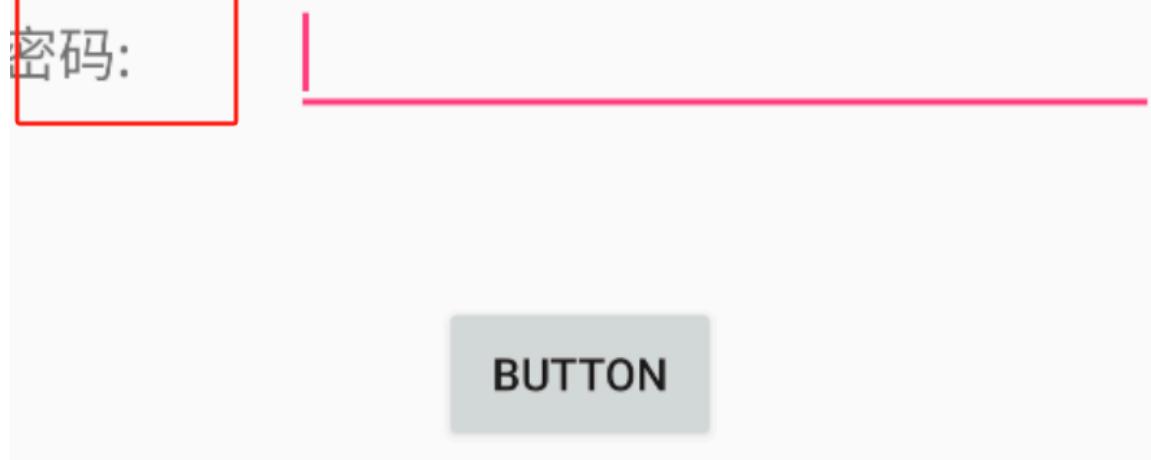
- ▼ 8.关闭后还会出现签名不一致安装失败的现象。(可以卸载原应用或利用XP的模块核心破解来解决这类问题)





▼ 9.完成后我们会发现修改部分已经变为中文模式

CTF02



(2) 利用开发者助手搜索

- ▼ 1. 下载安装后会有一个小图标



- ▼ 2. 我们到需要修改的app页面点击图标，点击开始资源分析



▼ 3.点击需要查看的亮条部分，点击复制我们需要的文字信息，再回到MT管理器中重复上述步骤即可

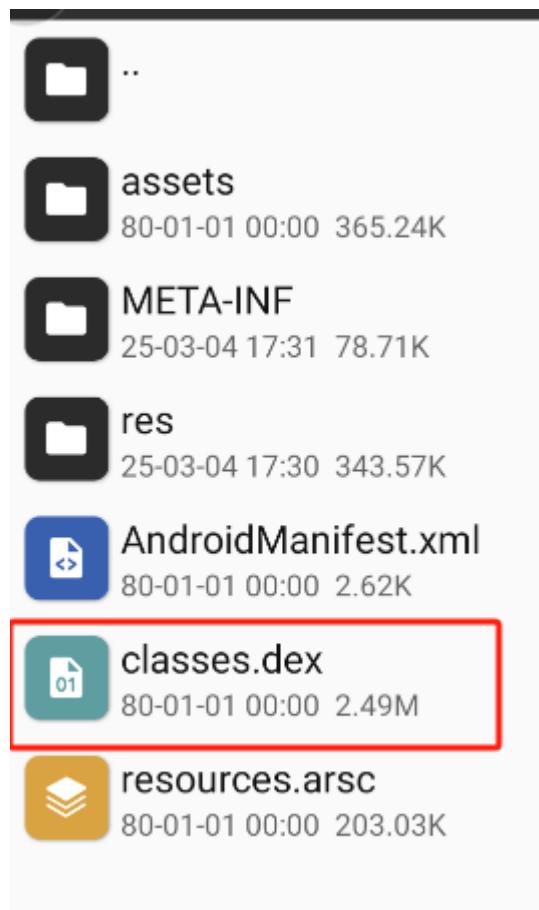
控件属性	
Package	com.example.test.ctf02
Class	android.widget.Button
Id-Name	R.id.button
Id-Dec	2131427414
Id-Hex	0x7F0B0056
Text	BUTTON
Content-Desc	
Bounds	[408,618][672,762]
Selected	false
Checkable	false
Checked	false
Clickable	true
Enable	true
Focusable	true
Focused	false
Scrollable	false

也可以直接再MT浏览器中开通汉化翻译插件

我们发现第二次签名后软件可以直接安装成功，原因是MT自带的签名信息一致，不会出现签名信息冲突的情况，可以直接覆盖安装

(3) 放在dex文件中

▼ 1.选择dex编辑器++





▼ 2.同样来搜索我们修改的字符，搜索类型可以自行选择。





▼ 3.很快就能的得到搜索结果

浏览 最近 搜索 常量

功能

发起新搜索 在当前结果中搜索 在当前结果中替换 清除搜索

搜索结果(54) - button

- **android.support.v4.media.session**
 - C MediaButtonReceiver**
 - .source "MediaButtonReceiver.java"
 - ...ic final TAG:Ljava/lang/String; = "MediaButtonReceiver"
 - const-string v1, "MediaButtonReceiver"
 - const-string v2, "A unique media button receiver could no..."
 - const-string v3, "MediaButtonReceiver"
 - ...string v4, "The component name of media button receiv..."
 - const-string v5, "Cannot build a media button pending int..."

▼ 4.修改后保存退出重复上述步骤

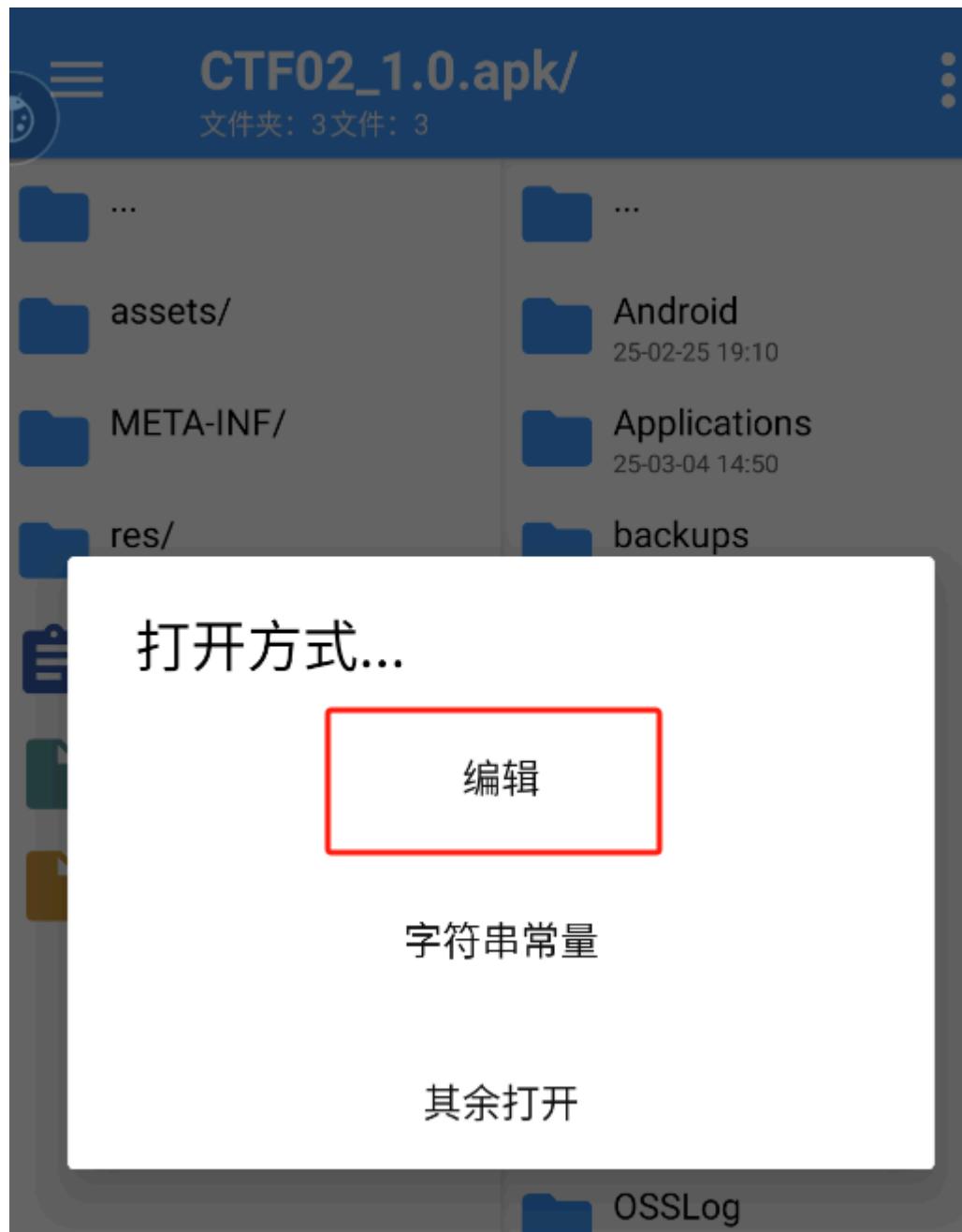
4.初识AndroidManifest.xml

AndroidManifest.xml文件是整个应用程序的信息描述文件，定义了应用程序中包含的Activity,Service,Content provider和BroadcastReceiver组件信息。每个应用程序在根目录下必须包含一个AndroidManifest.xml文件，且文件名不能修改。它描述了package中暴露的组件，他们各自的实现类，各种能被处理的数据和启动位置。

属性	定义
versionCode	版本号，主要用来更新，例如:12
versionName	版本名，给用户看的，例如:1.2
package	包名，例如：com.zj.52pj.demo
uses-permission android:name=""	应用权限，例如：android.permission.INTERNET 代表网络权限
android:label="@string/app_name"	应用名称
android:icon="@mipmap/ic_launcher"	应用图标路径
android:debuggable="true"	应用是否开启debug权限

使用NT浏览器修改AndroidManifest.xml文件

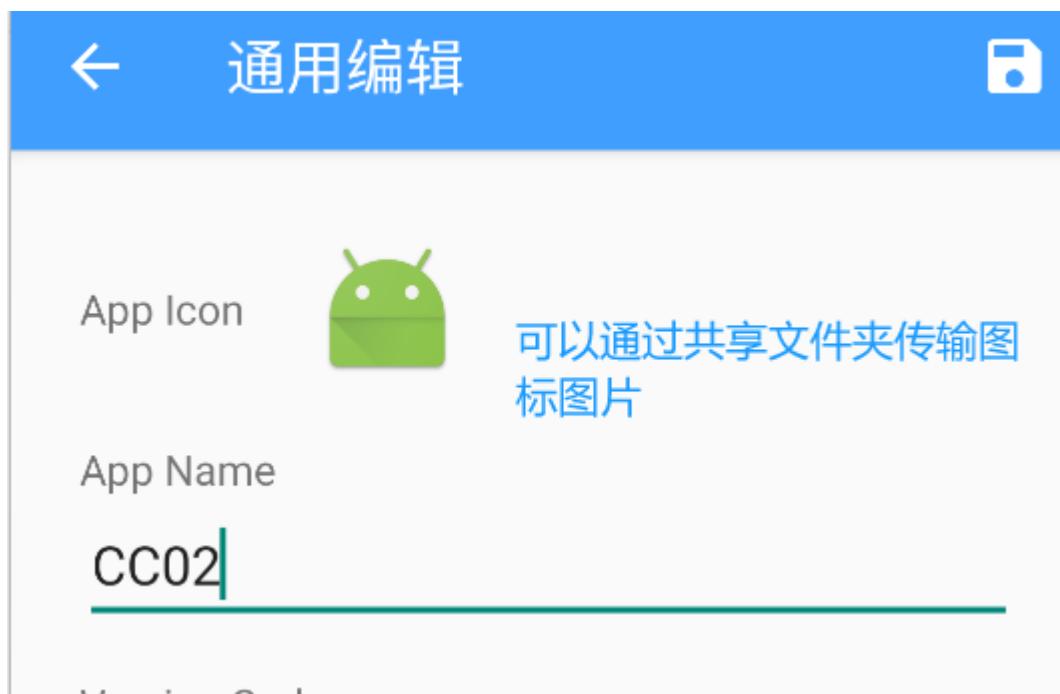
- ▼ 1.直接进入查看编译



```
6 android:compileSdkVersionCodename="12"
7 package="com.zj.wuaipojie"
8 platformBuildVersionCode="32"
9 platformBuildVersionName="12">
0 <uses-sdk android:minSdkVersion="27" android:targetSdkVersion="3
1 <!-- 拥有完全的网络访问权限 -->
2 <uses-permission android:name="android.permission.INTERNET"/>
3 <!-- 修改或删除您共享存储空间中的内容 -->
4 <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
5 <application
6     android:theme="@style/Theme_Wuaipojie"
```

▼ 2.修改apk图标和名字，在NP管理器中使用通用编辑进行修改





▼ 3. 保存安装发现又出现签名信息不一致因为上边用的MT管理器签名，和现在NP签名不一致，按上述方法解决即可，安装后可以观察到修改成功。

