安卓逆向学习

一、模拟器配置

模拟器：雷电模拟器9

1、开启root权限

2、系统磁盘设置为可写入

面具magisk：https://github.com/HuskyDG/magisk-files/releases

1、安装面具magisk

安装面具后选择magisk，安装到系统分区

安装后重启

选择设置，开启

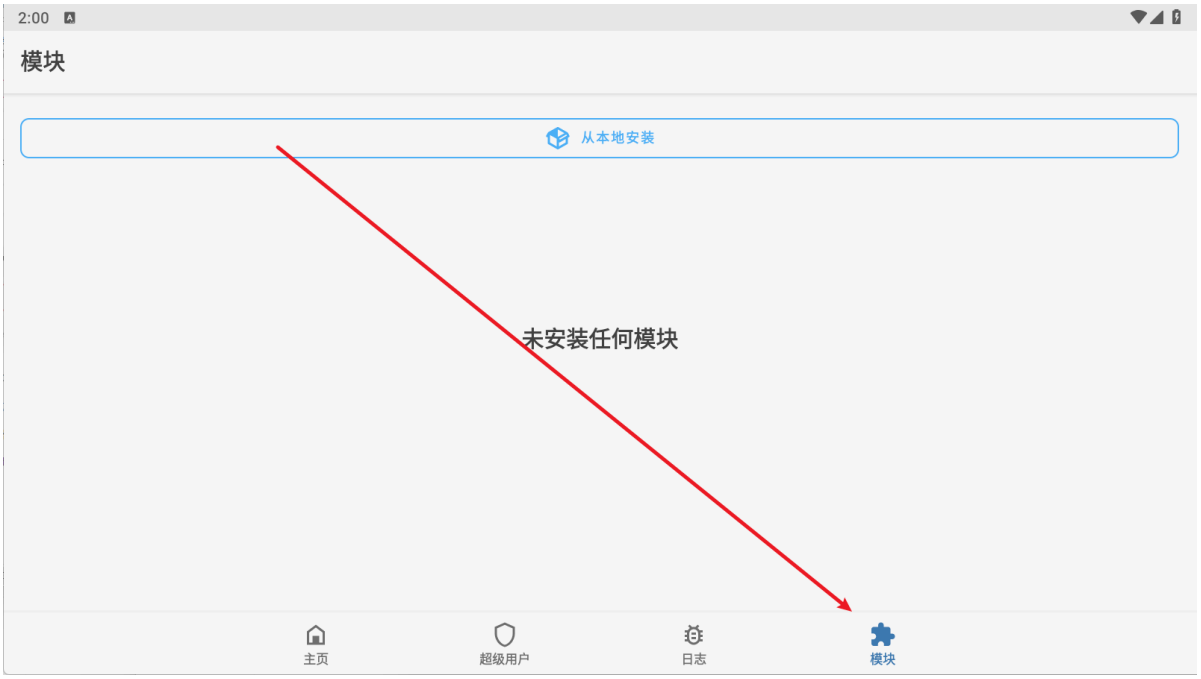
输入LSPosed模块

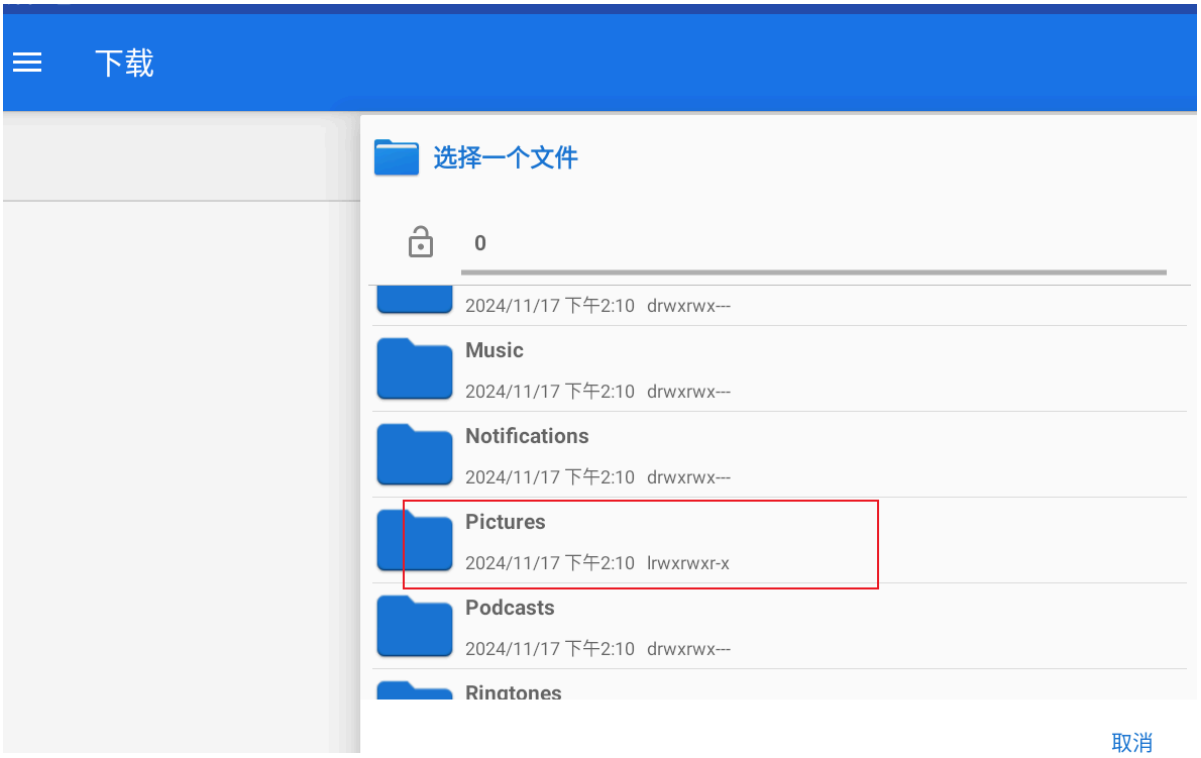
2、安装LSPosed模块

工具地址:https://github.com/LSPosed/LSPosed/releases

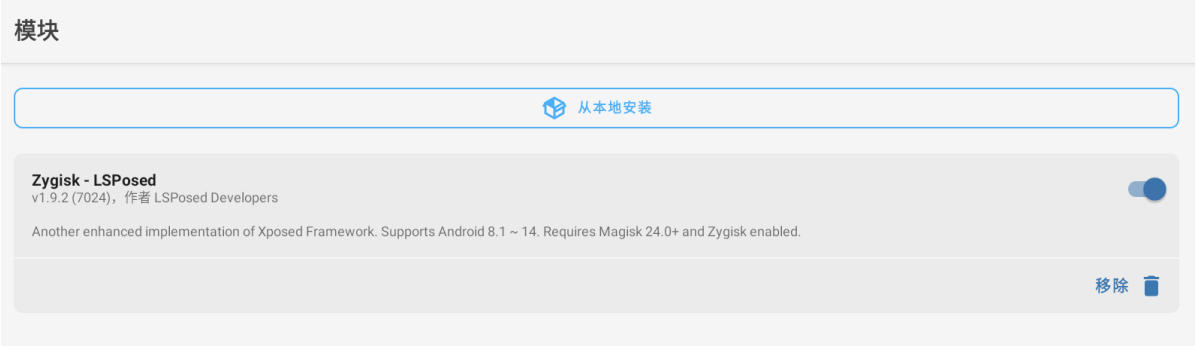
移动到模拟器中，设置-更多-共享文件夹，就文件复制到文件中

选择面具-模块安装


2:00
模块
从本地安装
未安装任何模块
主页
超级用户
日志
模块


下载
选择一个文件
0
2024/11/17 下午2:10 drwxrwx---
Music
2024/11/17 下午2:10 drwxrwx---
Notifications
2024/11/17 下午2:10 drwxrwx---
Pictures
2024/11/17 下午2:10 lrwxrwxr-x
Podcasts
2024/11/17 下午2:10 drwxrwx---
Ringtones
取消

重启



配置成功

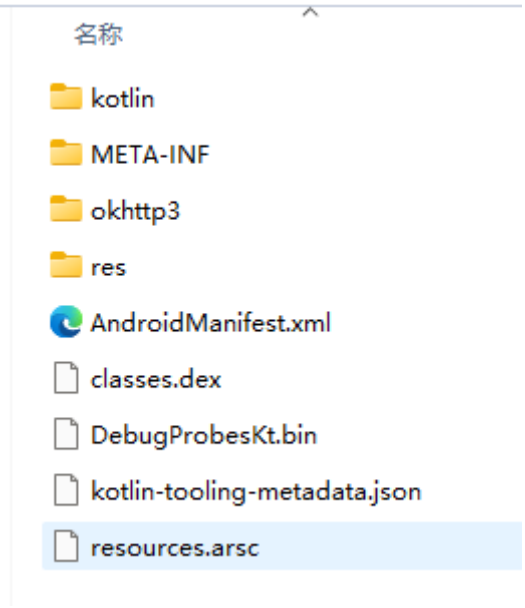
二、APK文件结构、双开、汉化、基础

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

文件	注释
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



2、双开apk原理

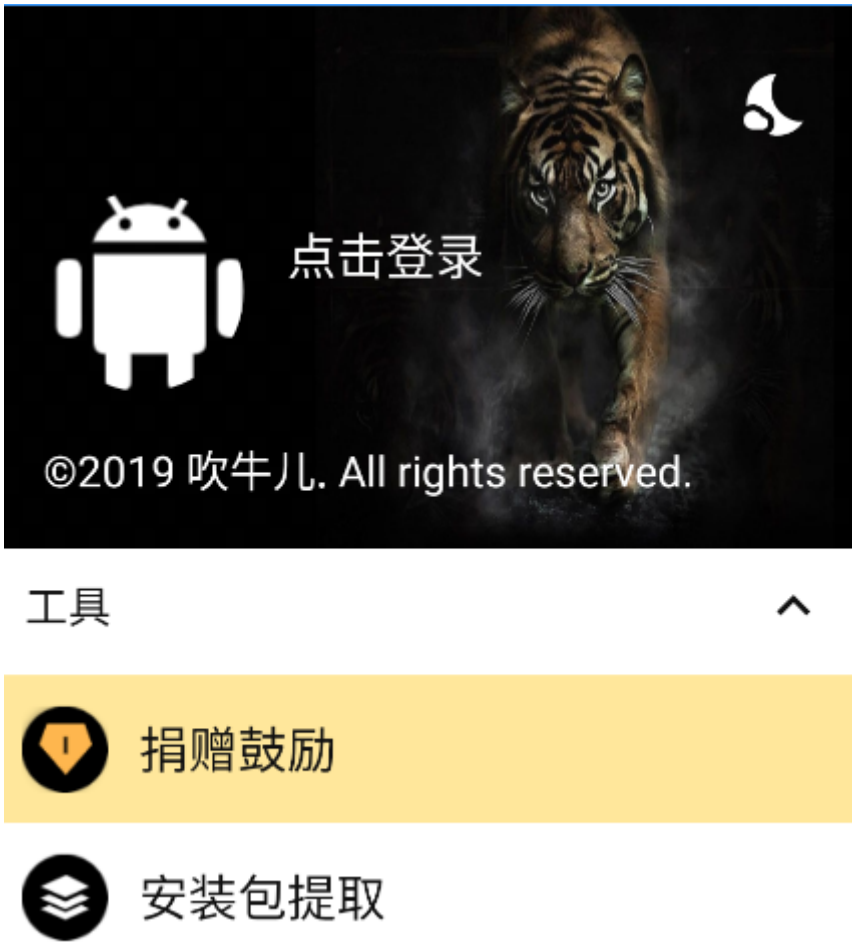
双开：简单来说，就是手机同时运行两个或多个相同的应用，例如同时运行两个微信

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{过}{滤}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、双开

学习双开,这里使用np管理器，（mt管理器部分功能需要会员，np管理器不需要）

使用np管理器，（断网使用）点击左侧 提取安装包，选择要测试的app，提取安装包

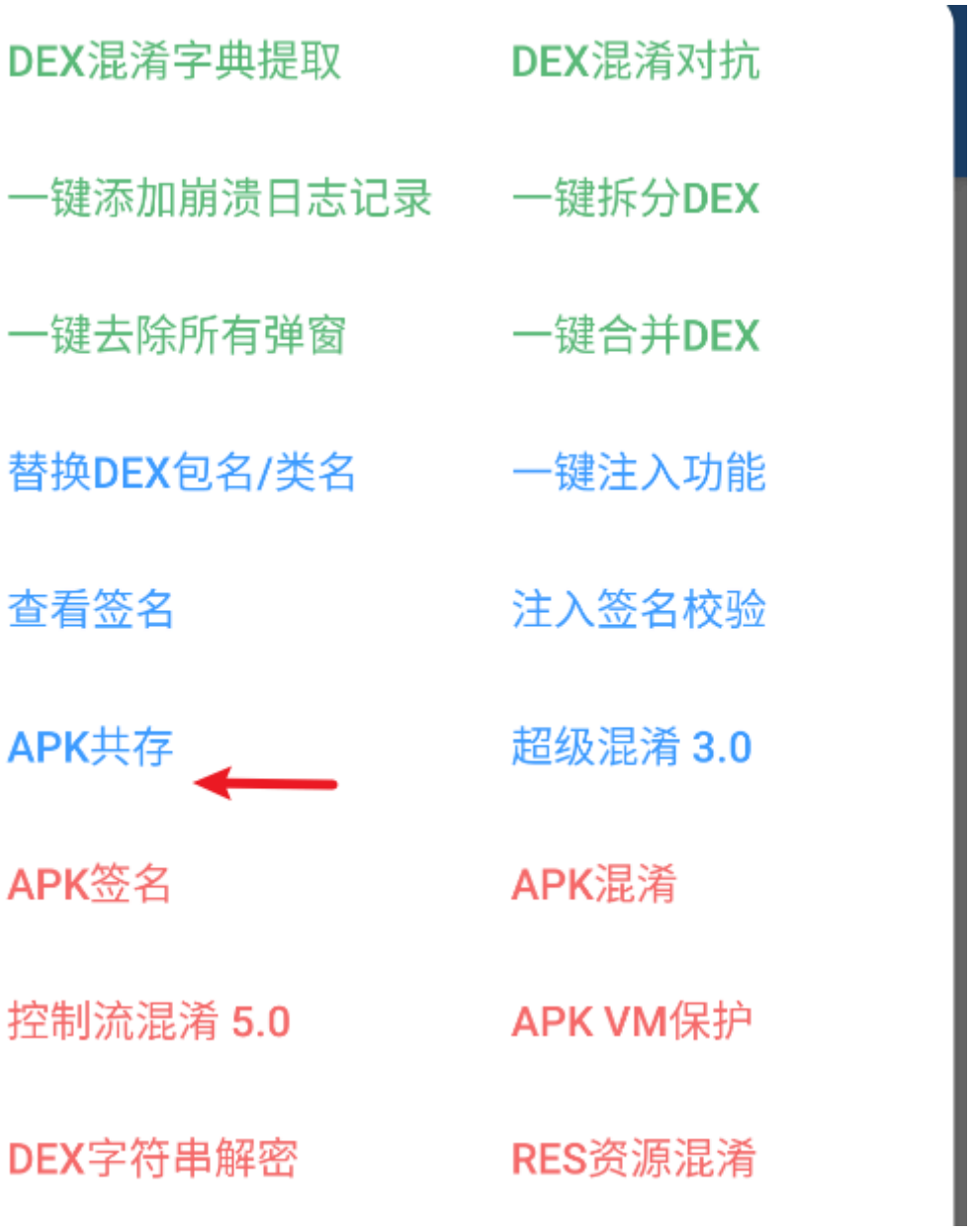


提取安装包后进行定位，点击左侧---功能-apk共存

wuaipojie
1.0

包名	com.zj.wuaipojie
版本号	1
签名状态	V2
加固状态	未加固
安装包大小	5.57M
已安装	1.0 (1)

功能　　查看　　安装


DEX混淆字典提取
DEX混淆对抗
一键添加崩溃日志记录
一键拆分DEX
一键去除所有弹窗
一键合并DEX
替换DEX包名/类名
一键注入功能
查看签名
注入签名校验
APK共存
超级混淆 3.0
APK签名
APK混淆
控制流混淆 5.0
APK VM保护
DEX字符串解密
RES资源混淆

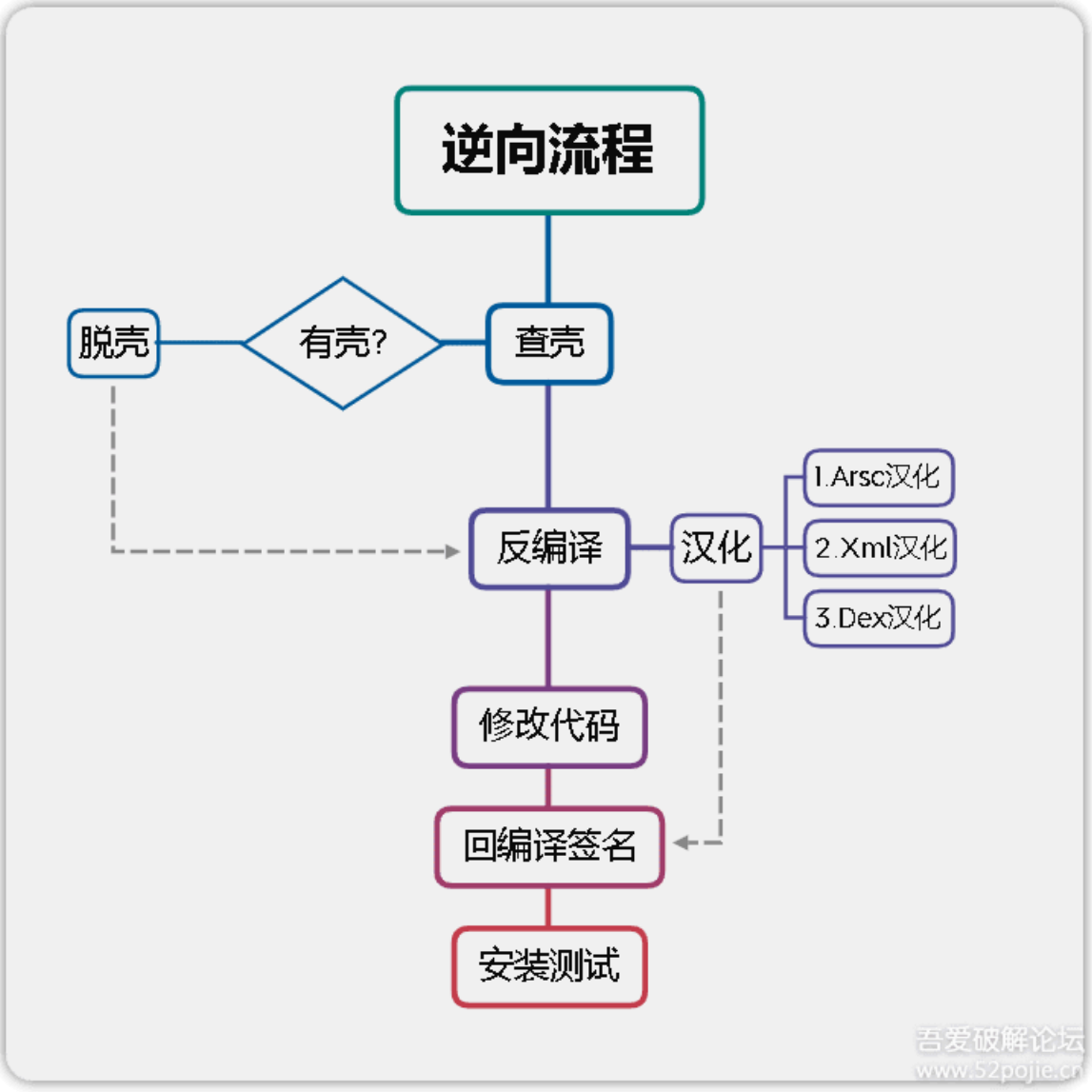
安装

2、汉化

汉化：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

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

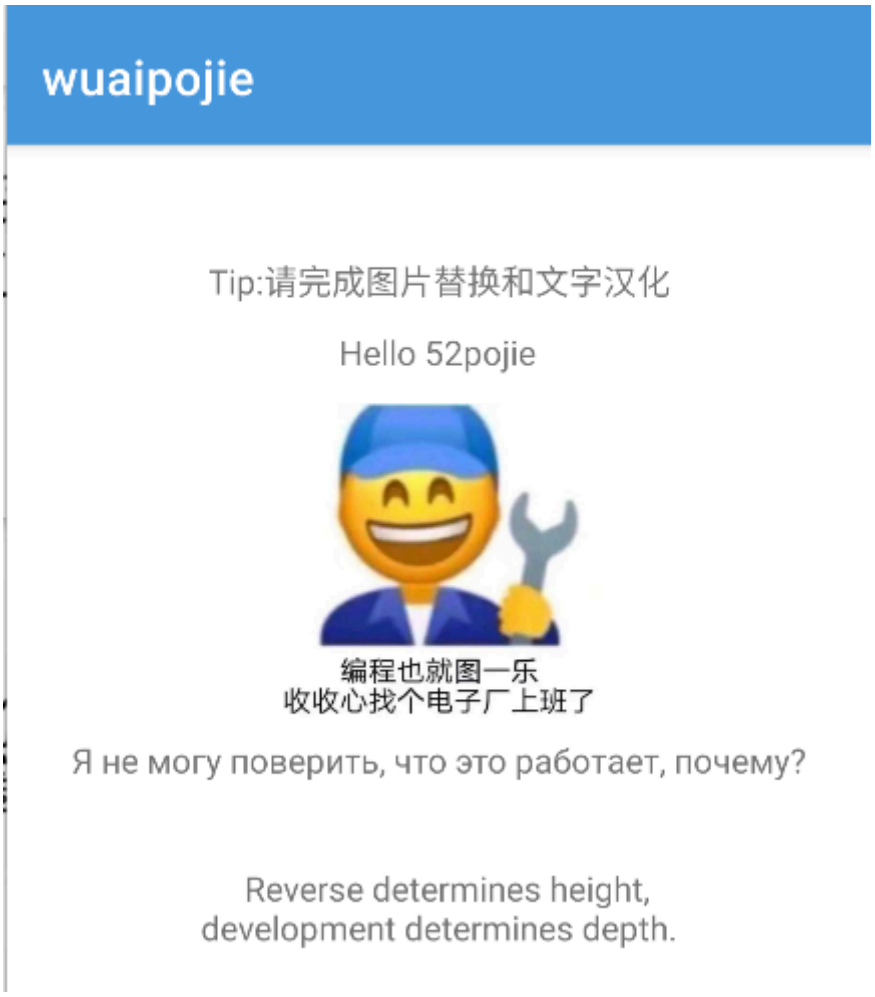
流程图


逆向流程
脱壳
有壳?
查壳
反编译
汉化
1.Arsc汉化
2.Xml汉化
3.Dex汉化
修改代码
回编译签名
安装测试
吾爱破解论坛
www.52pojie.cn

打开app

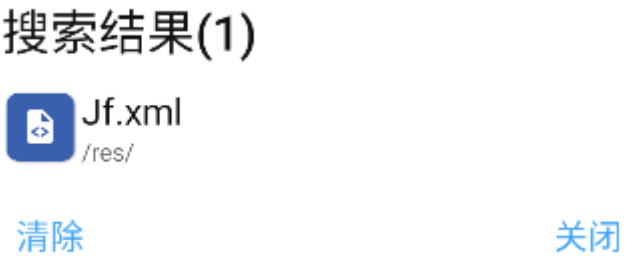
8:07
挑战
APK结构&
双开&汉化&
APK基础修改
第一关
初识smail语法&
快速定位之id与
字符
第二关
去广告
&布局优化
&xml&弹窗定位
主页
挑战
关于

第一关:

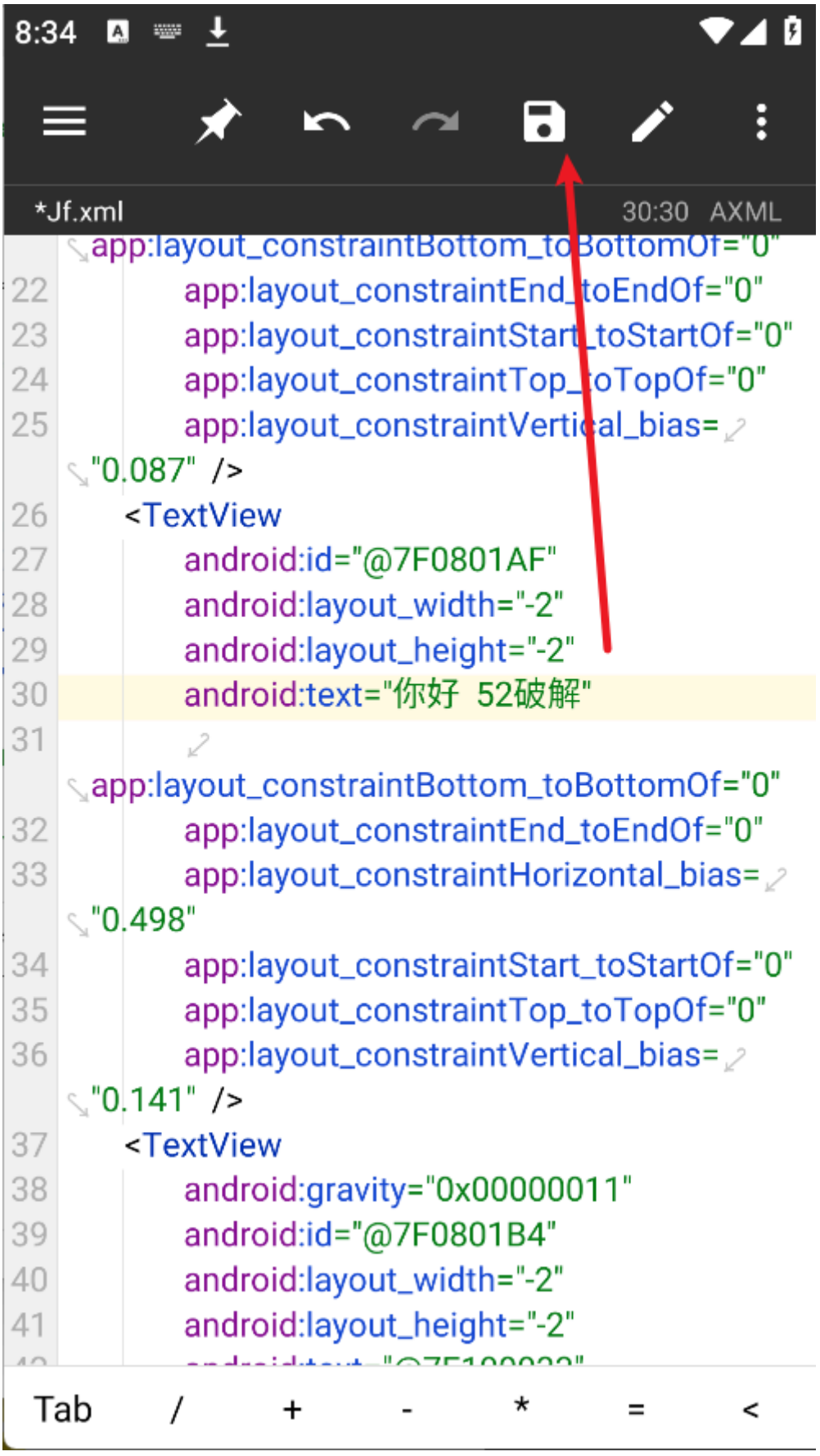


第一步打开mt管理器，找到安装包，点击查看

然后点击右侧的设置找到搜索，全局搜索hello 52pojie



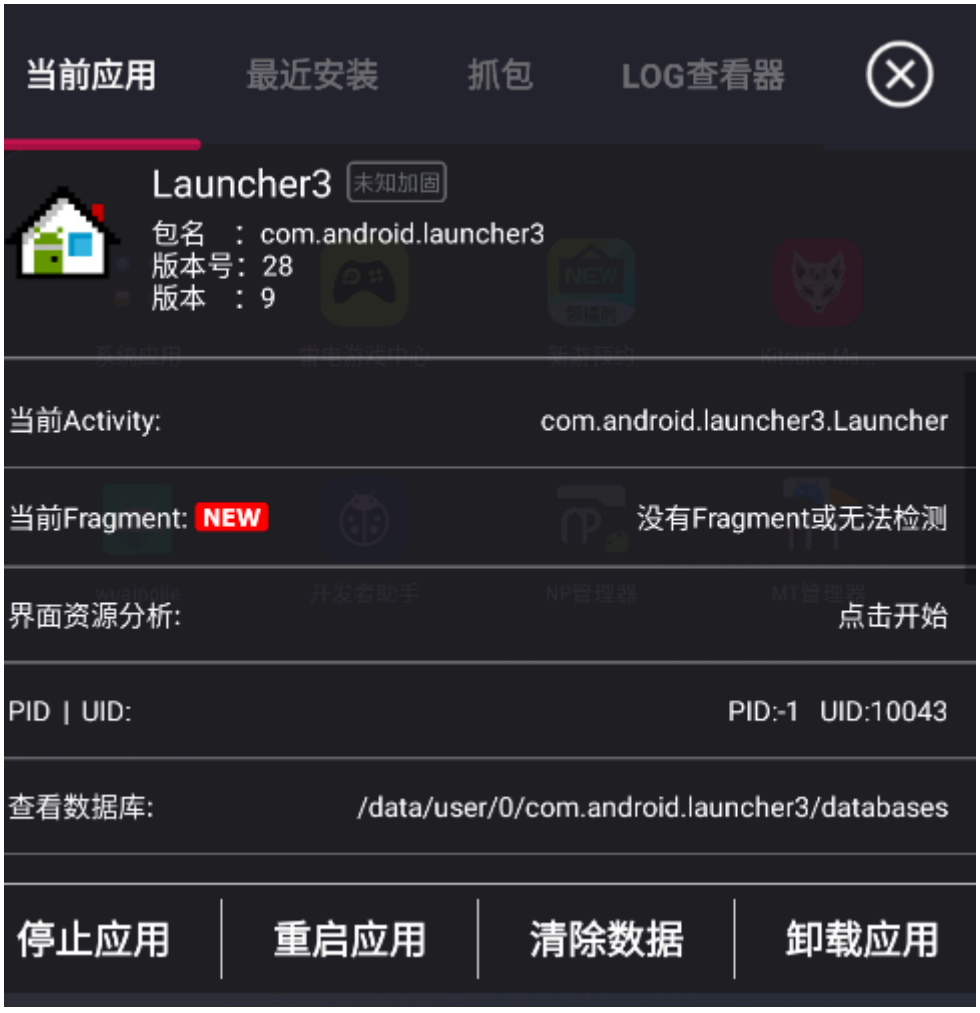
双击 --打开方式，使用反编译 ，修改，保存



卸载掉原来的文件。自动签名 重新安装

文件夹: 0 文件: 2 储存: 1.96G/32.73G
..
wuaipojie_1.0.apk
24-11-21 20:36 5.61M
wuaipojie_1.0.apk.bak
24-11-20 22:57 5.57M
acct
24-11-21 19:58
bin
24-11-07 15:37
cache
24-11-21 19:58
wuaipojie
1.0
包名 com.zj.wuaipojie
版本号 1
安装包大小 5.61M
签名状态 V1 + V2 + V3
加固状态 未加固
功能
查看
安装

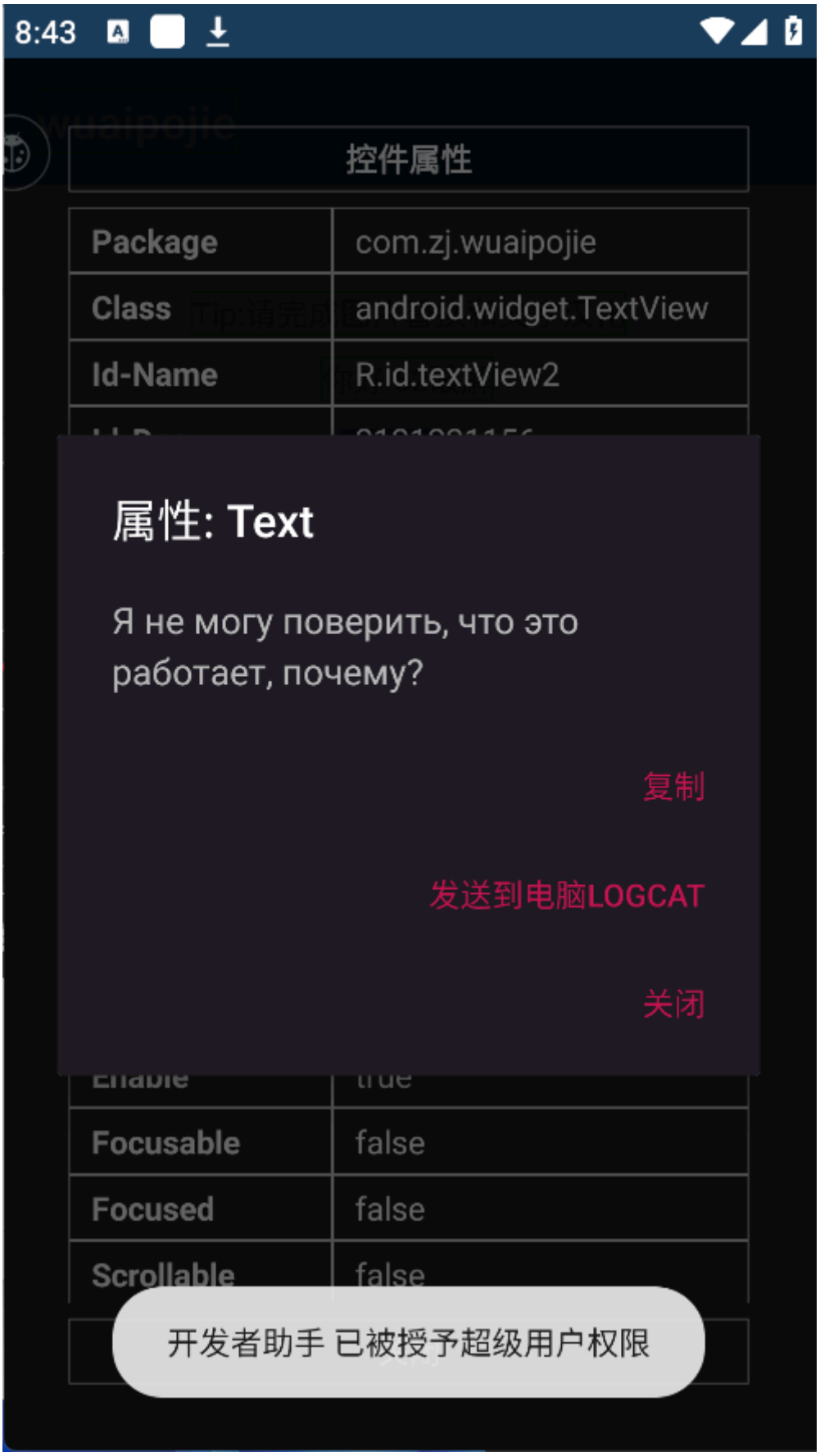
使用开发者助手app，打开app给予权限



wuaipojie
Tip:请完成图片替换和文字汉化
你好 52破解
编程也就图一乐
收收心找个电子厂上班了
Я не могу поверить, что это работает, почему?
Reverse determines height,
development determines depth.

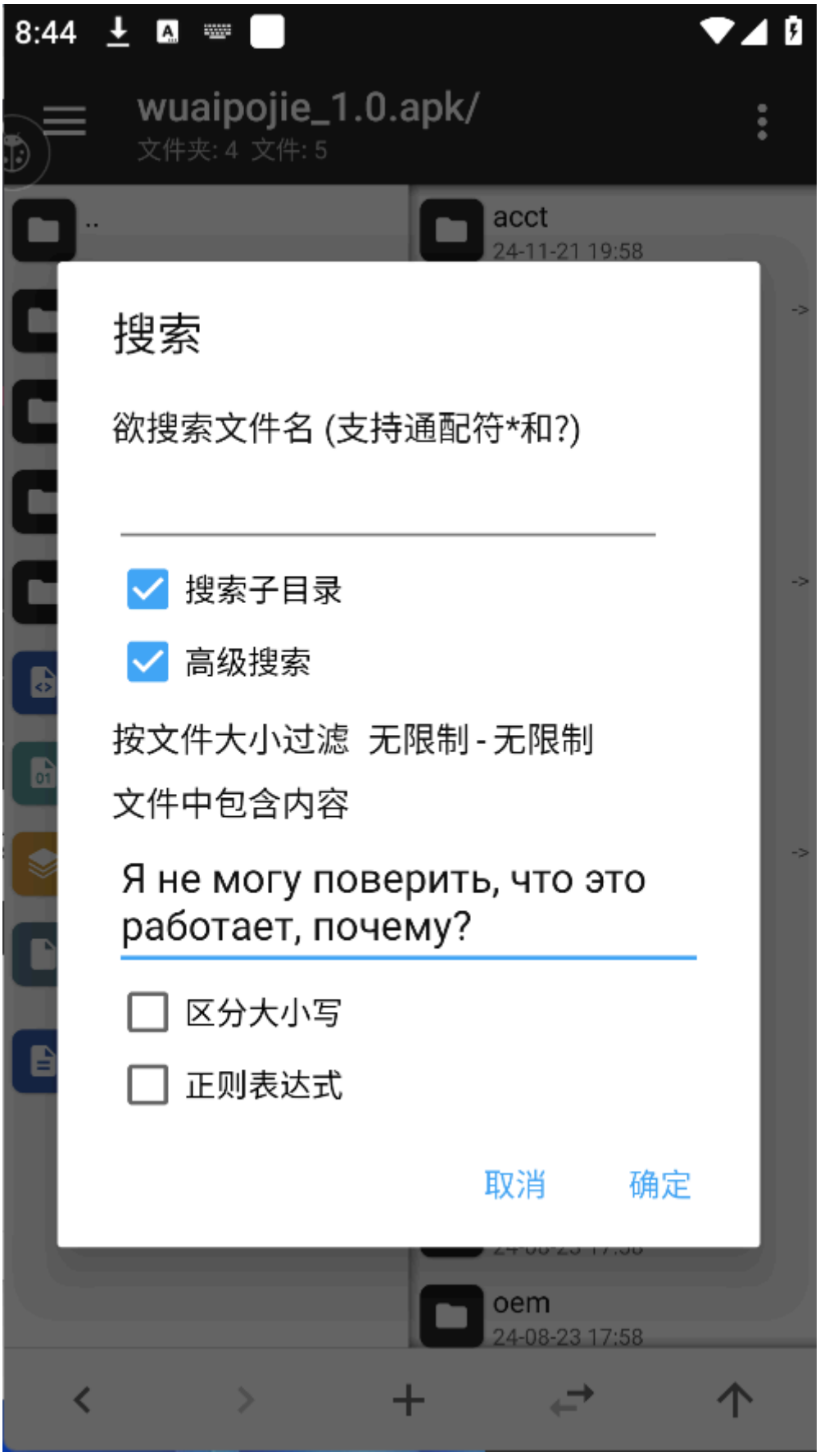
点击页面资源分析

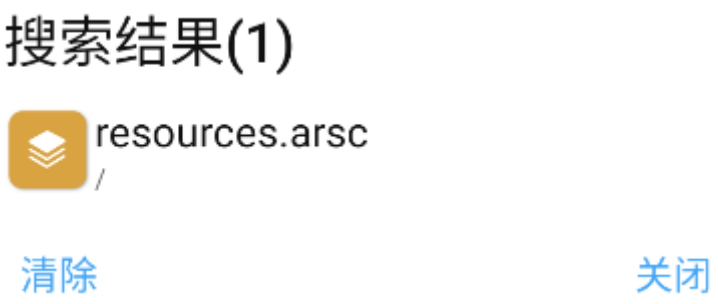
点击分析，然后点击需要复制的文字



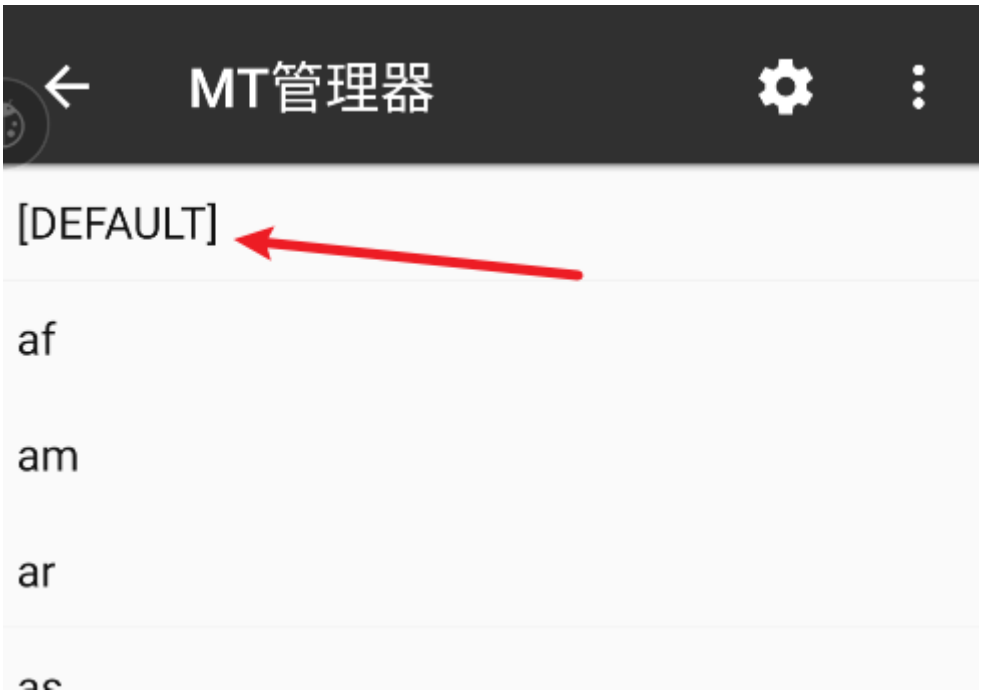
Я не могу поверить, что это работает, почему?

重复以上步骤

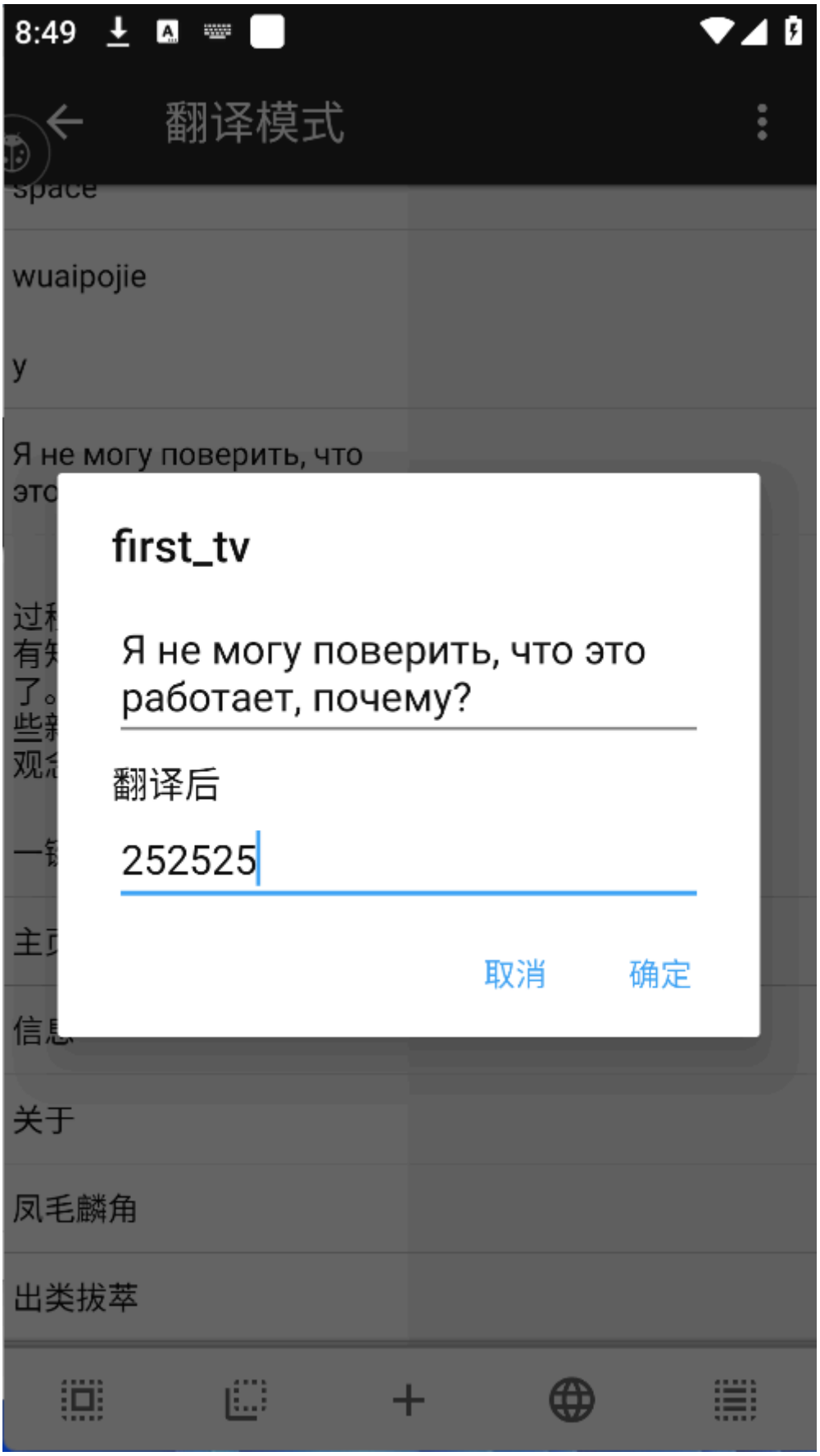

8:44
wuaipojie_1.0.apk/
文件夹: 4 文件: 5
..
acct
24-11-21 19:58
搜索
欲搜索文件名 (支持通配符*和?)
搜索子目录
高级搜索
按文件大小过滤 无限制 - 无限制
文件中包含内容
Я не могу поверить, что это работает, почему?
区分大小写
正则表达式
取消
确定
oem
24-08-23 17:58



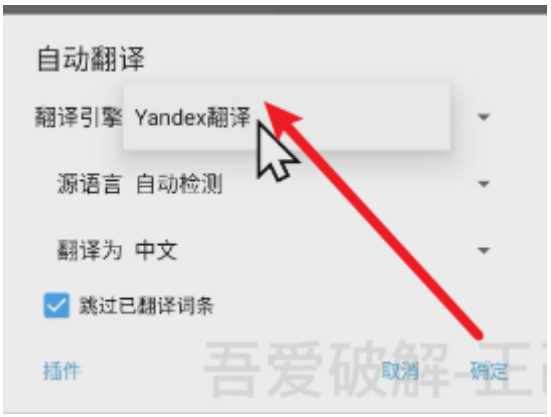
打开文件-方式--翻译模式，选择第一个



然后找到要修改文字，

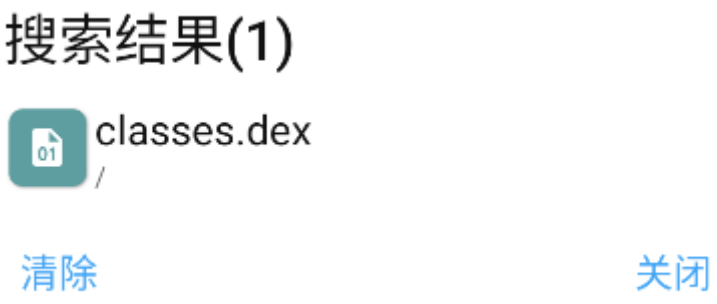


或者使用vip功能一键翻译（yandex翻译需要mt管理器插件管理功能），点击确定后自动化翻译



之后保存，签名退出，这次签名属于2次签名，签名一致，可以覆盖安装

先试用开发者管理工具（界面分析）进行复制

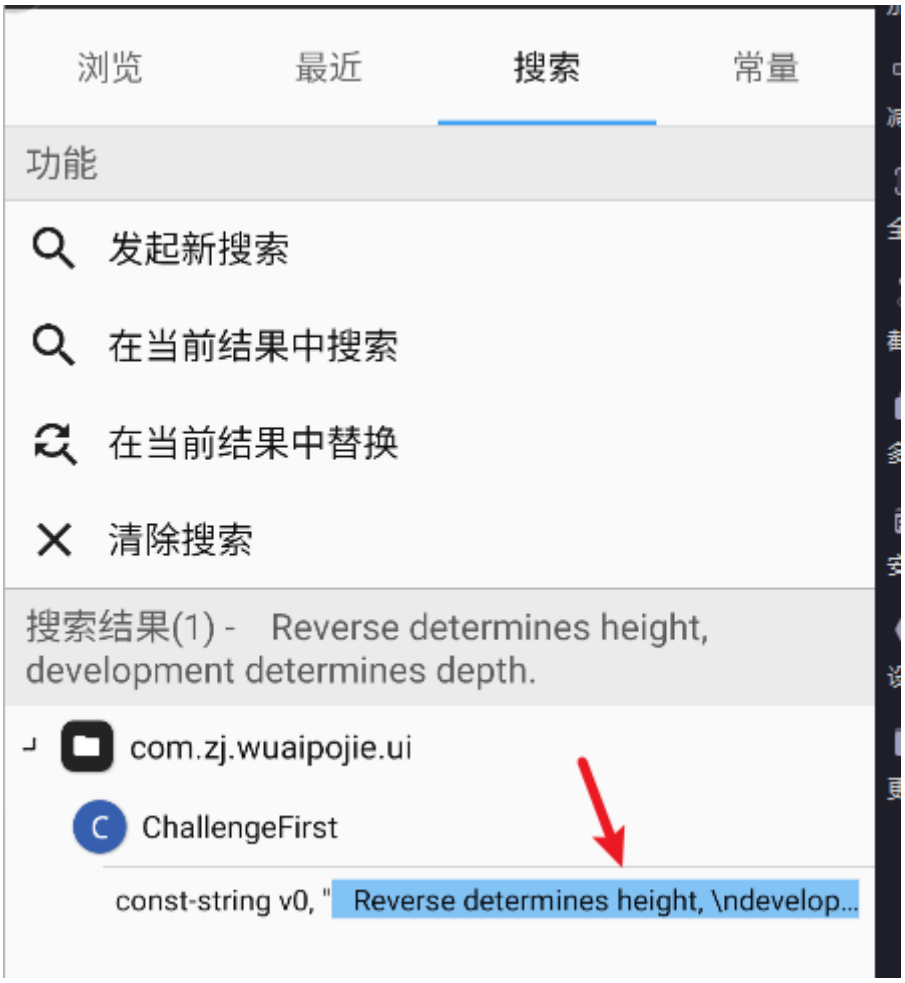


打开方式使用dex编辑器++，然

打开方式...

- Dex编辑器++
- Dex编辑器
- Dex修复
- Dex转Jar
- Dex转Smali
- 翻译模式

9:08
Dex编辑器++
classes.dex
浏览
最近
搜索
常量
搜索
查找内容
velopment determines depth.
路径
/
搜索类型 字符串
搜索子目录
区分大小写
正则表达式
完全匹配
取消
确定
开发者助手 已被授予超级用户权限



进去后，点击，可以选择自动翻译（要会员），或者手工改，保存，退出，安装

9:12

*ChallengeFirst [43 - 71] : onCreate()

```
54      const p1, 0x7f0801b5
55
56      .line 12
57      invoke-virtual {p0, p1}, Lcom/zj/wuaipojie/
        ui/ChallengeFirst;->findViewById(I)Landroid/
        view/View;
58
59      move-result-object p1
60
61      check-cast p1, Landroid/widget/TextView;
62
63      const-string v0, "33333333333333333"
64
65      .line 13
66      check-cast v0, Ljava/lang/CharSequence;
67
68      invoke-virtual {p1, v0}, Landroid/widget/
        TextView;->setText(Ljava/lang/
        CharSequence;)V
69
70      return-void
71  .end method
72
```

开发者助手 已被授予超级用户权限

Tab / + - * = <

Tip:请完成图片替换和文字汉化

你好 52破解

编程也就图一乐
收收心找个电子厂上班了

252525

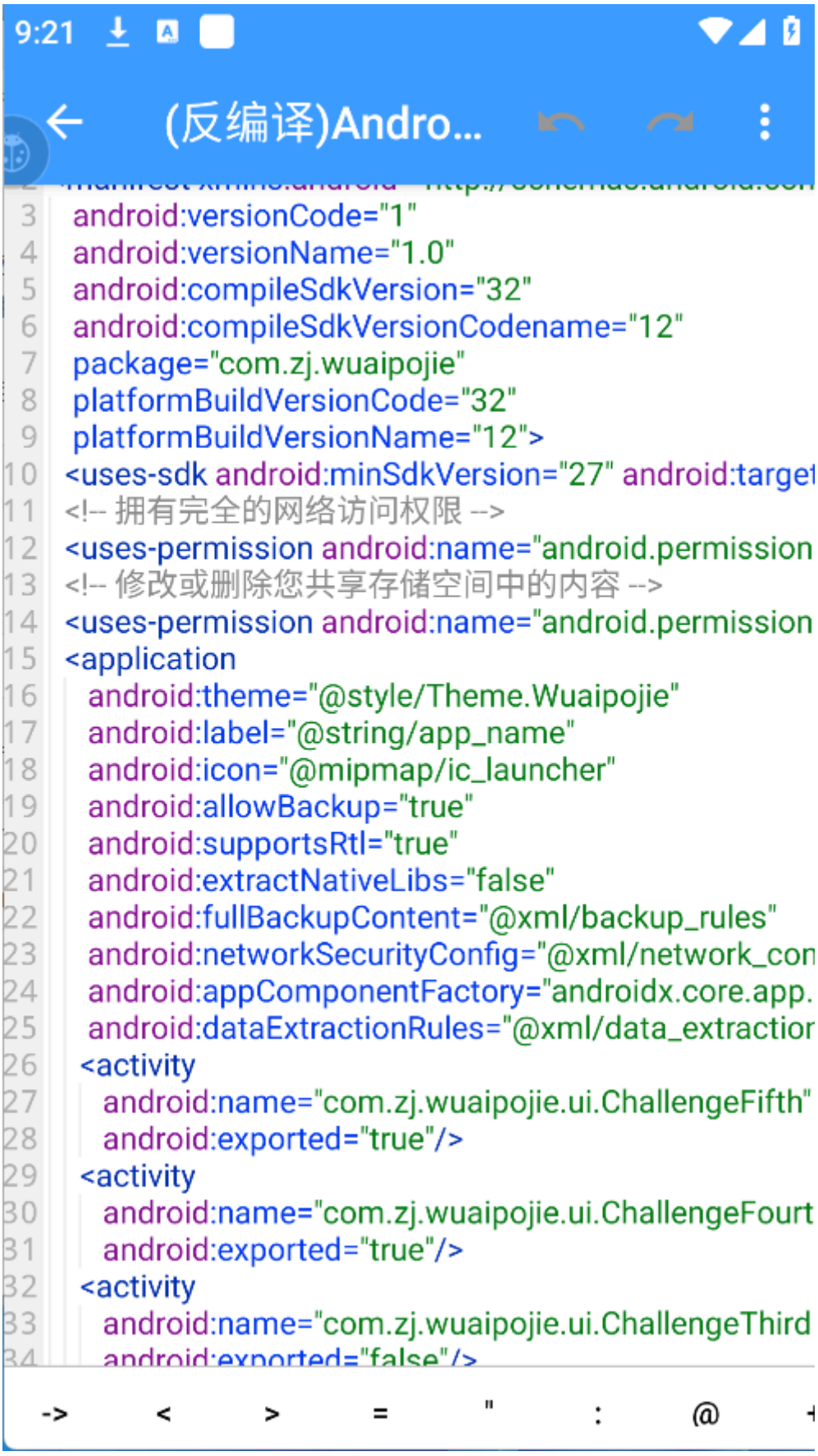
3333333333333333

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

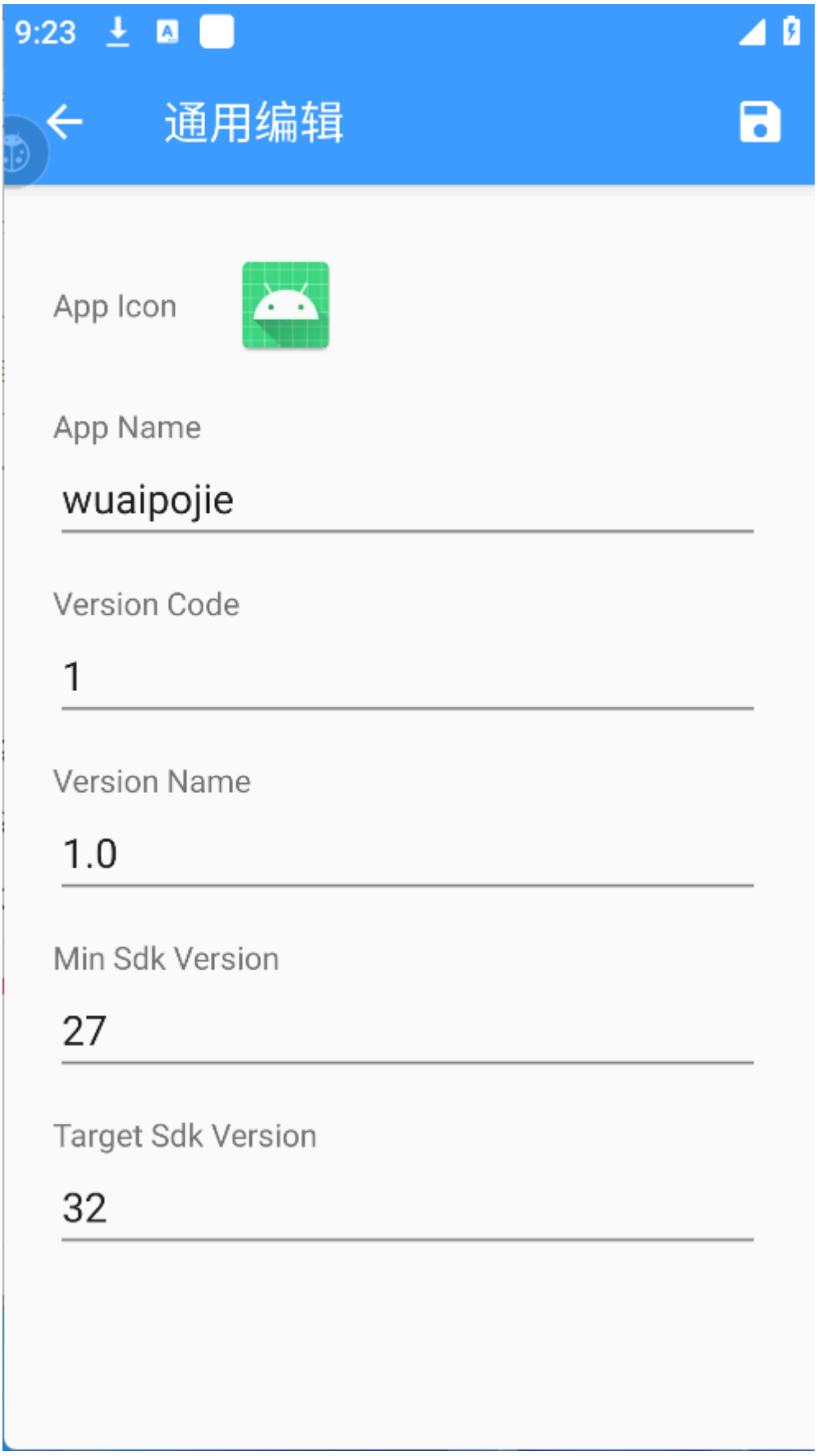
查看文件，可以按需修改



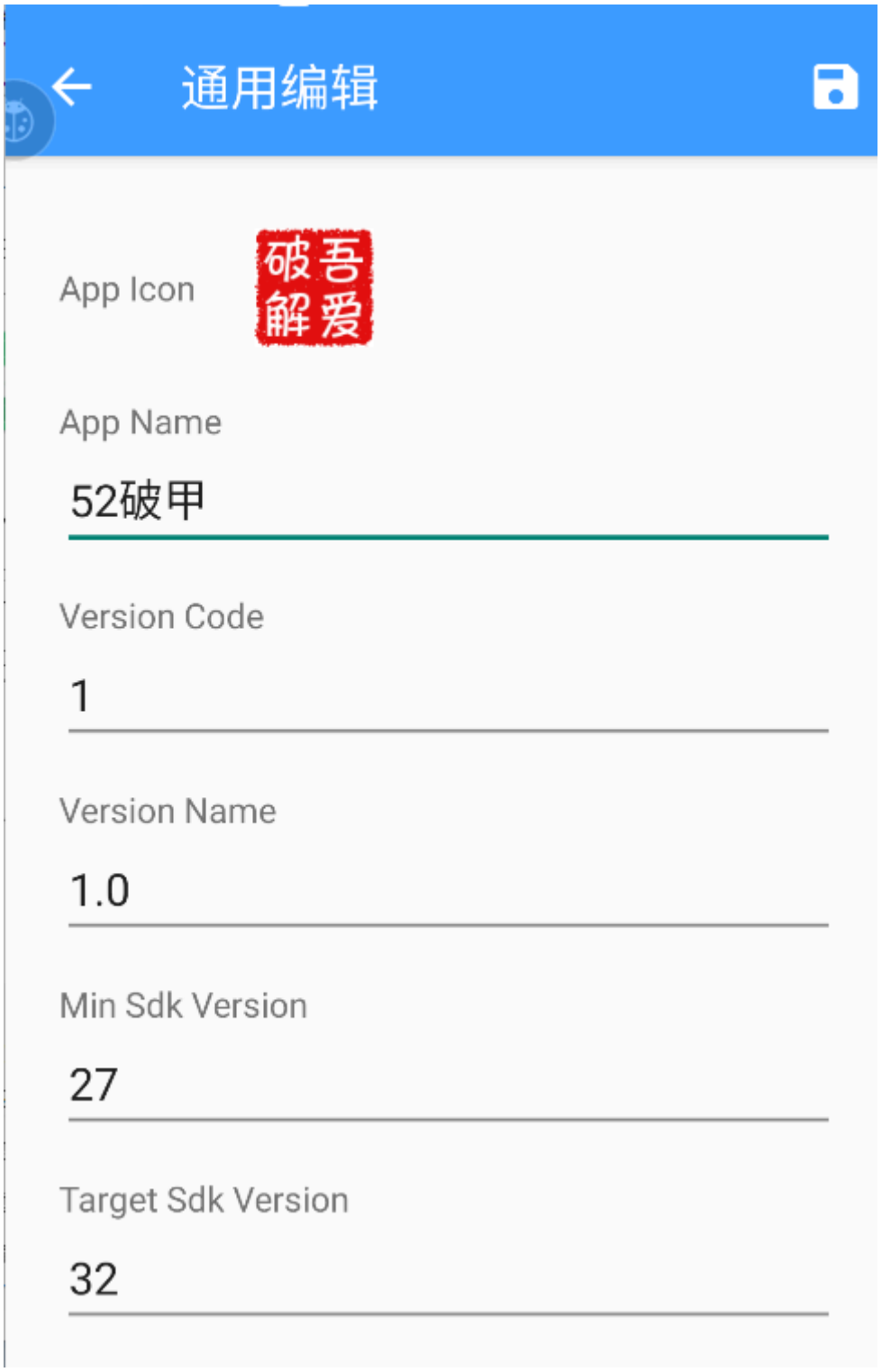
3、修改app包名、图标

修改app的图片，版本，姓名。使用np管理器-功能--通用编辑

wuaipojie
1.0
包名
com.zj.wuaipojie
版本号
1
签名状态
V1 + V2 + V3
加固状态
未加固
安装包大小
5.69M
功能
查看
安装


9:23
通用编辑
App Icon
App Name
wuaipojie
Version Code
1
Version Name
1.0
Min Sdk Version
27
Target Sdk Version
32

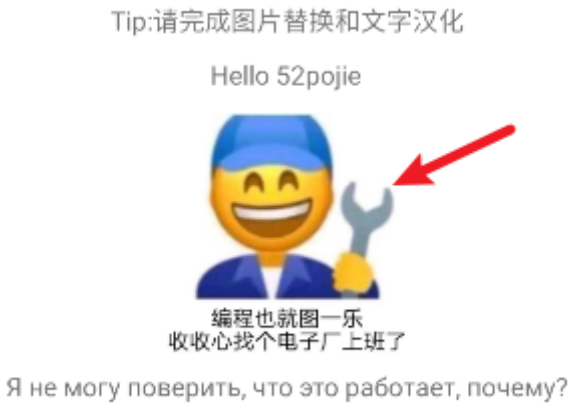
修改后，签名，安装


通用编辑
App Icon
破解
吾爱
App Name
52破甲
Version Code
1
Version Name
1.0
Min Sdk Version
27
Target Sdk Version
32

其他----修改图片

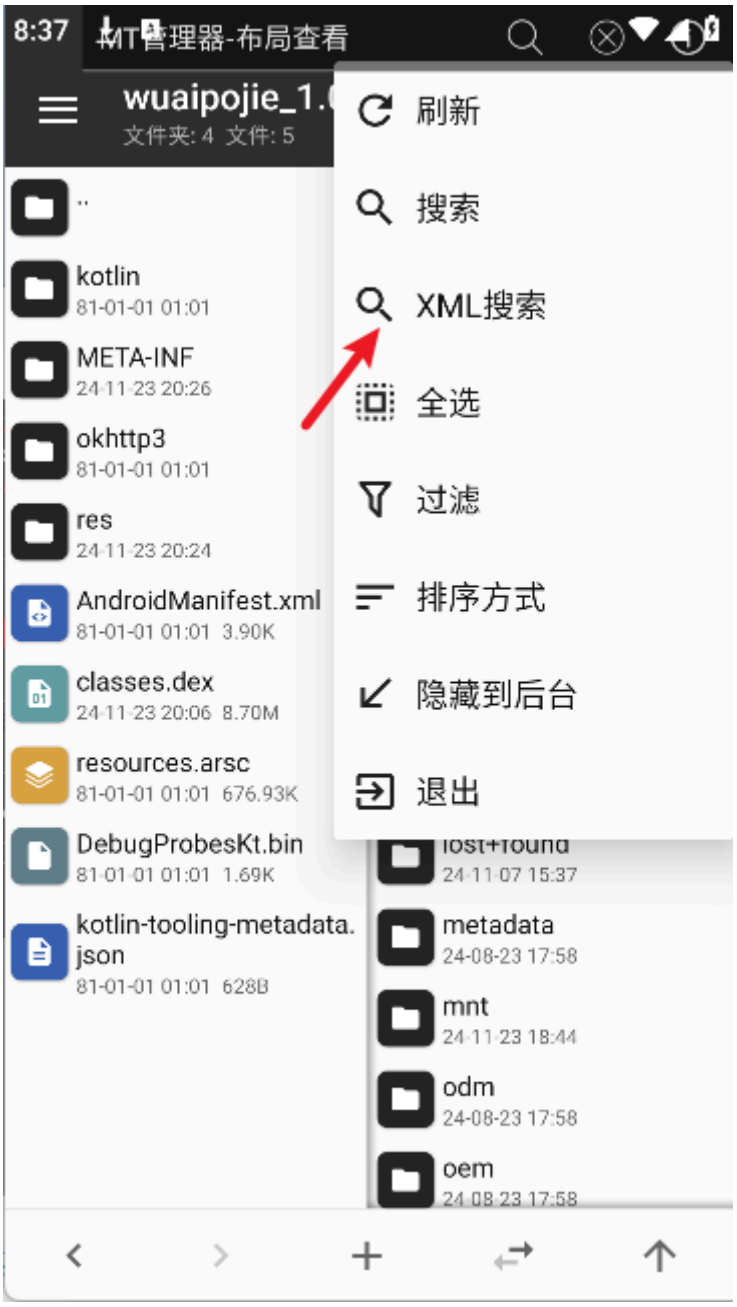
（开发者助手app使用方法可以到第三章的《修改寄存器的值》中学习）

（开发者助手app使用方法可以到第四章的《去除界面文字或其他的布局》中学习）



首先使用开发者助手app或者开发助手app，界面布局查看功能获取hex

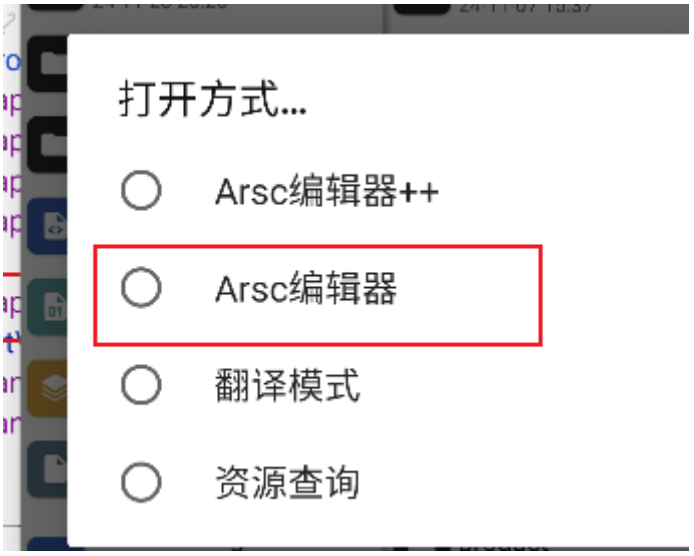
复制 0x7f0800d7 然后再xml搜，资源+16进制


8:37
MT管理器-布局查看
wuaipojie_1.
文件夹: 4 文件: 5
刷新
搜索
XML搜索
全选
过滤
排序方式
隐藏到后台
退出
..
kotlin
81-01-01 01:01
META-INF
24-11-23 20:26
okhttp3
81-01-01 01:01
res
24-11-23 20:24
AndroidManifest.xml
81-01-01 01:01 3.90K
classes.dex
24-11-23 20:06 8.70M
resources.arsc
81-01-01 01:01 676.93K
DebugProbesKt.bin
81-01-01 01:01 1.69K
kotlin-tooling-metadata.
json
81-01-01 01:01 628B
lost+found
24-11-07 15:37
metadata
24-08-23 17:58
mnt
24-11-23 18:44
odm
24-08-23 17:58
oem
24-08-23 17:58

进入文件，查看

下面这个代表了图片地址 7F0D000C

在 resources.arsc文件 右键 arsc编辑器打开 ID定位资源搜索



8:45
MT管理器-布局查看
7F0D000C
com.zj.wuaipojie/mipmap-xxxhdpi/first_img
res/cf.jpg

图片名为cf.jpg

8:46
MT管理器-布局查看
wuaipojie_1.0.apk/
文件夹: 4 文件: 5
..
kotlin
81-01-01 01:01
META-INF
24-11-23 20:26
data
24-11-17 14:10
dev
24-11-23 20:26
etc
24-11-07 15:37
搜索
欲搜索文件名 (支持通配符*和?)
cf.jpg
搜索子目录
高级搜索
取消
确定
json
81-01-01 01:01 628B
24-11-07 15:37
sbin
24-11-23 18:44
sdcard
24-11-07 15:37
storage

替换即可

三、初识smali，vip终结者

1.什么是JVM、Dalvik、ART

JVM是JAVA虚拟机，运行JAVA字节码程序
Dalvik是Google专门为Android设计的一个虚拟机，Dalvik有专属的文件执行格式dex(Dalvik executable)
Art(Android Runtime)相当于Dalvik的升级版，本质与Dalvik无异

2.smali及其语法

smali是Dalvik的寄存器语言，smali代码是dex反编译而来的。

关键字

名称	注释
.class	类名

名称	注释
.super	父类名，继承的上级类名名称
.source	源名
.field	变量
.method	方法名
.register	寄存器
.end method	方法名的结束
public	公有
protected	半公开，只有同一家人才能用
private	私有，只能自己使用
.parameter	方法参数
.prologue	方法开始
.line xxx	位于第xxx行

数据类型对应

smali类型	java类型	注释
V	void	无返回值
Z	boolean	布尔值类型，返回0或1
B	byte	字节类型，返回字节
S	short	短整数类型，返回数字
C	char	字符类型，返回字符
I	int	整数类型，返回数字
J	long （64位 需要2个寄存器存储）	长整数类型，返回数字
F	float	单浮点类型，返回数字
D	double （64位 需要2个寄存器存储）	双浮点类型，返回数字
string	String	文本类型，返回字符串
Lxxx/xxx/xxx	object	对象类型，返回对象

常用指令

关键字	注释
const	重写整数属性，真假属性内容，只能是数字类型

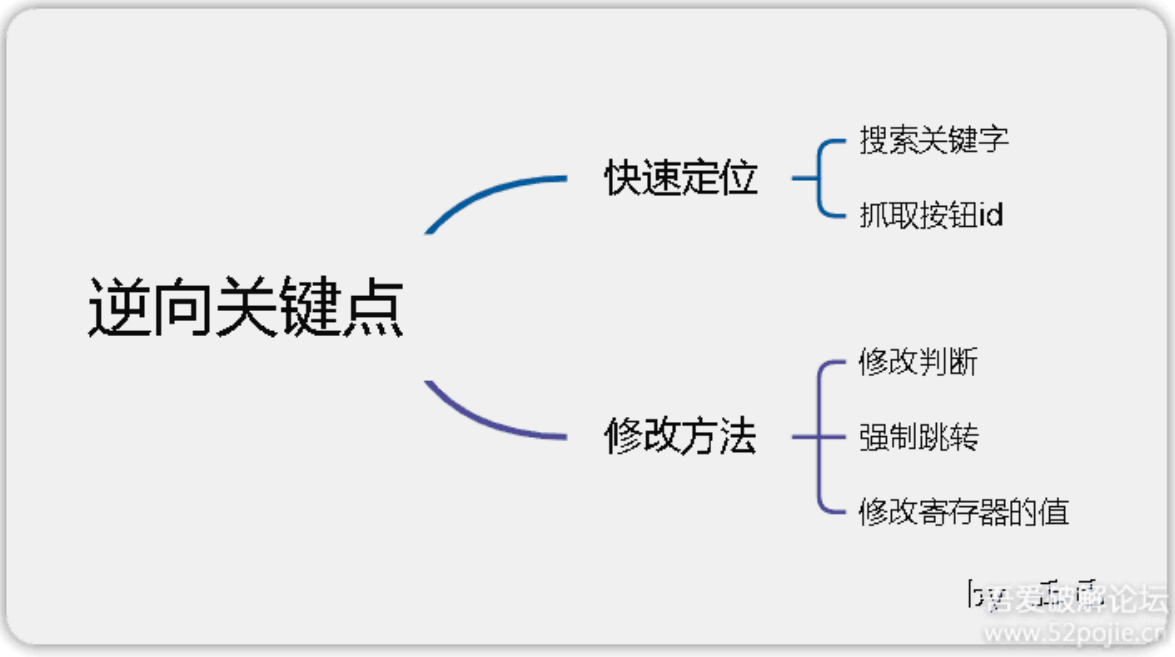
关键字	注释
const-string	重写字符串内容
const-wide	重写长整数类型，多用于修改到期时间。
return	返回指令
if-eq	全称equal(a=b)，比较寄存器ab内容，相同则跳
if-ne	全称not equal(a!=b)，ab内容不相同则跳
if-eqz	全称equal zero(a=0)，z即是0的标记，a等于0则跳
if-nez	全称not equal zero(a!=0)，a不等于0则跳
if-ge	全称greater equal(a>=b)，a大于或等于则跳
if-le	全称little equal(a<=b)，a小于或等于则跳
goto	强制跳到指定位置
switch	分支跳转，一般会有多个分支线，并根据指令跳转到适当位置
iget	获取寄存器数据

其余指令可用语法工具查询

定位方法：搜索弹窗关键字、抓取按钮id

其余指令可用语法工具查询
定位方法：搜索弹窗关键字、抓取按钮id
例子：

修改方法：修改判断、强制跳转、修改寄存器的值



3.寄存器

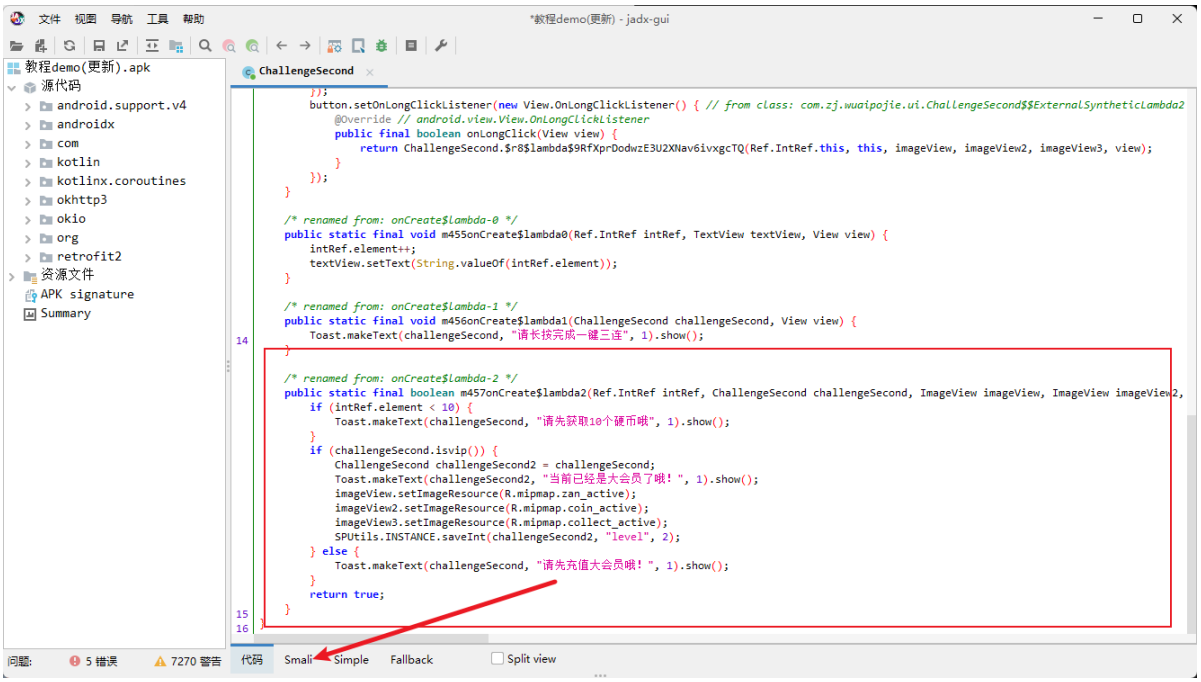
在smali里的所有操作都必须经过寄存器来进行:本地寄存器用v开头数字结尾的符号来表示，如v0、 v1、v2。 参数寄存器则使用p开头数字结尾的符号来表示，如p0、p1、p2。特别注意的是，p0不一定是函数中的第一个参数，在非static函数中，p0代指“this"，p1表示函数的第一个 参数，p2代表函数中的第二个参数。而在static函数中p0才对应第一个参数(因为Java的static方法中没有this方法)

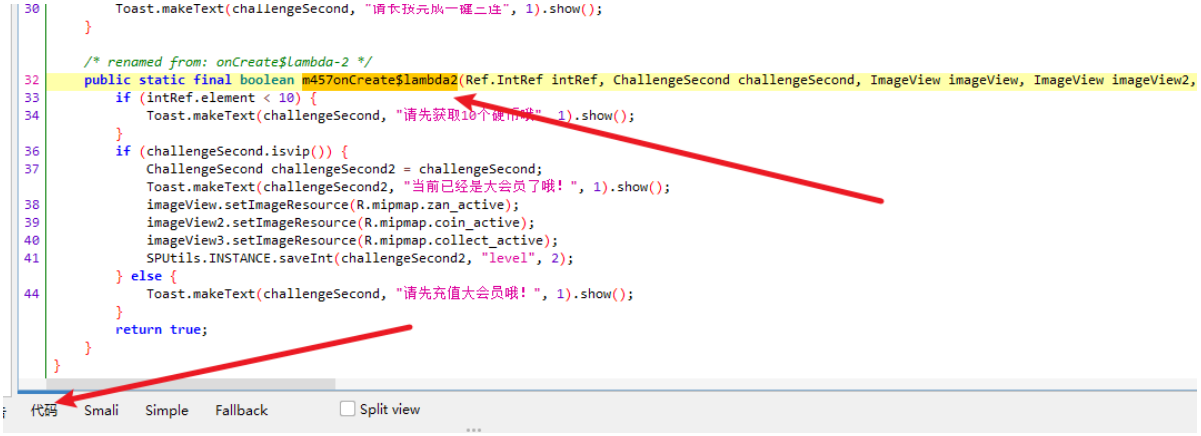
安装核心破解工具

或者直接在LSPosed模块的模块功能安装，安装后重启

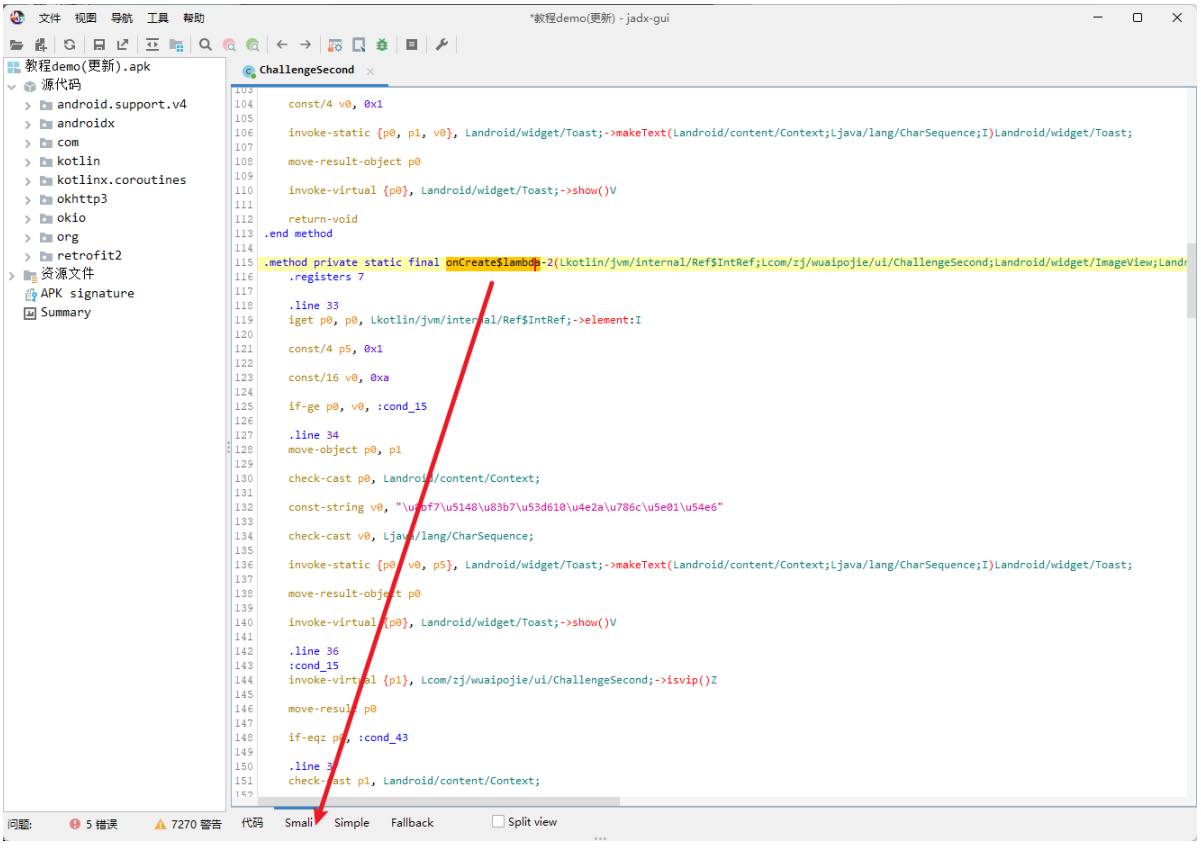
进入新app的第二关

反编译app代码，查看smali


文件 视图 导航 工具 帮助
*教程demo(更新) - jadx-gui
教程demo(更新).apk
源代码
android.support.v4
androidx
com
kotlin
kotlinx.coroutines
okhttp3
okio
org
retrofit2
资源文件
APK signature
Summary
ChallengeSecond
}});
button.setOnLongClickListener(new View.OnLongClickListener() { // from class: com.zj.wuaipojie.ui.ChallengeSecond$$ExternalSyntheticLambda2
@Override // android.view.View.OnLongClickListener
public final boolean onLongClick(View view) {
return ChallengeSecond.$r8$lambda$9RfXprDodwzE3U2XNav6ivxgcTQ(Ref.IntRef.this, this, imageView, imageView2, imageView3, view);
}
});
}
/* renamed from: onCreate$lambda-0 */
public static final void m455onCreate$lambda0(Ref.IntRef intRef, TextView textView, View view) {
intRef.element++;
textView.setText(String.valueOf(intRef.element));
}
/* renamed from: onCreate$lambda-1 */
public static final void m456onCreate$lambda1(ChallengeSecond challengeSecond, View view) {
Toast.makeText(challengeSecond, "请长按完成一键三连", 1).show();
}
/* renamed from: onCreate$lambda-2 */
public static final boolean m457onCreate$lambda2(Ref.IntRef intRef, ChallengeSecond challengeSecond, ImageView imageView, ImageView imageView2,
if (intRef.element < 10) {
Toast.makeText(challengeSecond, "请先获取10个硬币哦", 1).show();
}
if (challengeSecond.isvip()) {
ChallengeSecond challengeSecond2 = challengeSecond;
Toast.makeText(challengeSecond2, "当前已经是大会员了哦！", 1).show();
imageView.setImageResource(R.mipmap.zan_active);
imageView2.setImageResource(R.mipmap.coin_active);
imageView3.setImageResource(R.mipmap.collect_active);
SPUtils.INSTANCE.saveInt(challengeSecond2, "level", 2);
} else {
Toast.makeText(challengeSecond, "请先充值大会员哦！", 1).show();
}
return true;
}
}
问题: 5 错误 7270 警告
代码 Smali Simple Fallback Split view


Toast.makeText(challengeSecond, "请长按完成一键三连", 1).show();
}
/* renamed from: onCreate$lambda-2 */
public static final boolean m457onCreate$lambda2(Ref.IntRef intRef, ChallengeSecond challengeSecond, ImageView imageView, ImageView imageView2,
if (intRef.element < 10) {
Toast.makeText(challengeSecond, "请先获取10个硬币哦", 1).show();
}
if (challengeSecond.isvip()) {
ChallengeSecond challengeSecond2 = challengeSecond;
Toast.makeText(challengeSecond2, "当前已经是大会员了哦！", 1).show();
imageView.setImageResource(R.mipmap.zan_active);
imageView2.setImageResource(R.mipmap.coin_active);
imageView3.setImageResource(R.mipmap.collect_active);
SPUtils.INSTANCE.saveInt(challengeSecond2, "level", 2);
} else {
Toast.makeText(challengeSecond, "请先充值大会员哦！", 1).show();
}
return true;
}
}
代码 Smali Simple Fallback Split view

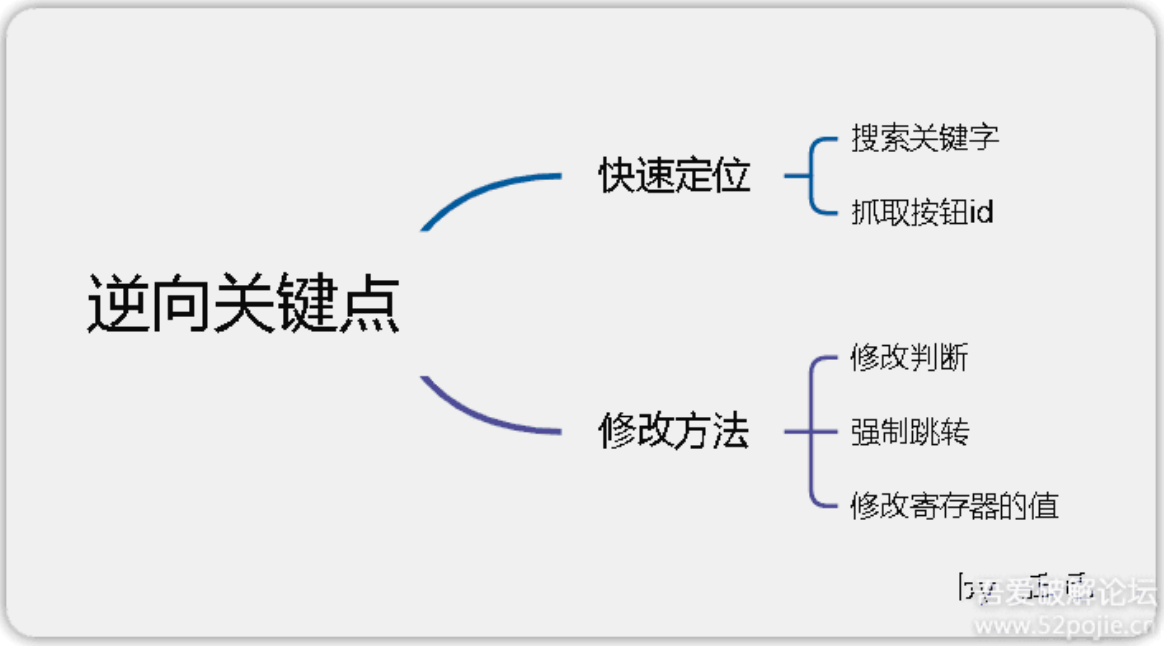
对应


文件 视图 导航 工具 帮助
*教程demo(更新) - jadx-gui
教程demo(更新).apk
源代码
android.support.v4
androidx
com
kotlin
kotlinx.coroutines
okhttp3
okio
org
retrofit2
资源文件
APK signature
Summary
ChallengeSecond
const/4 v0, 0x1
invoke-static {p0, p1, v0}, Landroid/widget/Toast;->makeText(Landroid/content/Context;Ljava/lang/CharSequence;I)Landroid/widget/Toast;
move-result-object p0
invoke-virtual {p0}, Landroid/widget/Toast;->show()V
return-void
.end method
.method private static final onCreate$lambda-2(Lkotlin/jvm/internal/Ref$IntRef;Lcom/zj/wuaipojie/ui/ChallengeSecond;Landroid/widget/ImageView;Landr
.registers 7
.line 33
iget p0, p0, Lkotlin/jvm/internal/Ref$IntRef;->element:I
const/4 p5, 0x1
const/16 v0, 0xa
if-ge p0, v0, :cond_15
.line 34
move-object p0, p1
check-cast p0, Landroid/content/Context;
const-string v0, "\u8bf7\u5148\u83b7\u53d610\u4e2a\u786c\u5e01\u54e6"
check-cast v0, Ljava/lang/CharSequence;
invoke-static {p0, v0, p5}, Landroid/widget/Toast;->makeText(Landroid/content/Context;Ljava/lang/CharSequence;I)Landroid/widget/Toast;
move-result-object p0
invoke-virtual {p0}, Landroid/widget/Toast;->show()V
.line 36
:cond_15
invoke-virtual {p1}, Lcom/zj/wuaipojie/ui/ChallengeSecond;->isvip()Z
move-result p0
if-eqz p0, :cond_43
.line 3
check-cast p1, Landroid/content/Context;
问题: 5 错误 7270 警告
代码 Smali Simple Fallback Split view

```
//一个私有、静态、不可变的方法    方法名
.method private static final onCreate$lambda-2(Lkotlin/jvm/internal/Ref$IntRef;Lcom/zj/wuaipojie/ui/ChallengeSecond;Landroid/widget/ImageView;Landroid/widget/ImageView;Landroid/widget/ImageView;Landroid/view/View;)Z //(这里面是方法的参数)这里是方法返回值类型，表示布尔值类型，返回假或真
    .registers 7  //寄存器数量

    .line 33  //代码所在的行数
    iget p0, p0, Lkotlin/jvm/internal/Ref$IntRef;->element:I  //读取p0(第一个参数，参考寄存器知识)中element的值赋值给p0

    const/4 p5, 0x1  //p5赋值1

    const/16 v0, 0xa //v0赋值10，在16进制里a表示10

    if-ge p0, v0, :cond_15  //判断p0的值是否大于或等于v0的值(即p0的值是否大于或等于10)，如果大于或等于则跳转到:cond_15

    .line 34  //以下是常见的Toast弹窗代码
    check-cast p1, Landroid/content/Context; //检查Context对象引用

    const-string p0, "请先获取10个硬币哦" //弹窗文本信息，把""里的字符串数据赋值给p0

    check-cast p0, Ljava/lang/CharSequence; //检查CharSequence对象引用

    invoke-static {p1, p0, p5}, Landroid/widget/Toast;->makeText(Landroid/content/Context;Ljava/lang/CharSequence;I)Landroid/widget/Toast;
    //将弹窗文本、显示时间等信息传给p1

    move-result-object p0  //结果传递给p0

    invoke-virtual {p0}, Landroid/widget/Toast;->show()V  //当看到这个Toast;->show你就应该反应过来这里是弹窗代码

    goto :goto_31  //跳转到:goto_31

    :cond_15 //跳转的一个地址

    invoke-virtual {p1}, Lcom/zj/wuaipojie/ui/ChallengeSecond;->isvip()Z  //判断isvip方法的返回值是否为真(即结果是否为1)

    move-result p0  //结果赋值给p0

    if-eqz p0, :cond_43 //如果结果为0则跳转cond_43地址

    const p0, 0x7f0d0018  //在arsc中的id索引，这个值可以进行查询

    .line 37
    invoke-virtual {p2, p0}, Landroid/widget/ImageView;->setImageResource(I)V //设置图片资源

    const p0, 0x7f0d0008

    .line 38
    invoke-virtual {p3, p0}, Landroid/widget/ImageView;->setImageResource(I)V
```

```
    const p0, 0x7f0d000a

    .line 39
    invoke-virtual {p4, p0}, Landroid/widget/ImageView;->setImageResource(I)V

    .line 40
    sget-object p0, Lcom/zj/wuaipojie/util/SPUtils;-
>INSTANCE:Lcom/zj/wuaipojie/util/SPUtils;

    check-cast p1, Landroid/content/Context;

    const/4 p2, 0x2 //p2赋值2

    const-string p3, "level" //sp的索引

    invoke-virtual {p0, p1, p3, p2}, Lcom/zj/wuaipojie/util/SPUtils;-
>saveInt(Landroid/content/Context;Ljava/lang/String;I)V //写入数据

    goto :goto_50 //跳转地址

    :cond_43

    check-cast p1, Landroid/content/Context;

    const-string p0, "\u8bf7\u5148\u5145\u503c\u5927\u4f1a\u5458\u54e6\uff01" //
请先充值大会员哦!

    check-cast p0, Ljava/lang/CharSequence;

    invoke-static {p1, p0, p5}, Landroid/widget/Toast;-
>makeText(Landroid/content/Context;Ljava/lang/CharSequence;I)Landroid/widget/Toas
t;

    move-result-object p0

    invoke-virtual {p0}, Landroid/widget/Toast;->show()V

    :goto_50
    return p5  //返回p5的值
.end method //方法结束

//判断是否是大会员的方法
.method public final isvip()Z
    .registers 2

    const/4 v0, 0x0 //v0赋值0

    return v0 //返回v0的值

.end method
```

思路



根据代码分析（此处可以使用ai分析）

第一步判断是否是10个硬币，第二部判断是否是大会员

第一种方法

1、vip修改

（1）修改判断

这一步判断是10个金币，修改ge为le改为相反的逻辑

if-ge	全称greater equal(a>=b)，a大于或等于则跳
if-le	全称little equal(a<=b)，a小于或等于则跳

接下来跳转到这

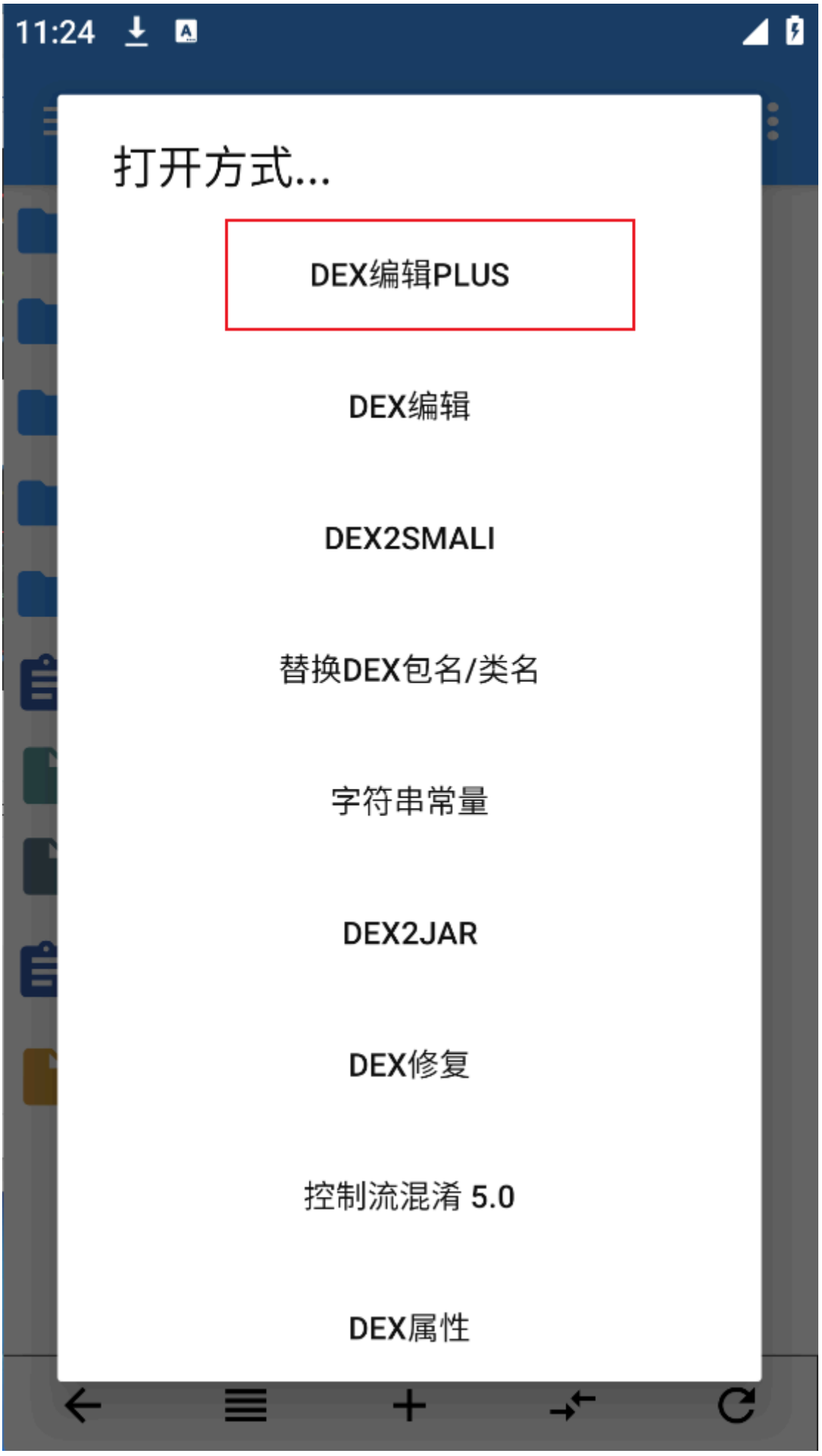
```
check-cast v0, Ljava/lang/CharSequence;

invoke-static {p0, v0, p5}, Landroid/widget/Toast;->makeText(Landroid/content/Context;Ljava/lang/CharS

move-result-object p0

invoke-virtual {p0}, Landroid/widget/Toast;->show()V

.line 36
:cond_15
invoke-virtual {p1}, Lcom/zj/wuaipojie/ui/ChallengeSecond;->isvip()Z

move-result p0

if-eqz p0, :cond_43

.line 37
check-cast p1, Landroid/content/Context;

const-string p0, "\u5f53\u524d\u5df2\u7ecf\u662f\u5927\u4f1a\u5458\u4e86\u54e6\uff01"

check-cast p0, Ljava/lang/CharSequence;
```

```
invoke-virtual {p0}, Landroid/widget/Toast;->show()V  //当看到这个Toast->show的就已经成功了不过至是弹窗代码

goto :goto_31  //跳转到:goto_31

:cond_15 //跳转的一个地址

invoke-virtual {p1}, Lcom/zj/wuaipojie/ui/ChallengeSecond;->isvip()Z  //判断isvip方法的返回值是否为真(则结果是否为1

move-result p0  //结果赋值给p0

if-eqz p0, :cond_43 //如果结果为0则跳转cond_43地址

const p0, 0x7f0d0018  //在arsc中的id索引，这个值可以进行查询

.line 37
invoke-virtual {p2, p0}, Landroid/widget/ImageView;->setImageResource(I)V //设置图片资源

const p0, 0x7f0d0008
```

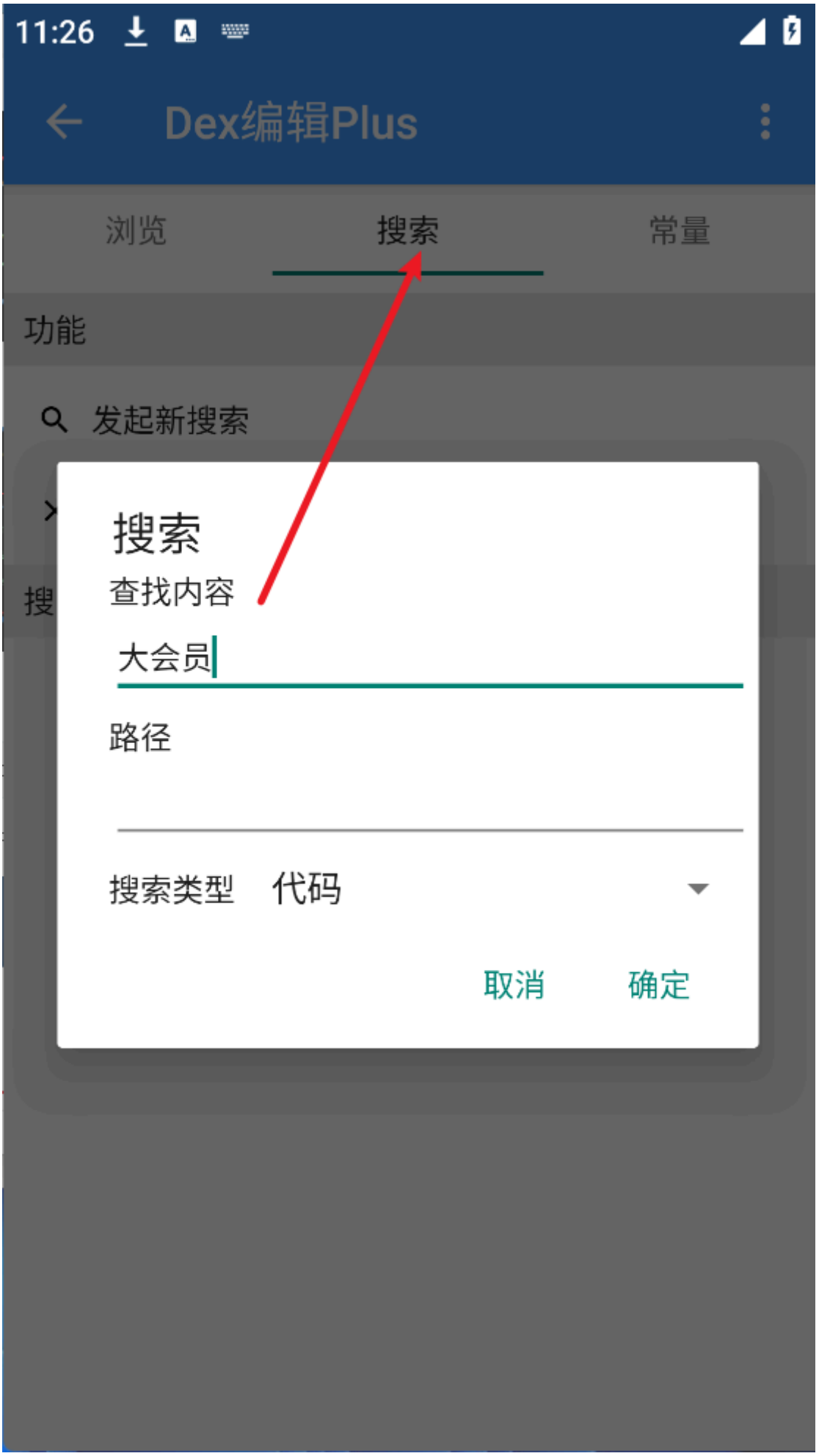
注释掉

```
# if-eqz p0, :cond_43 //如果结果为0则跳转cond_43地址
```

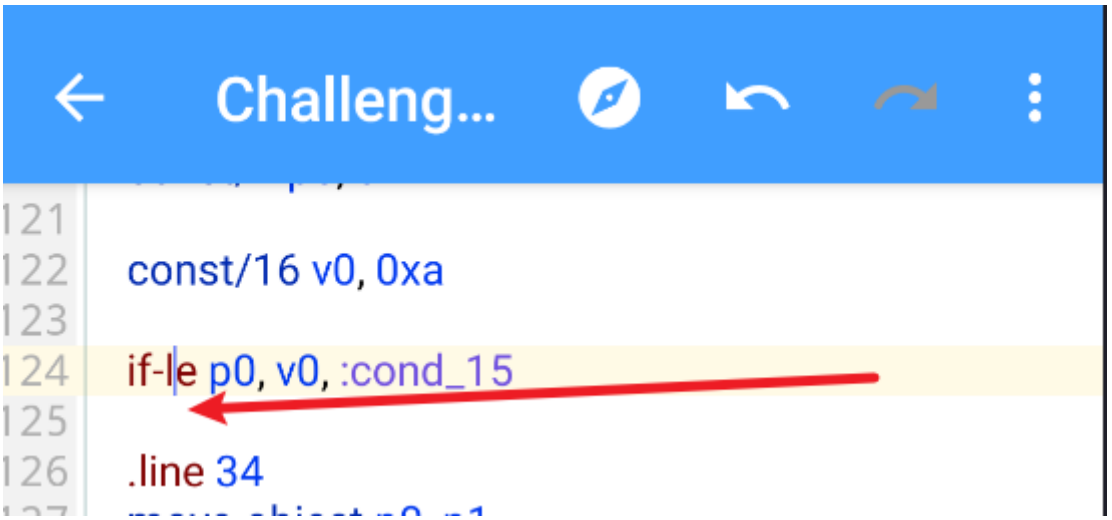
使用np管理器修改（模拟器断网），提取安装包，打开文件，打开dex文件

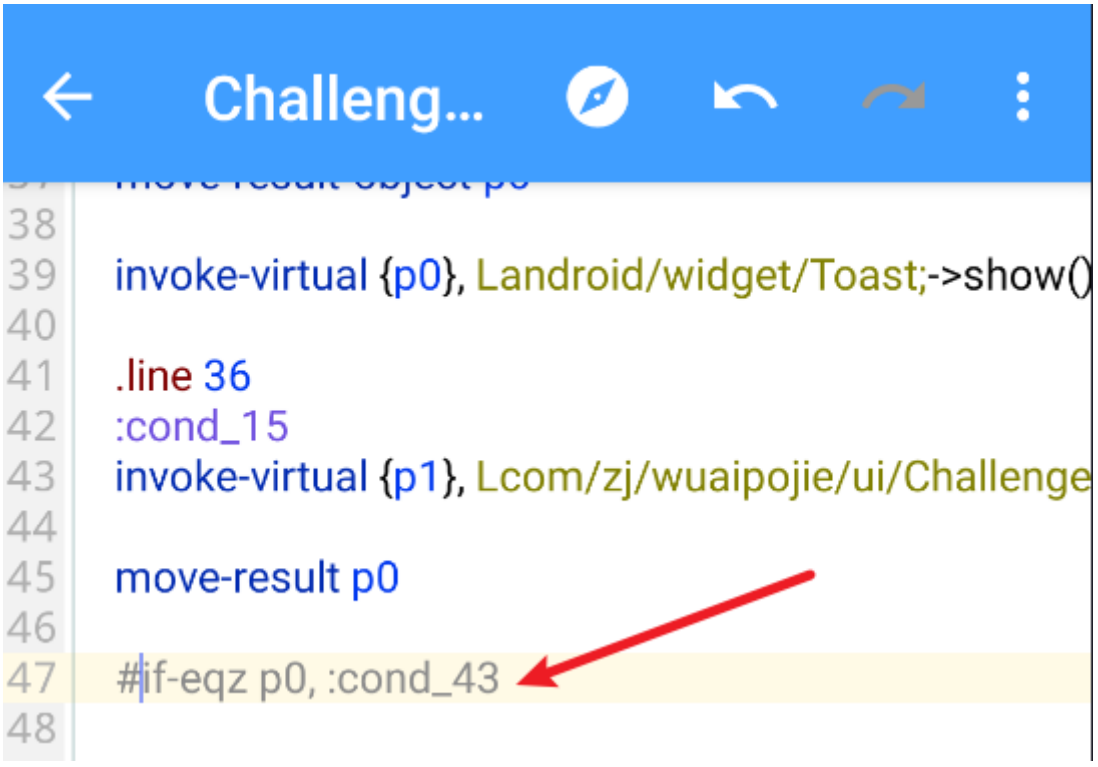

11:24
打开方式...
DEX编辑PLUS
DEX编辑
DEX2SMALI
替换DEX包名/类名
字符串常量
DEX2JAR
DEX修复
控制流混淆 5.0
DEX属性

搜索



修改


Challeng...
121
122 const/16 v0, 0xa
123
124 if-le p0, v0, :cond_15
125
126 .line 34


Challeng...
38
39 invoke-virtual {p0}, Landroid/widget/Toast;->show()
40
41 .line 36
42 :cond_15
43 invoke-virtual {p1}, Lcom/zj/wuaipojie/ui/Challenge
44
45 move-result p0
46
47 #if-eqz p0, :cond_43
48

然后保存，安装

wuaipojie

Tip:请收集硬币并完成一键三连

硬币数量: 1 获取硬币

一键三连

当前已经是大会员了哦!

（2）修改寄存器的值

```
end method

method private static final onCreate$lambda-2(Lkotlin/jvm/internal/Ref$IntRef;Lcom/zj/wuaipojie/ui/Challe
    .registers 7

    .line 33
    iget p0, p0, Lkotlin/jvm/internal/Ref$IntRef;->element:I

    const/4 p5, 0x1

    const/16 v0, 0xa

    if-ge p0, v0, :cond_15
```

```
    .line 33  //代码所在的行数
    iget p0, p0, Lkotlin/jvm/internal/Ref$IntRef;->element:I  //读取p0(第一个参数，参考寄存器知识)中element的值赋值给p0

    const/4 p5, 0x1  //p5赋值1

    const/16 v0, 0xa //v0赋值10，在16进制里a表示10

    if-ge p0, v0, :cond_15  //判断p0的值是否大于或等于v0的值(即p0的值是否大于或等于10)，如果大于或等于则跳转到:cond_15

    .line 34  //以下是常见的Toast弹窗代码
    check-cast p1, Landroid/content/Context; //检查Context对象引用
```

将const/16 v0, 0xa改为

```
const/16 v0, 0x0 #数值为0
```

```
# virtual methods
.method public final isvip()Z
    .registers 2

    const/4 v0, 0x0

    return v0
.end method

.method protected onCreate(Landroid/os/Bundle;)V
    .registers 9

    .line 16
    invoke-super {p0, p1}, Landroidx/appcompat/app/AppCompatActivity;->onCreate(Landroid/os/Bundle;)V

    const p1, 0x7f0b001f
```

```
    invoke-virtual {p0}, Landroid/widget/Toast;->show()V

    :goto_50
    return p5  //返回p5的值
.end method //方法结束

//判断是否是大会员的方法
.method public final isvip()Z
    .registers 2

    const/4 v0, 0x0 //v0赋值0

    return v0 //返回v0的值

.end method
```

此处吧const/4 v0, 0x0 //v0赋值0

改为

```
const/4 v0, 0x1 //v0赋值1
```

使用np管理器修改，保存，安装

其他--按钮快速定位

1、打开开发者助手app，打开待测试的app

界面资源分析，左侧-虫子图标，然后点击界面资源分析

当前应用
最近安装
抓包
LOG查看器
wuaipojie
未知加固
包名 ： com.zj.wuaipojie
版本号：1
版本 ：1.0
Tip:请收集硬币并完成一键三连
当前Activity:
com.zj.wuaipojie.ui.ChallengeSecond
当前Fragment:
-> ReportFragment
硬币数量:
0
获取硬币
界面资源分析:
点击开始
PID | UID:
PID:-1 UID:10080
查看数据库:
/data/user/0/com.zj.wuaipojie/databases
停止应用
重启应用
清除数据
卸载应用

长按复制

0x7F08007B

之后进入MT管理器进行dex文件编辑 搜索-全局搜索

11:56
Dex编辑器++
classes.dex
浏览
最近
搜索
常量
功能
搜索
查找内容
0x7F08007B
路径
/
搜索类型 整数
搜索子目录
十六进制
取消
确定
开发者助手 已被授予超级用户权限

（3）强制跳转

goto	强制跳到指定位置

源代码

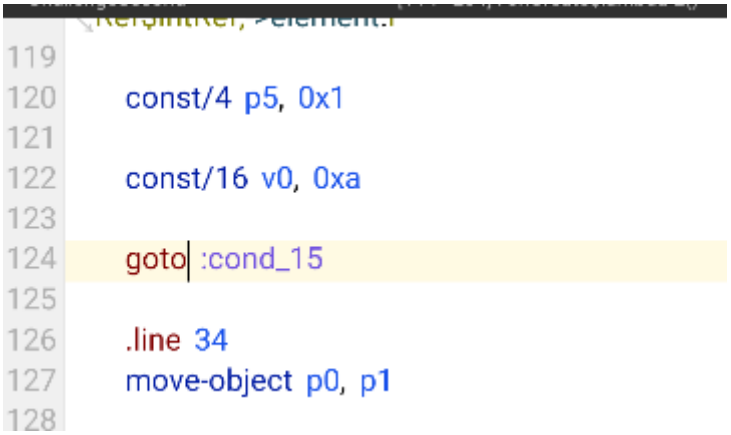
```
    const/4 p5, 0x1  //p5赋值1

    const/16 v0, 0xa //v0赋值10，在16进制里a表示10

    if-ge p0, v0, :cond_15  //判断p0的值是否大于或等于v0的值(即p0的值是否大于或等于10)，如果大于或等于则跳转到:cond_15

    .line 34  //以下是常见的Toast弹窗代码
    check-cast p1, Landroid/content/Context; //检查Context对象引用

    const-string p0, "请先获取10个硬币哦" //弹窗文本信息，把""里的字符串数据赋值给p0
```

此处可以把

```
 if-ge p0, v0, :cond_15  //判断p0的值是否大于或等于v0的值(即p0的值是否大于或等于10)，如果大于或等于则跳转到:cond_15
```

改为

```
goto :cond_15
```



然后找到isvip

```
    invoke-virtual {p1}, Lcom/zj/wuaipojie/ui/ChallengeSecond;->isvip()Z  //判断isvip方法的返回值是否为真(即结果是否为1)

    move-result p0  //结果赋值给p0

    if-eqz p0, :cond_43 //如果结果为0则跳转cond_43地址

    const p0, 0x7f0d0018  //在arsc中的id索引，这个值可以进行查询
```

此处可以把注释掉 if-eqz p0, :cond_43

```
44
45      move-result p0
46
47      #if-eqz p0, :cond_43
48
49      .line 37
50      check-cast p1, Landroid/content/Context;
51
```

四、广告&弹窗静默卡

1、关闭广告

1.广告类型

启动广告　弹窗&更新广告　横幅广告

2.安卓四大组件

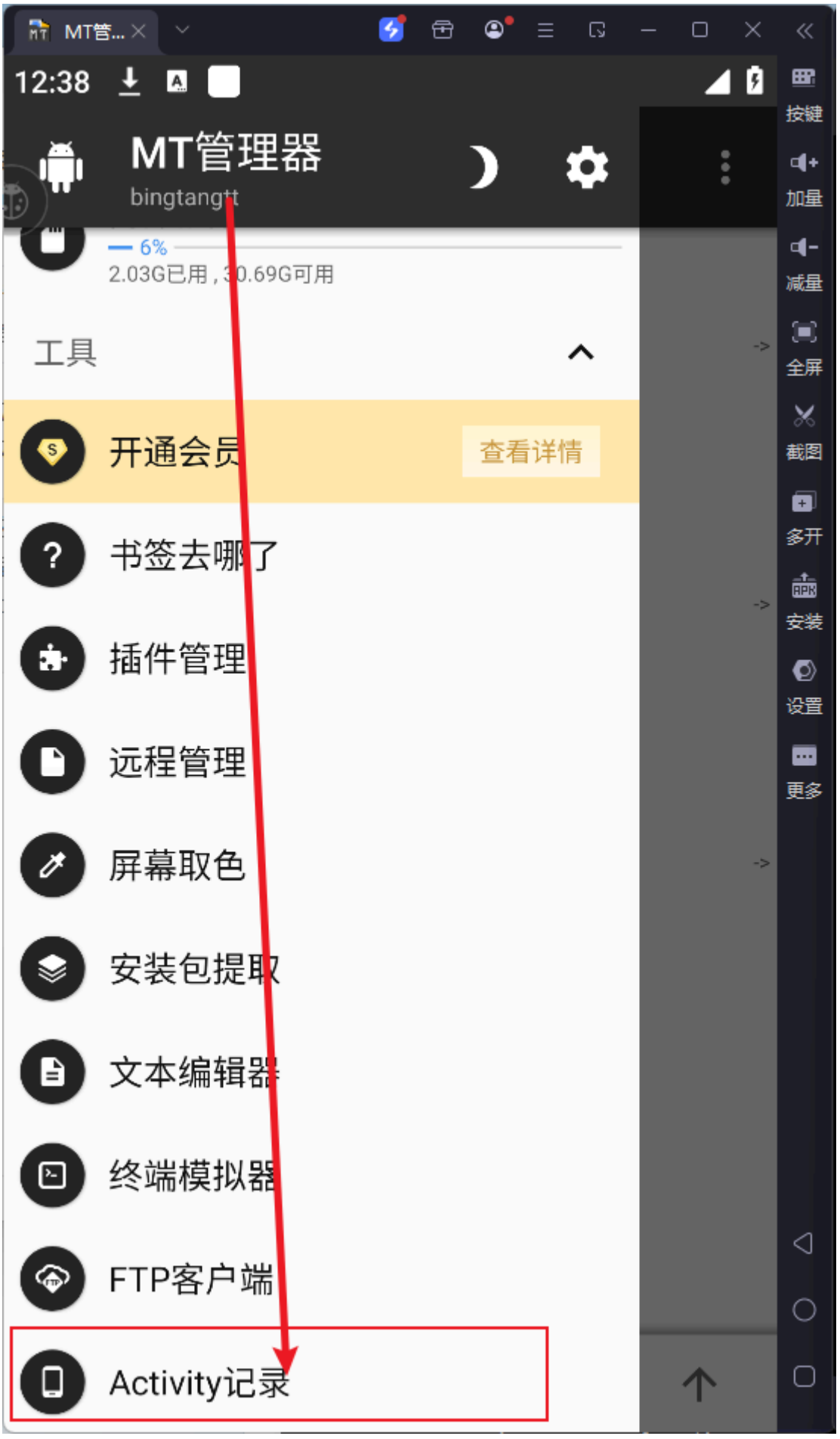
组件	描述
Activity(活动)	在应用中的一个Activity可以用来表示一个界面，意思可以理解为“活动”，即一个活动开始，代表 Activity组件启动，活动结束，代表一个Activity的生命周期结束。一个Android应用必须通过Activity来运行和启动，Activity的生命周期交给系统统一管理。
Service(服务)	Service它可以在后台执行长时间运行操作而没有用户界面的应用组件，不依赖任何用户界面，例如后台播放音乐，后台下载文件等。
Broadcast Receiver(广播接收器)	一个用于接收广播信息，并做出对应处理的组件。比如我们常见的系统广播：通知时区改变、电量低、用户改变了语言选项等。
Content Provider(内容提供者)	作为应用程序之间唯一的共享数据的途径，Content Provider主要的功能就是存储并检索数据以及向其他应用程序提供访问数据的接口。Android内置的许多数据都是使用Content Provider形式，供开发者调用的（如视频，音频，图片，通讯录等）

1.activity的切换

启动广告流程：
启动Activity->广告Activity->主页Activity
修改方法：
1.修改加载时间
2.Acitivity切换定位，修改Intent的Activity类名

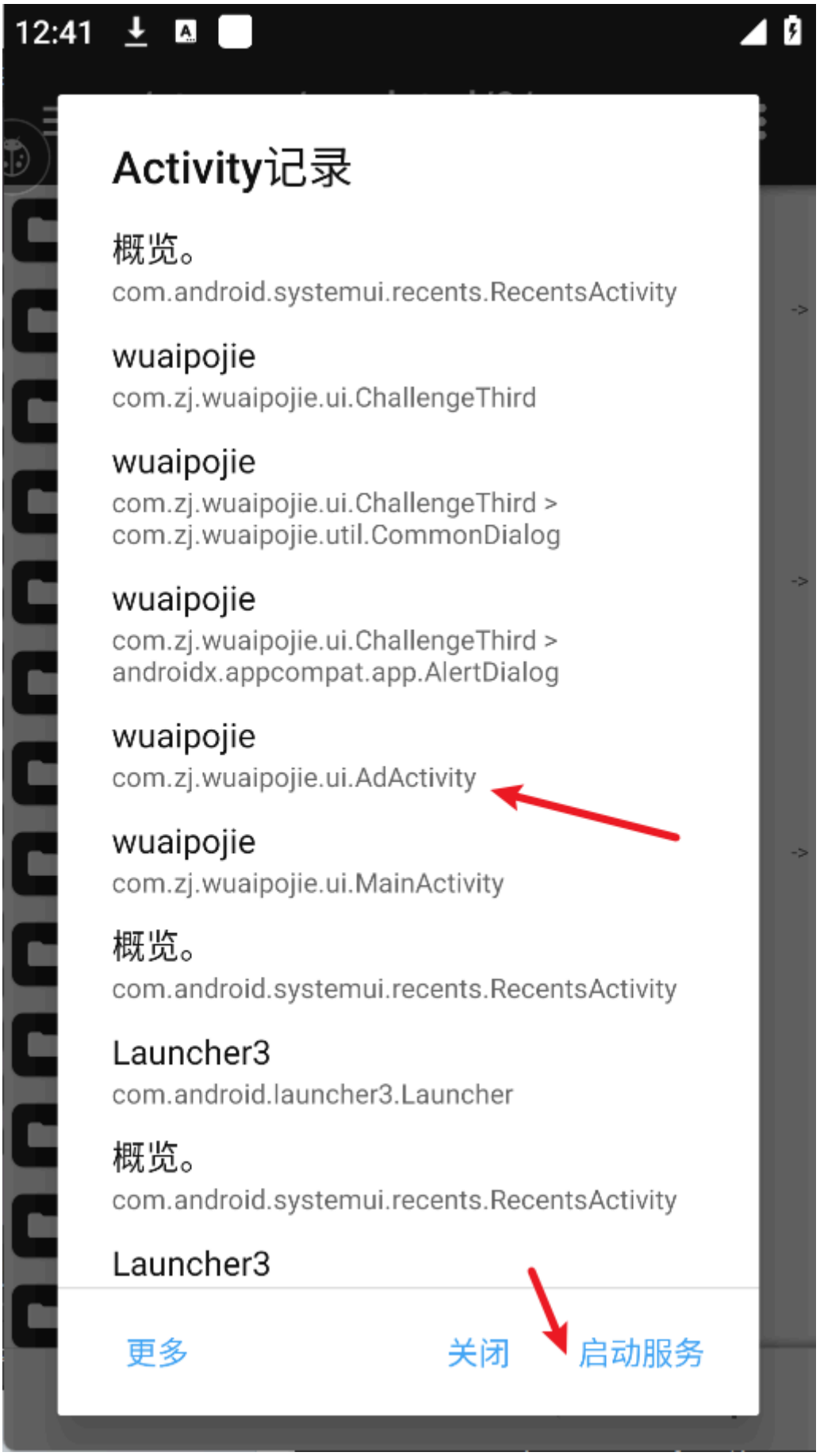
（1）修改广告加载时间为0

，打开mt管理器，左侧，然后启动



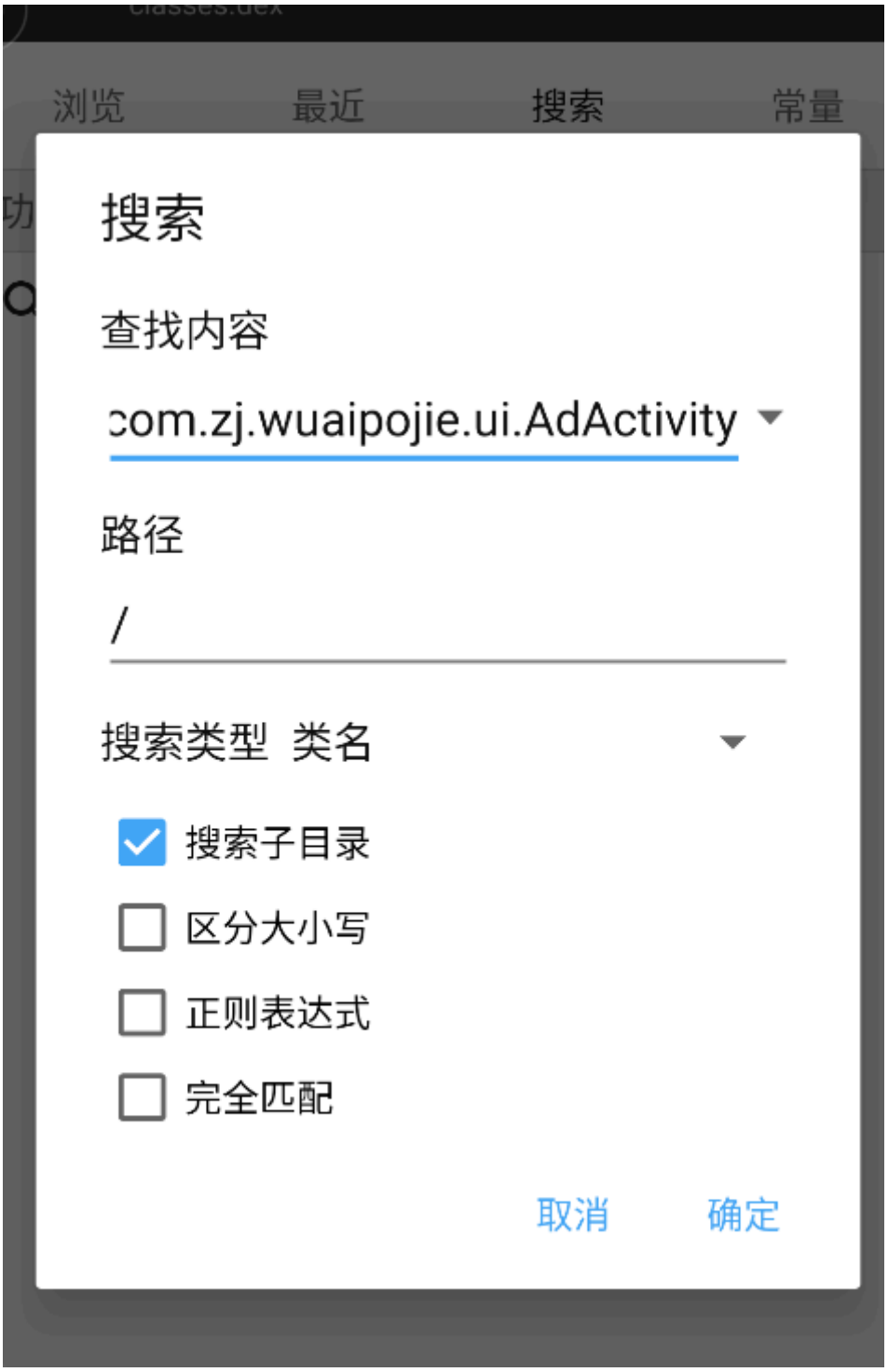
然后app放到后台运行，启动待测试的app

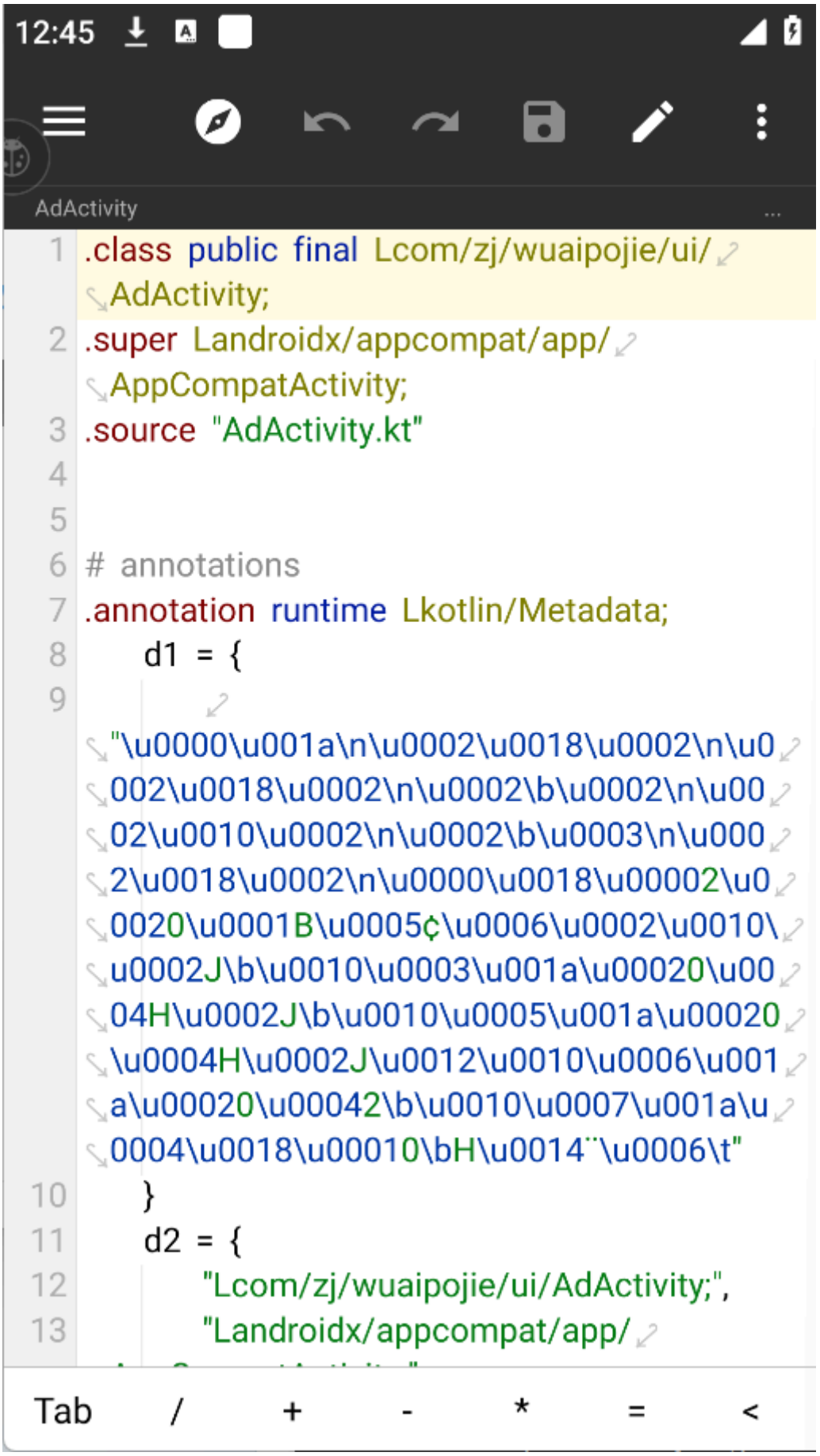
然后退出待测试的app，查看mt管理器的捕获，关闭服务,长按复制



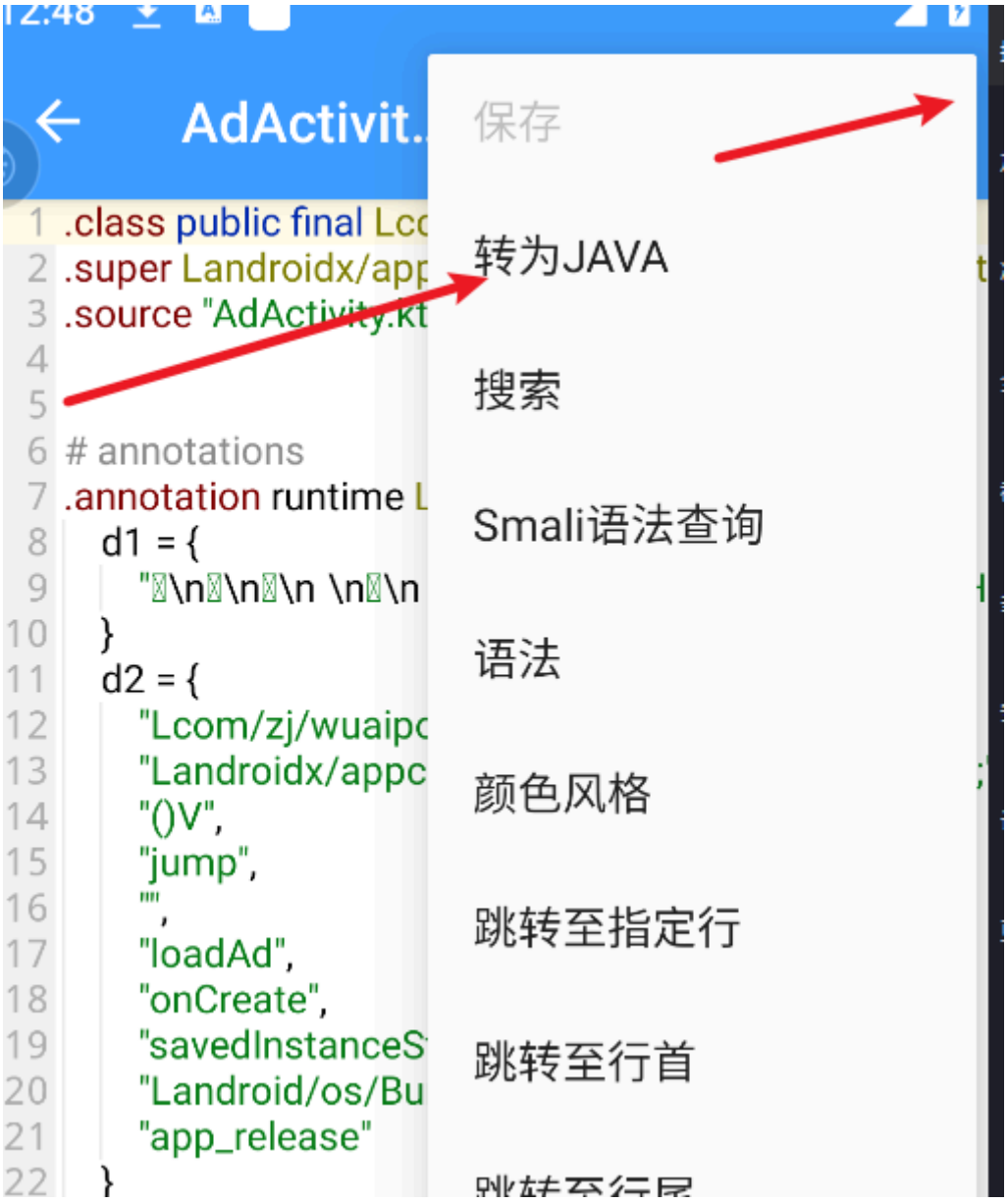
复制 com.zj.wuaipojie.ui.AdActivity，关闭

提取安装包，查看dex文件，使用dex编辑++打开，然后搜索，高级搜索


浏览
最近
搜索
常量
搜索
查找内容
com.zj.wuaipojie.ui.AdActivity
路径
/
搜索类型 类名
搜索子目录
区分大小写
正则表达式
完全匹配
取消
确定



mt管理器转成java需要会员，可以用np管理器，转成java


AdActivit..
保存
转为JAVA
搜索
Smali语法查询
语法
颜色风格
跳转至指定行
跳转至行首
.class public final Lcc
.super Landroidx/app
.source "AdActivity.kt
annotations
.annotation runtime L
d1 = {
}
d2 = {
"Lcom/zj/wuaipc
"Landroidx/appc
"()V",
"jump",
"loadAd",
"onCreate",
"savedInstanceS
"Landroid/os/Bu
"app_release"
}

AdActivity.java

```
//
// Decompiled by Jadx (from NP Manager)
//
package com.zj.wuaipojie.ui;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import androidx.appcompat.app.AppCompatActivity;
import com.zj.wuaipojie.ui.AdActivity$.ExternalSynthe
import kotlin.Metadata;

@Metadata(d1 = {"\u0000\u001a\n\u0002\u0018\u0(
/* compiled from: AdActivity.kt */
public final class AdActivity extends AppCompatActivi
    protected void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(2131427363);
        loadAd();
    }

    private final void jump() {
        startActivity(new Intent((Context) this, ChallengeT
        finish();
    }

    private final void loadAd() {
        new Handler().postDelayed(new ExternalSynthetic
    }

    /* renamed from: loadAd$lambda-0 */
    private static final void m0loadAd$lambda-0(AdActi
        adActivity.jump();
```

此处可以使用jadx查看代码，并将代码放到ai进行分析代码

```
package com.zj.wuaipojie.ui;

// 导入所需的类和接口
import android.content.Intent; // 用于在不同活动（Activity）之间进行跳转
import android.os.Bundle; // 用于保存和恢复活动状态
```

```
import android.os.Handler; // 用于处理延迟任务
import androidx.appcompat.app.AppCompatActivity; // 兼容库中的活动基类
import com.zj.wuaipojie.R; // 项目资源类
import kotlin.Metadata; // Kotlin元数据注解

/* 编译自: AdActivity.kt */
@Metadata( /* Kotlin元数据，描述类的结构、方法等 */ )
/* 从classes.dex加载 */
public final class AdActivity extends AppCompatActivity {

    // Kotlin生成的静态方法，用于lambda表达式调用
    public static /* synthetic */ void
$r8$lambda$Ba8g8Cpz5n6fbKN5g2ojikJ3IFo(AdActivity adActivity) {
        adActivity.jump(); // 调用jump方法
    }

    // 重写onCreate方法，这是活动生命周期的一部分
    @Override
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle); // 调用父类的onCreate方法
        setContentView(R.layout.ad_activity); // 设置活动的内容视图
        loadAd(); // 加载广告
    }

    // 定义一个公开的最终方法jump，用于跳转到另一个活动
    public final void jump() {
        startActivity(new Intent(this, ChallengeThird.class)); // 创建并启动一个新的
Intent，跳转到ChallengeThird活动
        finish(); // 结束当前活动
    }

    // 定义一个私有的最终方法loadAd，用于加载广告，并在延迟后跳转到另一个活动
    private final void loadAd() {
        new Handler().postDelayed(new Runnable() { // 创建一个新的Handler，并在延迟后
执行Runnable
            @Override
            public final void run() {

 AdActivity.$r8$lambda$Ba8g8Cpz5n6fbKN5g2ojikJ3IFo(AdActivity.this); // 通过Kotlin
生成的静态方法调用jump
            }
        }, 3000L); // 设置延迟时间为3000毫秒（3秒）
    }
}
```

```
    // 定义一个私有的最终方法loadAd，用于加载广告，并在延迟后跳转到另一个活动
    private final void loadAd() {
        new Handler().postDelayed(new Runnable() { // 创建一个新的Handler，并在延迟
后执行Runnable
            @Override
            public final void run() {

 AdActivity.$r8$lambda$Ba8g8Cpz5n6fbKN5g2ojikJ3IFo(AdActivity.this); // 通过
Kotlin生成的静态方法调用jump
            }
        }, 3000L); // 设置延迟时间为3000毫秒（3秒）
    }
}
```

在smali代码中找到loadAD()，ai分析

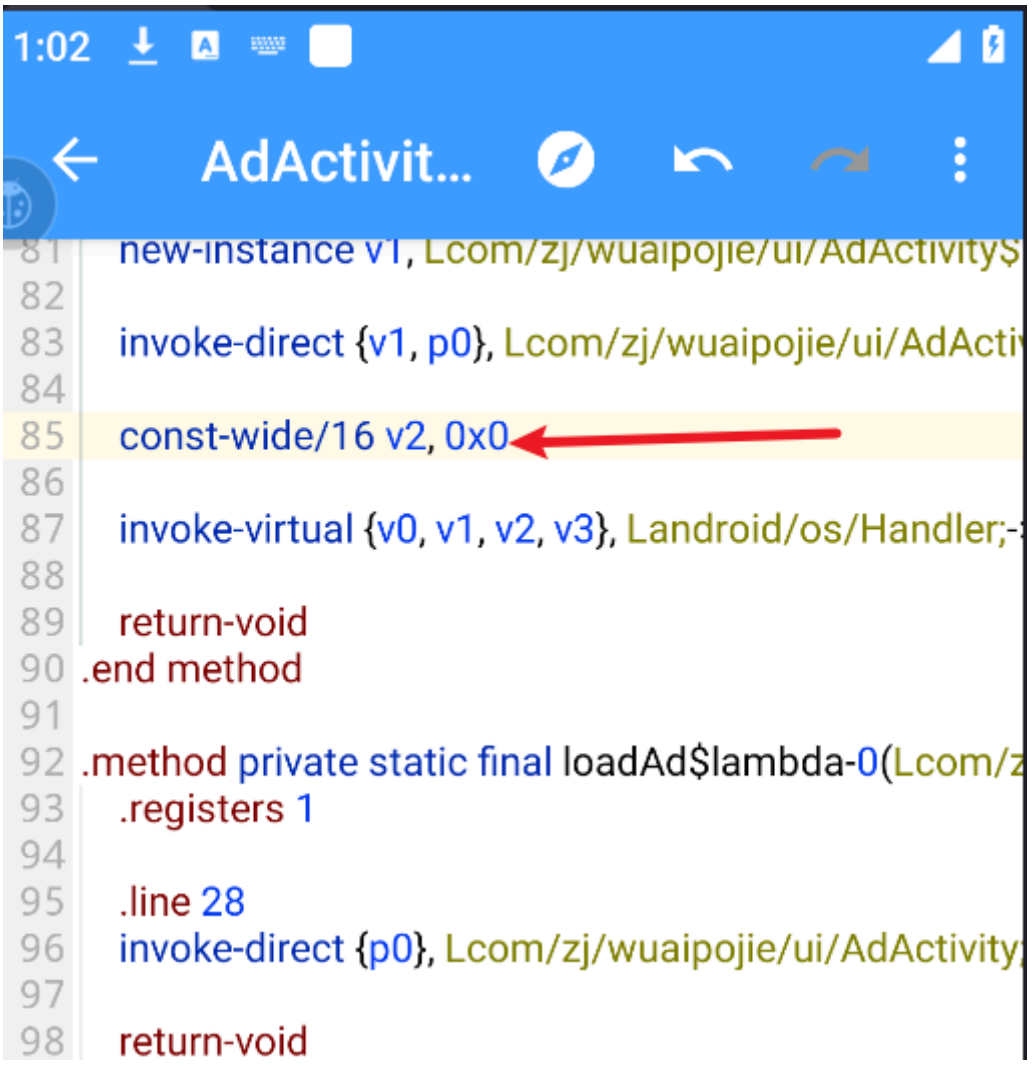
```
.method private final loadAd()V
    .registers 5  // 此方法需要5个寄存器

    .line 27  // 对应源代码的第27行
    new-instance v0, Landroid/os/Handler;  // 创建一个新的Handler实例，并将其引用存储在
寄存器v0中

    invoke-direct {v0}, Landroid/os/Handler;-><init>()V  // 调用Handler的无参构造函
数来初始化v0

    new-instance v1, Lcom/zj/wuaipojie/ui/AdActivity$$ExternalSyntheticLambda0;
 // 创建一个新的Lambda实例，并将其引用存储在寄存器v1中

    invoke-direct {v1, p0},
Lcom/zj/wuaipojie/ui/AdActivity$$ExternalSyntheticLambda0;-><init>
(Lcom/zj/wuaipojie/ui/AdActivity;)V  // 使用当前AdActivity实例初始化Lambda

    const-wide/16 v2, 0xbb8  // 将常量3000（十六进制bb8）加载到寄存器v2和v3中，作为延迟时
间

    invoke-virtual {v0, v1, v2, v3}, Landroid/os/Handler;-
>postDelayed(Ljava/lang/Runnable;J)Z  // 调用Handler的postDelayed方法，传入Runnable
和延迟时间

    return-void  // 方法返回，无返回值
.end method

.method private static final loadAd$lambda-0(Lcom/zj/wuaipojie/ui/AdActivity;)V
    .registers 1  // 此静态方法需要1个寄存器（除了参数寄存器p0）

    .line 28  // 对应源代码中Lambda表达式所在的行（可能不完全准确，因为Smali是字节码，不一定
直接对应源代码行）
    invoke-direct {p0}, Lcom/zj/wuaipojie/ui/AdActivity;->jump()V  // 调用
AdActivity的私有方法jump

    return-void  // 方法返回，无返回值
.end method
```

将

```
    const-wide/16 v2, 0xbb8  // 将常量3000（十六进制bb8）加载到寄存器v2和v3中，作为延迟时间
```

改为

```
    const-wide/16 v2, 0x0 //将广告时间改为0
```



转成java可以看到时间为0

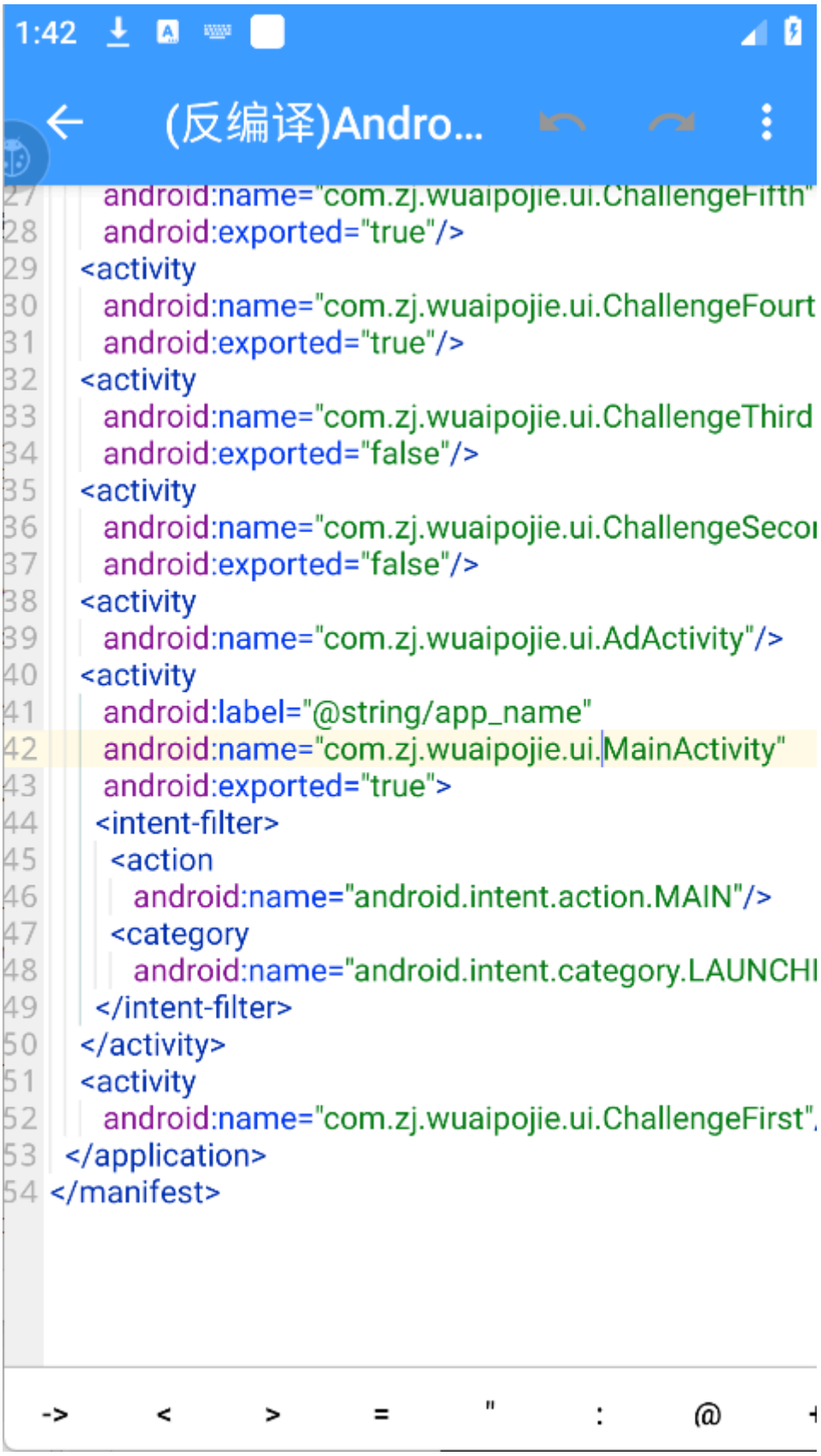
保存-退出-安装，直接取消时间限制

（2）修改xml文件 Activity类名

Acitivity切换定位，修改Intent的Activity类名**

（不推荐使用）

mt管理器查看xml文件（np管理器使用此方法会报错）

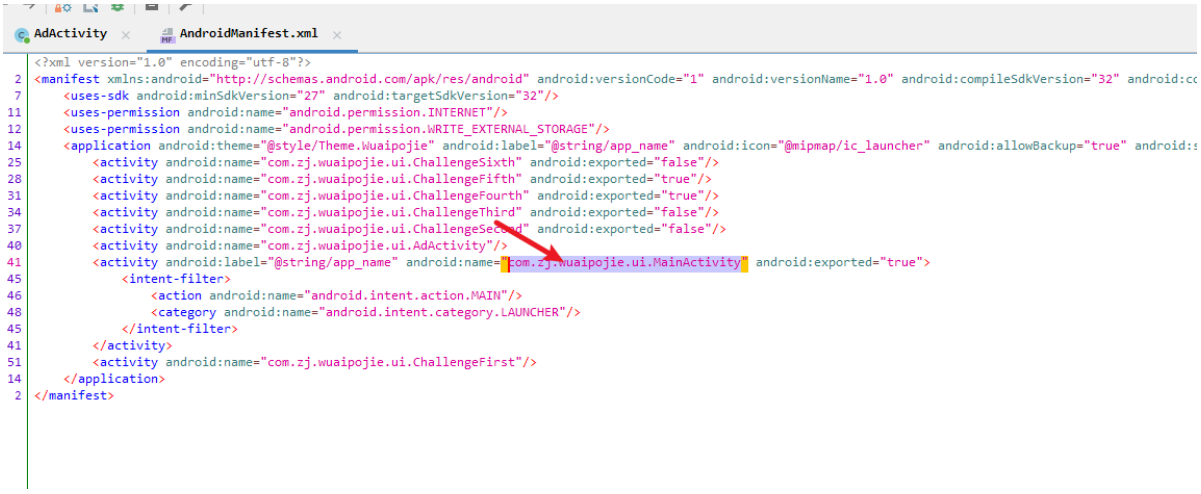


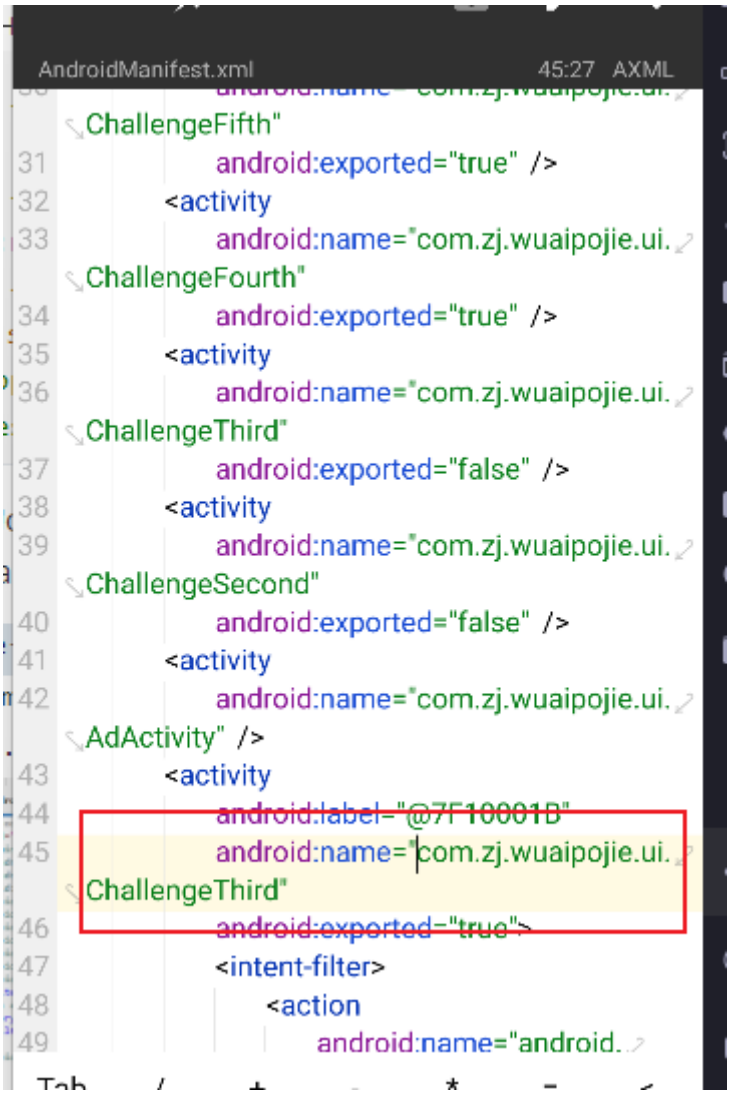

```
<?xml version="1.0" encoding="utf-8"?>
<!-- XML声明，指定文档版本为1.0，编码方式为utf-8 -->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    <!-- 应用版本代码，用于区分应用的不同版本，整数类型 -->
    android:versionName="1.0"
    <!-- 应用版本名称，展示给用户的版本信息 -->
    android:compileSdkVersion="32"
    <!-- 编译SDK版本，指定应用编译时使用的Android SDK版本 -->
```

```
    android:compileSdkVersionCodename="12"
    <!-- 编译SDK版本的代号，通常是Android的版本名，如"12"对应Android 12 -->
    package="com.zj.wuaipojie"
    <!-- 应用包名，唯一标识一个应用 -->
    platformBuildVersionCode="32"
    <!-- 平台构建版本代码，表示应用针对的Android平台版本 -->
    platformBuildVersionName="12">
    <!-- 平台构建版本名称，对应Android的版本名 -->

    <uses-sdk android:minSdkVersion="27"
        <!-- 最小支持SDK版本，应用能在此版本及以上的Android设备上运行 -->
        android:targetSdkVersion="32"/>
        <!-- 目标SDK版本，推荐应用针对此版本进行优化 -->

    <uses-permission android:name="android.permission.INTERNET"/>
    <!-- 请求互联网访问权限，允许应用访问网络 -->

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <!-- 请求写入外部存储权限，允许应用写入外部存储设备 -->

    <application android:theme="@style/Theme.Wuaipojie"
        <!-- 应用主题，指定应用使用的主题样式 -->
        android:label="@string/app_name"
        <!-- 应用标签，通常是应用名称，引用string资源 -->
        android:icon="@mipmap/ic_launcher"
        <!-- 应用图标，指定应用图标资源 -->
        android:allowBackup="true"
        <!-- 是否允许应用数据备份和恢复 -->
        android:supportsRtl="true"
        <!-- 是否支持从右到左的布局，主要用于支持阿拉伯语等语言 -->
        android:extractNativeLibs="false"
        <!-- 是否在APK安装时提取本地库文件 -->
        android:fullBackupContent="@xml/backup_rules"
        <!-- 完全备份内容的规则，引用xml资源 -->
        android:networkSecurityConfig="@xml/network_config"
        <!-- 网络安全配置，引用xml资源 -->
        android:appComponentFactory="androidx.core.app.CoreComponentFactory"
        <!-- 应用组件工厂，用于实例化应用组件，如Activity等 -->
        android:dataExtractionRules="@xml/data_extraction_rules">
        <!-- 数据提取规则，引用xml资源 -->

        <activity android:name="com.zj.wuaipojie.ui.ChallengeSixth"
            <!-- 定义一个Activity组件，指定其类名 -->
            android:exported="false"/>
            <!-- 该Activity是否对其他应用可见，false表示不可见 -->

        <activity android:name="com.zj.wuaipojie.ui.ChallengeFifth"
            <!-- 定义另一个Activity组件 -->
            android:exported="true"/>
            <!-- 该Activity对其他应用可见 -->

        <activity android:name="com.zj.wuaipojie.ui.ChallengeFourth"
            <!-- 定义另一个Activity组件 -->
            android:exported="true"/>
            <!-- 该Activity对其他应用可见 -->
```

```
        <activity android:name="com.zj.wuaipojie.ui.ChallengeThird"
            <!-- 定义另一个Activity组件 -->
            android:exported="false"/>
            <!-- 该Activity不可见 -->

        <activity android:name="com.zj.wuaipojie.ui.ChallengeSecond"
            <!-- 定义另一个Activity组件 -->
            android:exported="false"/>
            <!-- 该Activity不可见 -->

        <activity android:name="com.zj.wuaipojie.ui.AdActivity"/>
        <!-- 定义广告Activity组件，未指定exported属性，默认为false -->

        <activity android:label="@string/app_name"
            <!-- Activity标签，引用string资源作为名称 -->
            android:name="com.zj.wuaipojie.ui.MainActivity"
            <!-- 主Activity组件的类名 -->
            android:exported="true">
            <!-- 主Activity对其他应用可见 -->
            <intent-filter>
                <!-- 指定该Activity为应用入口，响应MAIN动作和LAUNCHER类别 -->
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>

        <activity android:name="com.zj.wuaipojie.ui.ChallengeFirst"/>
        <!-- 定义另一个Activity组件，未指定exported属性，默认为false -->
    </application>
</manifest>
```

将第三关的com.zj.wuaipojie.ui.ChallengeThird覆盖com.zj.wuaipojie.ui.MainActivity

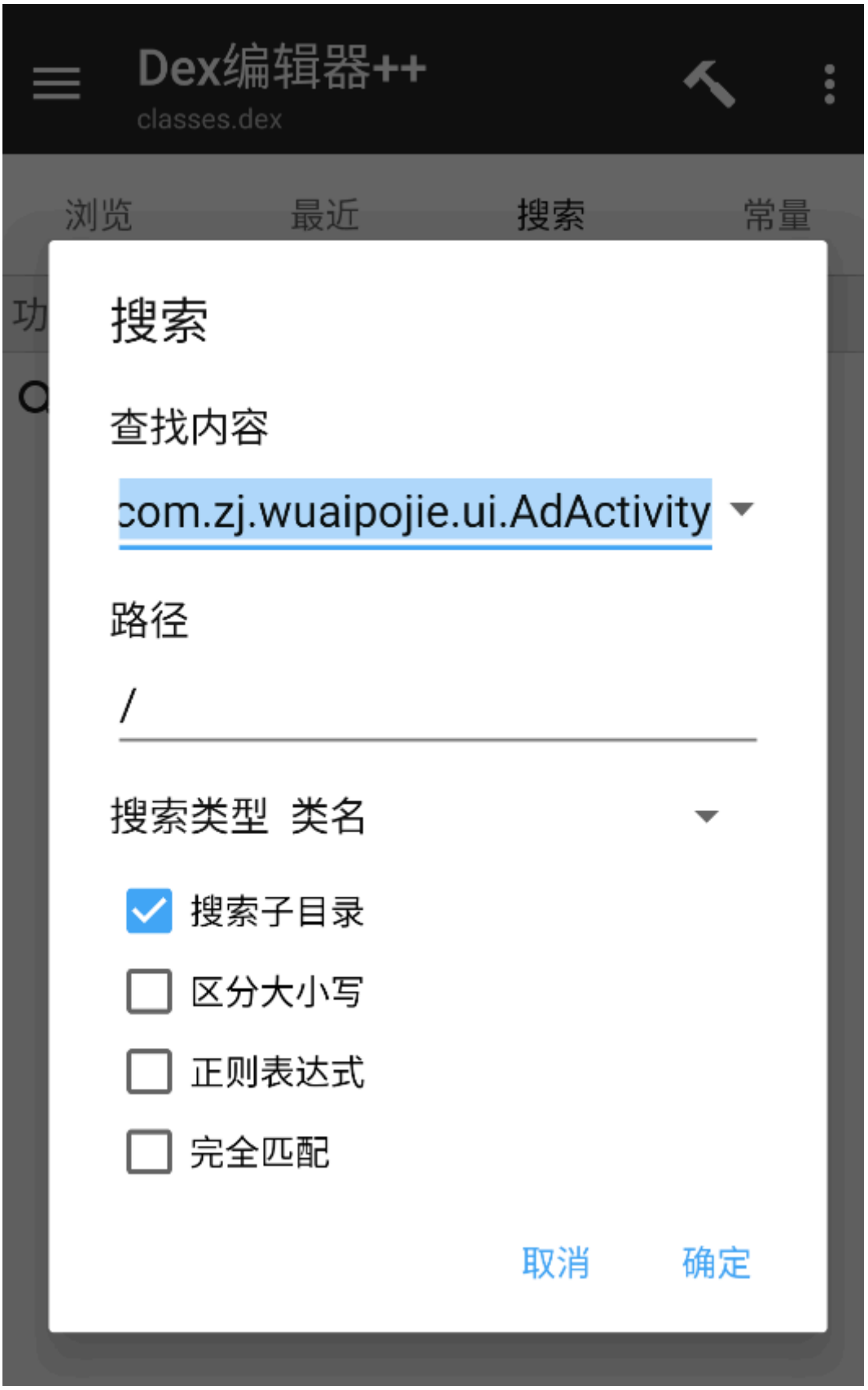


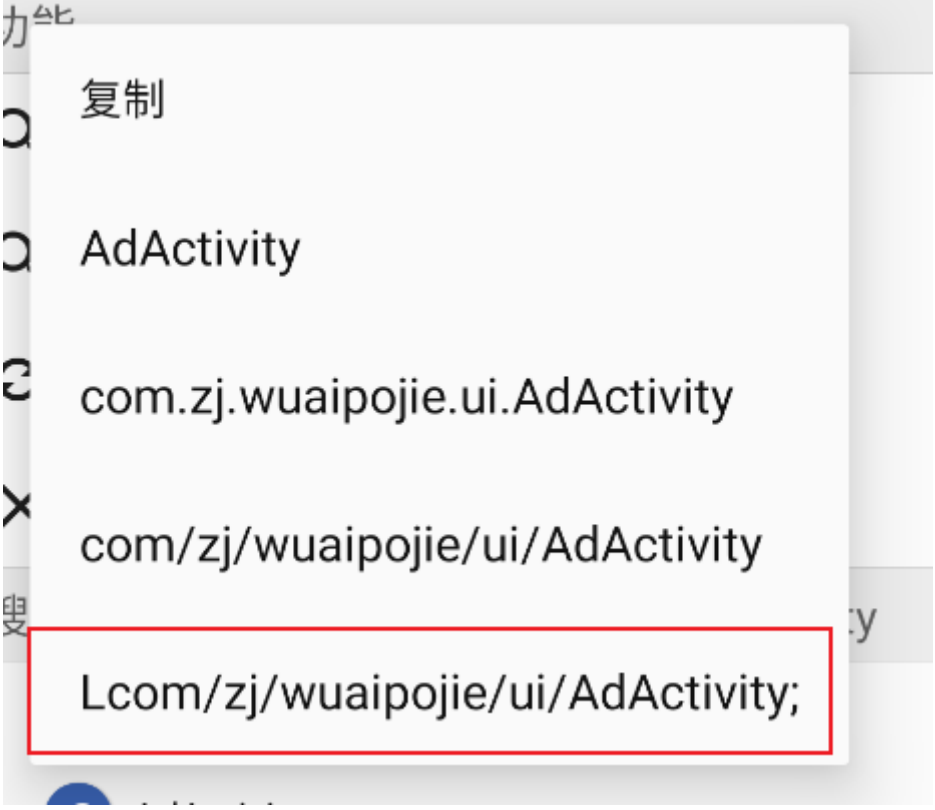


保存，安装，但是此时进去直接是第三关

（3） 修改dex文件中Activity类名

mt管理器打开dex文件，索广告


Dex编辑器++
classes.dex
浏览
最近
搜索
常量
搜索
查找内容
com.zj.wuaipojie.ui.AdActivity
路径
/
搜索类型 类名
搜索子目录
区分大小写
正则表达式
完全匹配
取消
确定



长按复制这个，搜索这个代码

搜索
查找内容
m/zj/wuaipojie/ui/AdActivity;
路径
/
搜索类型 代码
搜索子目录
区分大小写
正则表达式
取消
确定

此处进行了调用

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(19) - Lcom/zj/wuaipojie/ui/AdActivity;

com.zj.wuaipojie.ui

com.zj.wuaipojie.ui.Adapter

ChallengeAdapter

const-class v1, Lcom/zj/wuaipojie/ui/AdActivity;

```
allengeAdapter                    [98 - 274] : onCreateViewHolder$la...
Intent;)V

    goto :goto_9a

    .line 39
    :cond_59
    new-instance p0, Landroid/content/Intent;

    invoke-direct {p0}, Landroid/content/
Intent;-><init>()V

    .line 40
    invoke-virtual {p1}, Landroid/view/View;->
getContext()Landroid/content/Context;

    move-result-object v0

    const-class v1, Lcom/zj/wuaipojie/ui/
AdActivity;

    invoke-virtual {p0, v0, v1}, Landroid/
content/Intent;->setClass(Landroid/content/
Context;Ljava/lang/Class;)Landroid/
content/Intent;
```

这些代码可以使用np管理器转化为java代码

```java
// 根据position的值选择执行哪个case块
switch (position) {
    // 当position为0时执行以下代码
    case 0:
        // 创建一个新的Intent对象
        Intent intent = new Intent();
        // 设置Intent要启动的Activity类为ChallengeFirst
        intent.setClass(it.getContext(), ChallengeFirst.class);
        // 使用上下文启动Intent指向的Activity
        it.getContext().startActivity(intent);
        // 结束当前方法的执行
        return;

    // 当position为1时执行以下代码
    case 1:
        // 创建一个新的Intent对象
        Intent intent2 = new Intent();
        // 设置Intent要启动的Activity类为ChallengeSecond
        intent2.setClass(it.getContext(), ChallengeSecond.class);
```

```
        // 使用上下文启动Intent指向的Activity
        it.getContext().startActivity(intent2);
        // 结束当前方法的执行
        return;

    // 当position为2时执行以下代码
    case 2:
        // 创建一个新的Intent对象
        Intent intent3 = new Intent();  // new一个Intent实例
        // 设置Intent要启动的Activity类为AdActivity
        intent3.setClass(it.getContext(), AdActivity.class);  // 传入要切换的
Activity的类名
        // 使用上下文启动Intent指向的Activity
        it.getContext().startActivity(intent3);  // 启动对应的Activity
        // 结束当前方法的执行
        return;

    // 当position为3时执行以下代码
    case 3:
        // 创建一个新的Intent对象
        Intent intent4 = new Intent();
        // 设置Intent要启动的Activity类为ChallengeFourth
        intent4.setClass(it.getContext(), ChallengeFourth.class);
        // 使用上下文启动Intent指向的Activity
        it.getContext().startActivity(intent4);
        // 结束当前方法的执行
        return;

    // 当position不是上述任何一个值时执行以下代码
    default:
        // 直接结束方法的执行，不做任何操作
        return;
}
```

查看smali代码

将AdActivity广告改为第三关的类型类名ChallengeThird

```
211     move-result-object v0
212
213     const-class v1, Lcom/zj/wuaipojie/ui/
        AdActivity;
214
```

改为

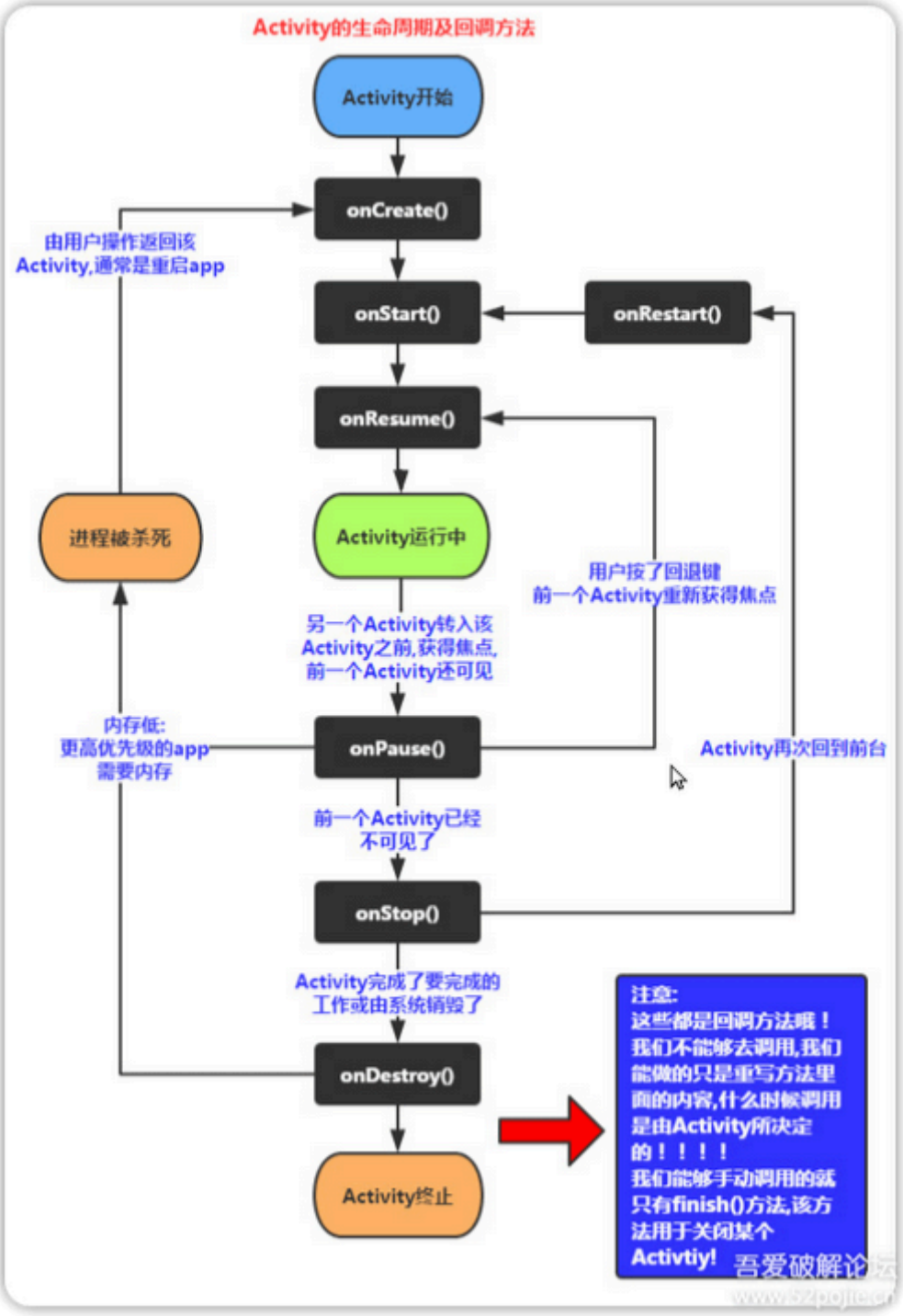
```
211     move-result-object v0
212
213     const-class v1, Lcom/zj/wuaipojie/ui/
        ChallengeThird;
214
215     invoke-virtual {p0, v0, v1}, Landroid/
        content/Intent;->setClass(Landroid/content/
```

保存 安装

2、关闭弹窗

Activity生命周期

函数名称	描述
onCreate()	一个Activity启动后第一个被调用的函数，常用来在此方法中进行Activity的一些初始化操作。例如创建View，绑定数据，注册监听，加载参数等。
onStart()	当Activity显示在屏幕上时，此方法被调用但此时还无法进行与用户的交互操作。
onResume()	这个方法在onStart()之后调用，也就是在Activity准备好与用户进行交互的时候调用，此时的Activity一定位于Activity栈顶，处于运行状态。
onPause()	这个方法是在系统准备去启动或者恢复另外一个Activity的时候调用，通常在这个方法中执行一些释放资源的方法，以及保存一些关键数据。
onStop()	这个方法是在Activity完全不可见的时候调用的。
onDestroy()	这个方法在Activity销毁之前调用，之后Activity的状态为销毁状态。
onRestart()	当Activity从停止stop状态恢进入start状态时调用状态。



4.弹窗定位&堆栈分析

修改方法:

1.修改xml中的versiocode

2.Hook弹窗(推荐算法助手开启弹窗定位)

3.修改dex弹窗代码

4.抓包修改响应体(也可以路由器拦截)

（1）关闭更新弹窗

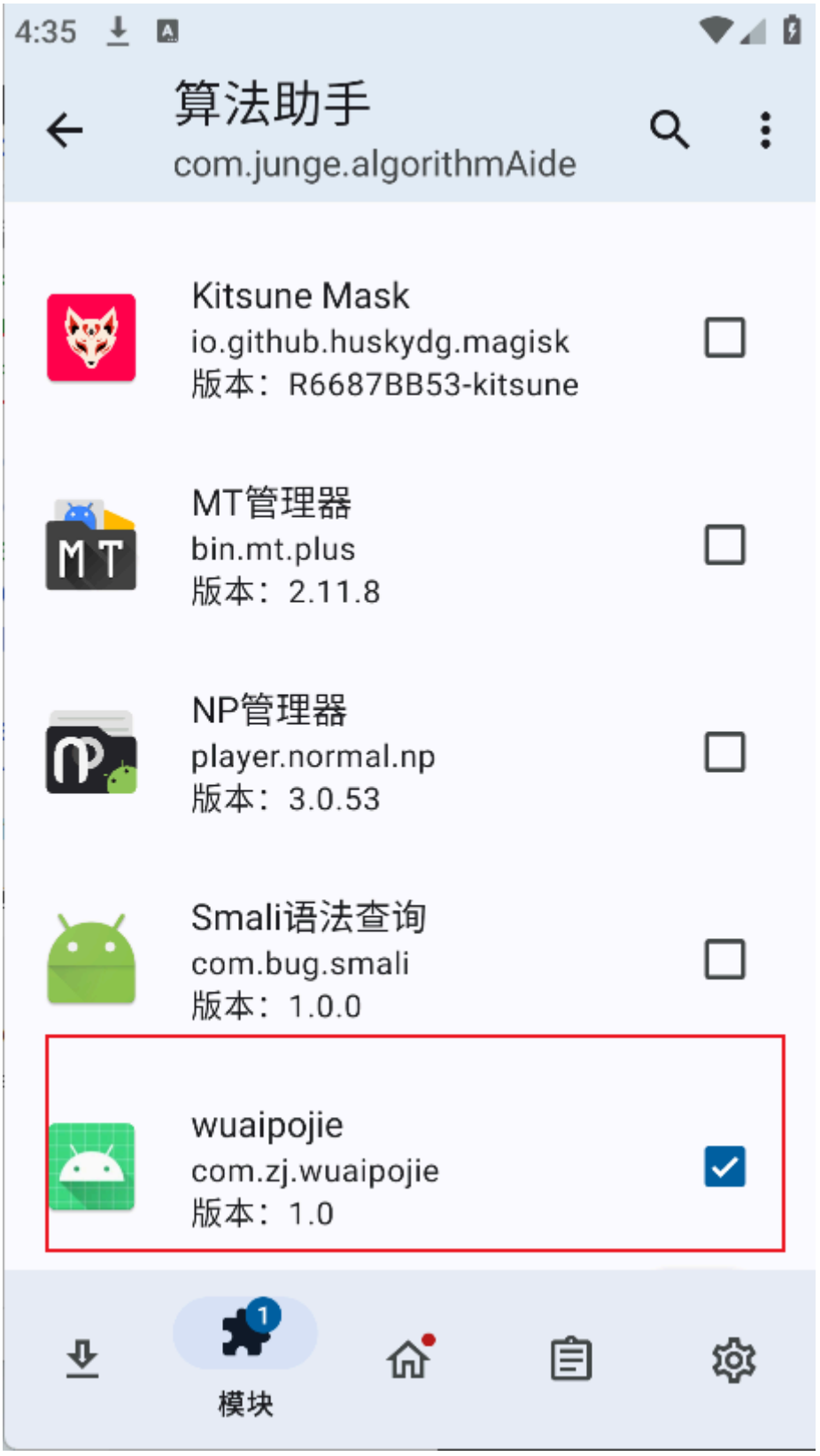
（不一定百分百好用）

将xml文件的版本号versioncode=1的版本号改为需要的版本号

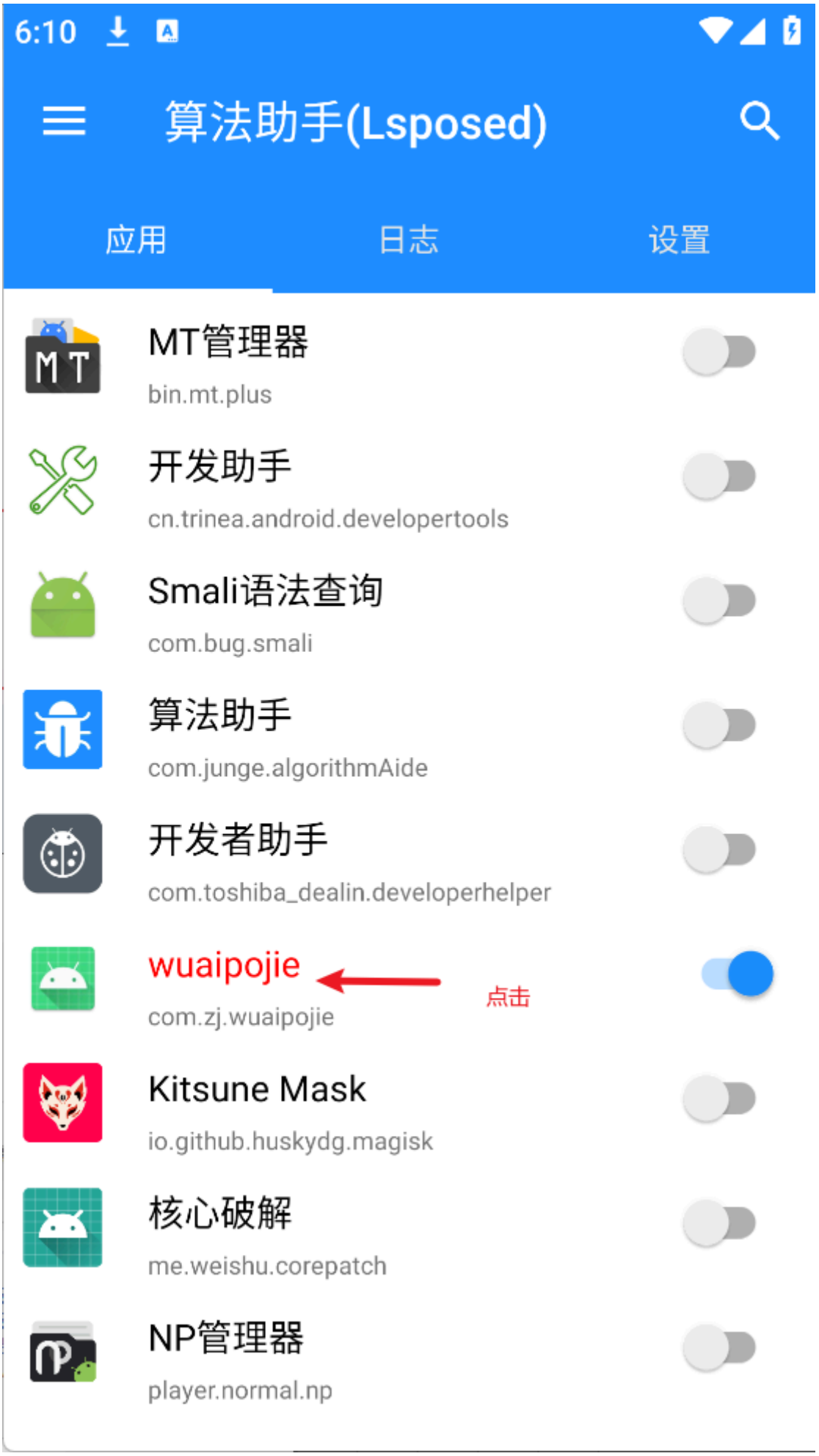
（2）Hook弹窗(推荐算法助手开启弹窗定位)

安装算法助手，并且使用LSPosed模块安装算法助手开启模块和待测试的app

4:35
算法助手
com.junge.algorithmAide
启用模块
系统框架
system
开发者助手
com.toshiba_dealin.developerhe..
版本：1.2.0



安装后，报错，需要重启解决，打开app


6:10
算法助手(Lsposed)
应用
日志
设置
MT管理器
bin.mt.plus
开发助手
cn.trinea.android.developertools
Smali语法查询
com.bug.smali
算法助手
com.junge.algorithmAide
开发者助手
com.toshiba_dealin.developerhelper
wuaipojie
点击
com.zj.wuaipojie
Kitsune Mask
io.github.huskydg.magisk
核心破解
me.weishu.corepatch
NP管理器
player.normal.np

点击后，会启动待测试的app

点击返回，可以过掉2个广告

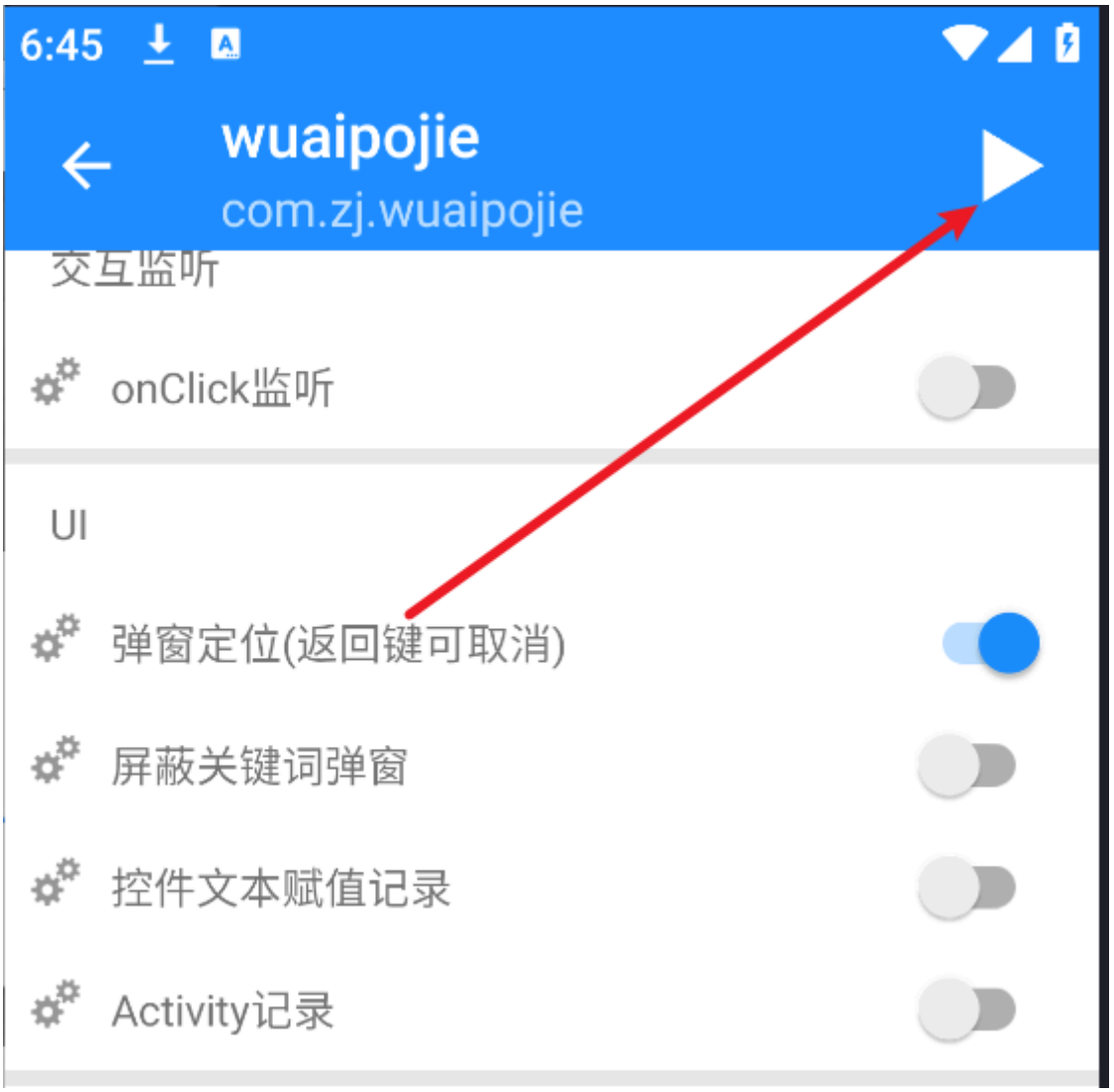
此外还可以通过关键词自动屏蔽弹窗

6:15
wuaipojie
com.zj.wuaipojie
文件写入监听
文件删除监听
shell命令行监听
交互监听
onClick监听
UI
弹窗定位(返回键可取消)
屏蔽关键词弹窗
控件文本赋值记录
Activity记录
webView

此时再打开app 则直接跳过了

（3）修改dex弹窗代码

先关闭之前的hook，关键字弹窗


6:45
wuaipojie
com.zj.wuaipojie
交互监听
onClick监听
UI
弹窗定位(返回键可取消)
屏蔽关键词弹窗
控件文本赋值记录
Activity记录

等待弹窗后

打开算法助手app的日志

wuaipojie
12条日志

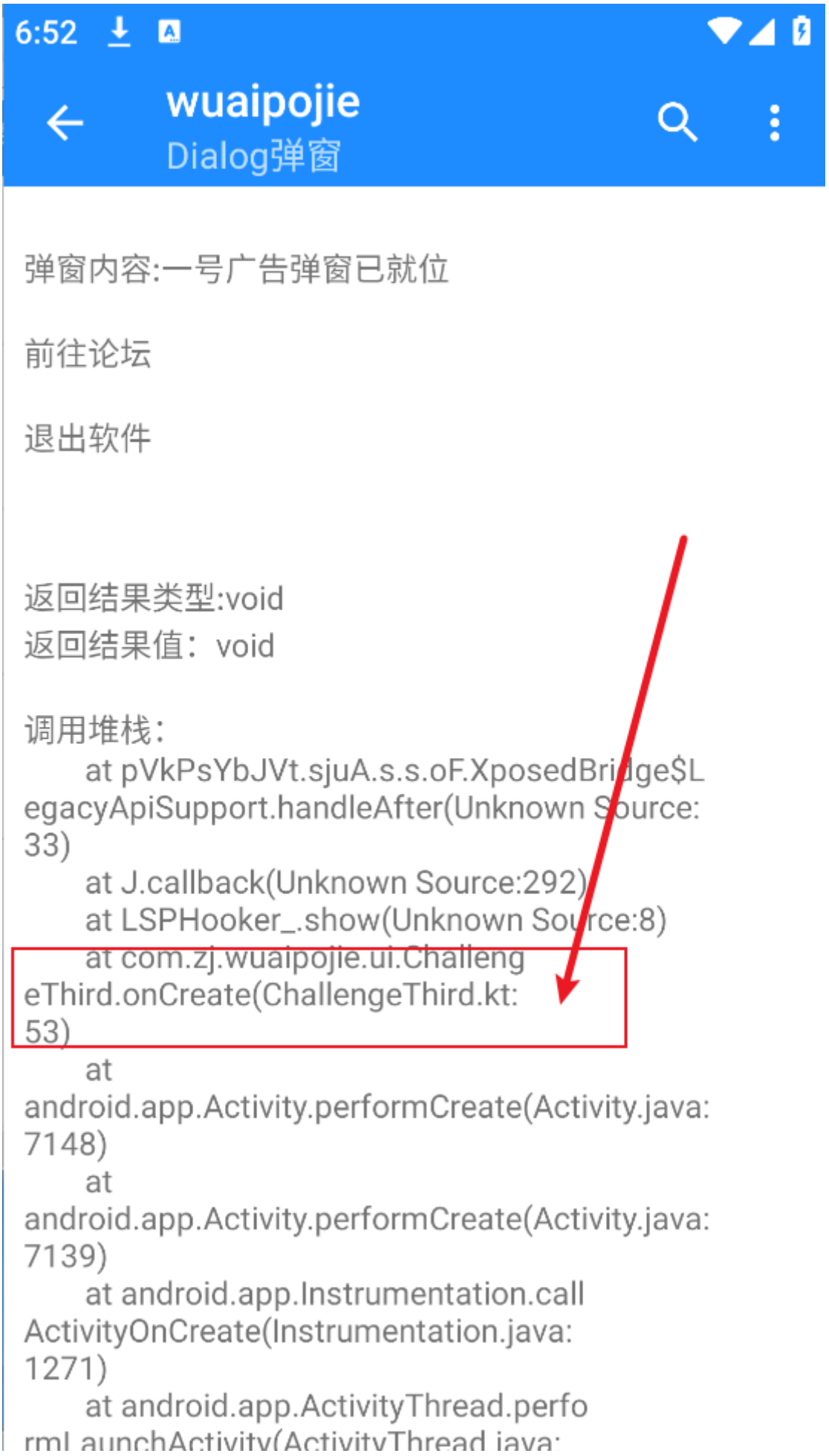
全部 加解密算法 摘要算法 其他

Dialog弹窗 18:13:56
这是二号广告标题 这是二号广告内容 退出软件 前往论坛
void 246.0B

Dialog弹窗 18:13:56
一号广告弹窗已就位 前往论坛 退出软件
void 223.0B

Dialog弹窗(已自动拦截) 18:16:10
一号广告弹窗已就位 前往论坛 退出软件
void 240.0B

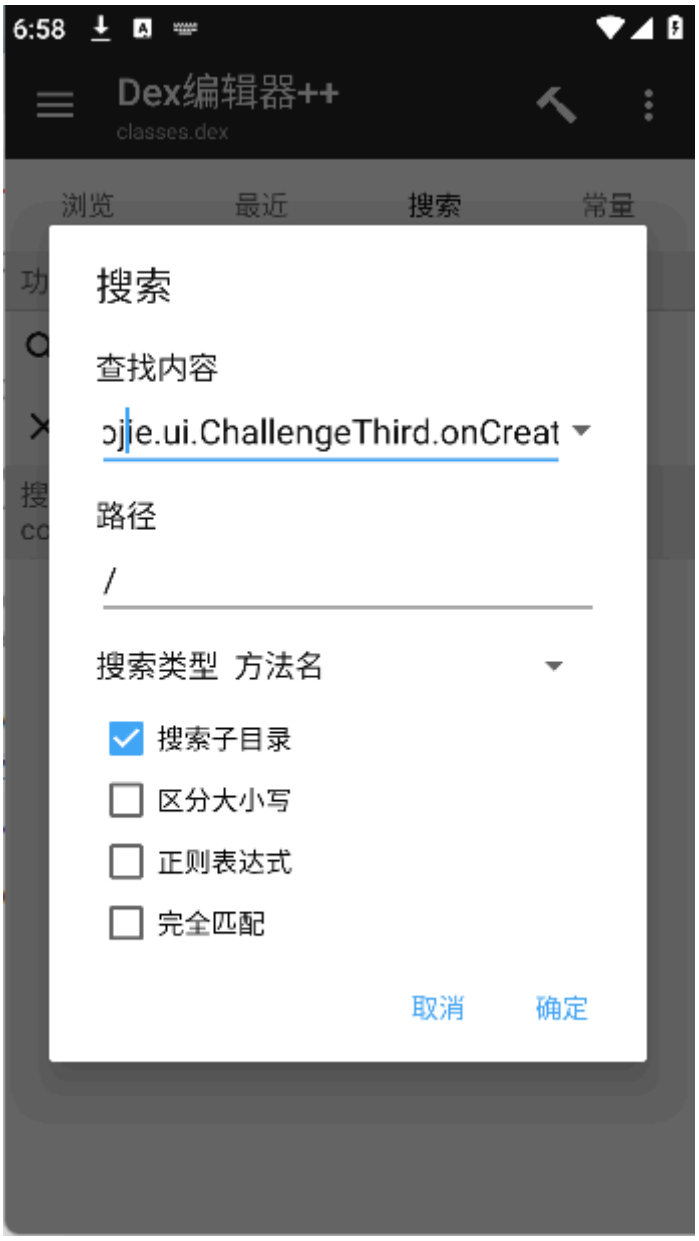
Dialog弹窗(已自动拦截) 18:16:10
这是二号广告标题 这是二号广告内容 退出软件 前往论坛
void 263.0B

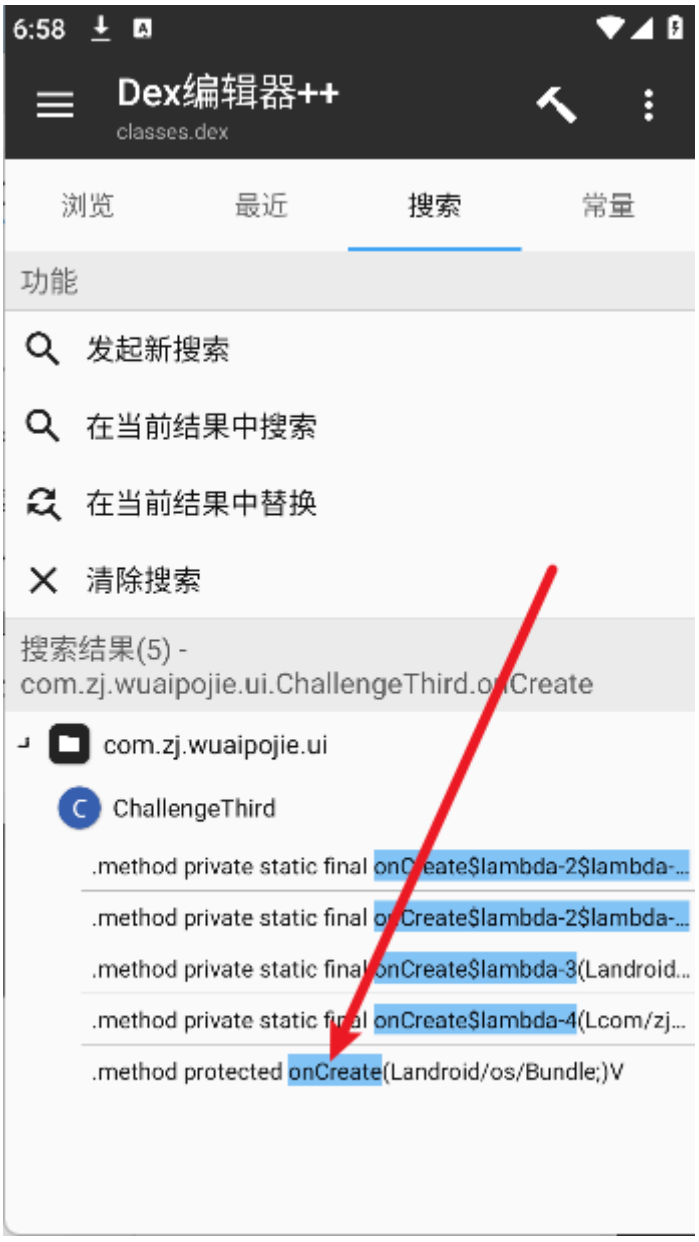


oncreate，一个Activity启动后第一个被调用的函数，常用来在此方法中进行Activity的一些初始化操作。例如创建View，绑定数据，注册监听，加载参数等。

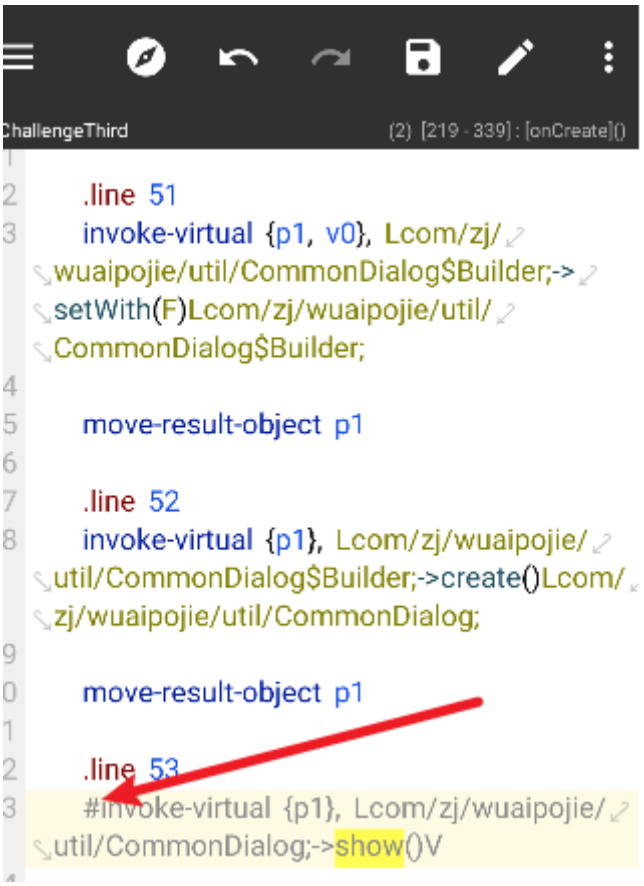
复制：com.zj.wuaipojie.ui.ChallengeThird.onCreate

然后dex文件打开搜索，oncreate是一种方法





这些弹窗一般是通过show，来弹窗的，所以可以吧show的代码注释掉



两个show的广告都注释掉，再打开界面取消了2个广告

3、去除界面文字或其他的布局

（开发助手比开发者助手的页面布局查看更有效）

打开app设置，配置无障碍

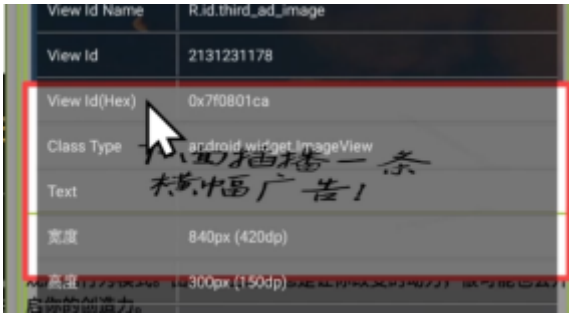
开发助手 - 开发效率工具
使用服务
我们使用此功能以快速自动打开或关闭一些开发者选项、获取当前打开的 Activity 类名(专业版)、获取当前页面布局信息(专业版)，从而节省您的时间。
提示：开启此功能时，会收到系统的警示提醒，这是系统对任何使用此功能应用的例行提醒。
我们承诺不会收集您的任何信息，请放心开启。
设置-布局查看
包名
com.android.settings
当前 Activity
com.android.settings.SubSettings

使用

8:10
系统界面-布局查看
wuaipojie
包名
com.android.systemui
当前 Activity
com.android.systemui.recents.RecentsActivity
下面插播一条
横幅广告!
产生的焦虑是让你改变的动力，很可能也会开启你的

下面插播一条
横幅广告!
2 个 View 可选
1
android.widget.ImageView
R.id.third_ad_image, 0 个 child
2
0 个 child

点击

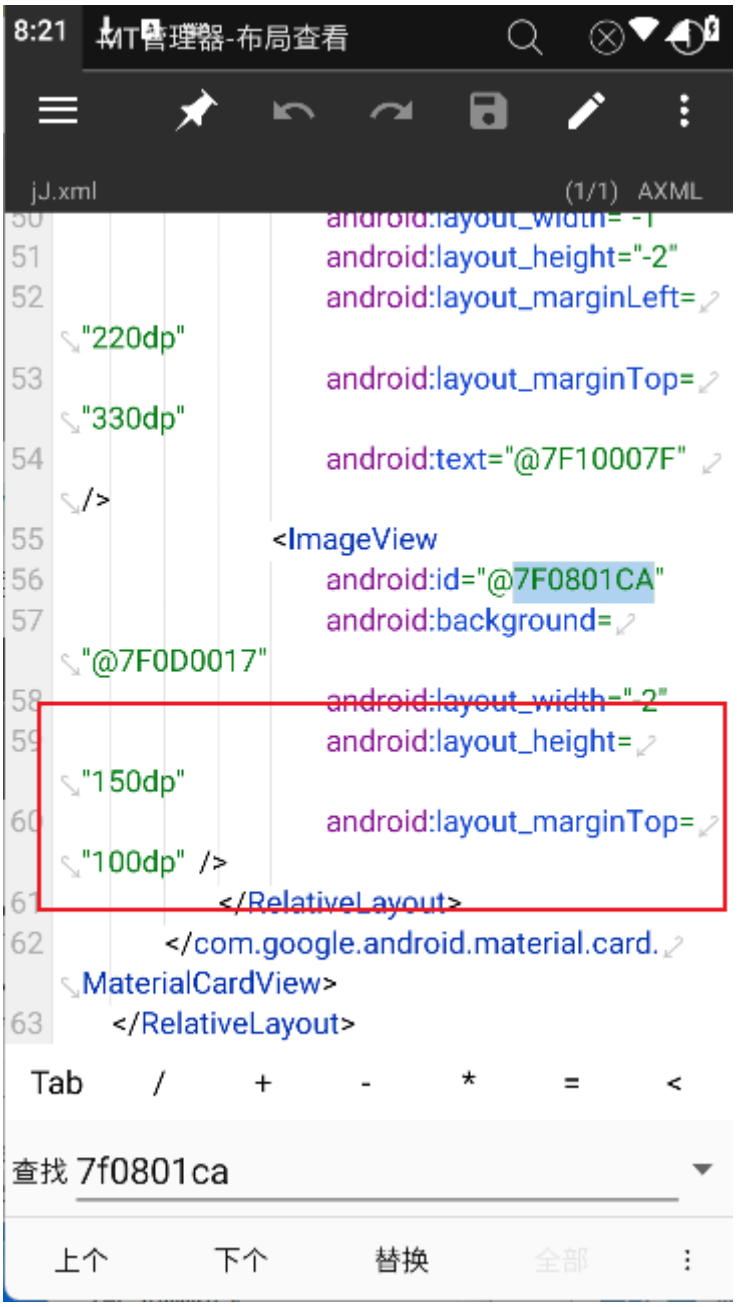

View Id Name
R.id.third_ad_image
View Id
2131231178
View Id(Hex)
0x7f0801ca
Class Type
android.widget.ImageView
Text
下面插播一条
横幅广告!
宽度
840px (420dp)

获取hex　0x7f0801ca

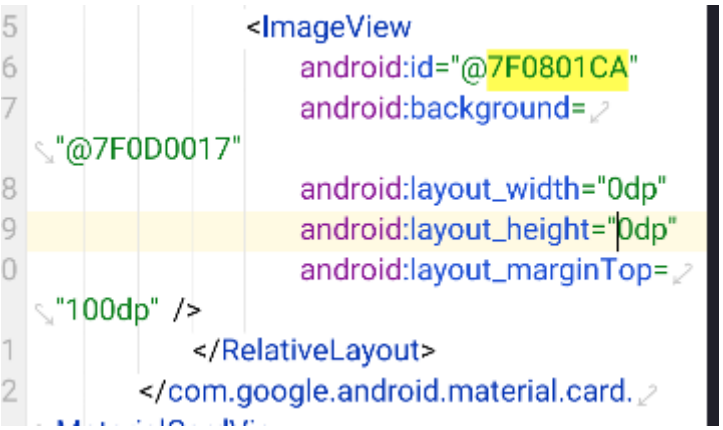
mt管理器搜索 xml搜索

使用资源搜索

8:19
MT管理器-布局查看
wuaipojie_1.0.apk/
文件夹: 4 文件: 5
..
kotlin
81-01-01 01:01
META-INF
24-11-23 20:06
acct
24-11-23 18:44
bin
24-11-07 15:37
cache
24-11-23 18:44
XML搜索
查找内容
0x7f0801ca
搜索类型 资源ID
十六进制
替换
取消
确定
json
81-01-01 01:01 628B
24-08-23 17:58
mnt
24-11-23 18:44
odm
24-08-23 17:58
oem
24-08-23 17:58



第一种方法，宽高 都改为0



第二种方法，隐藏起来

```
android:visibility="gone"
```

在Android开发中，`android:visibility="gone"` 是一个属性设置，用于控制UI组件（如按钮、文本框等）的可见性。当某个组件被设置为 `gone` 时，这个组件不仅不会显示在界面上，而且也不会占用布局空间。

五、1000-7=？&动态调试&Log插桩课件

1.配置java环境

安装java11，并配置环境

```
C:\Users\Administrator>java -version
java version "11.0.15.1" 2022-04-22 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.15.1+2-LTS-10)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.15.1+2-LTS-10, mixed mode)

C:\Users\Administrator>
```

2.什么是动态调试

动态调试是指自带的调试器跟踪自己软件的运行，可以在调试的过程中知道参数或者局部变量的值以及履清代码运行的先后顺序。多用于爆破注册码(CTF必备技能)

1、app调试

动态调试步骤

（1）修改debug权限

方法一:在AndroidManifest.xml里添加可调试权限
复制代码 隐藏代码

```
android:debuggable="true"
```

打开xml文件 查看application，文档没有直接复制进去

端口转发以及开启adb权限

《版本号点击七次开启开发者模式并开启adb调试权限
夜神模拟器：adb connect 127.0.0.1:62001》

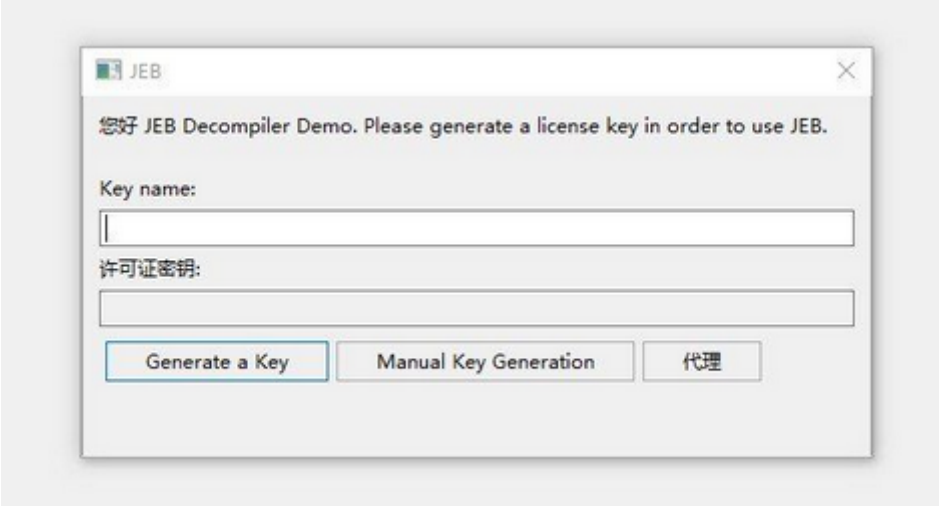
首先打开 usb调试

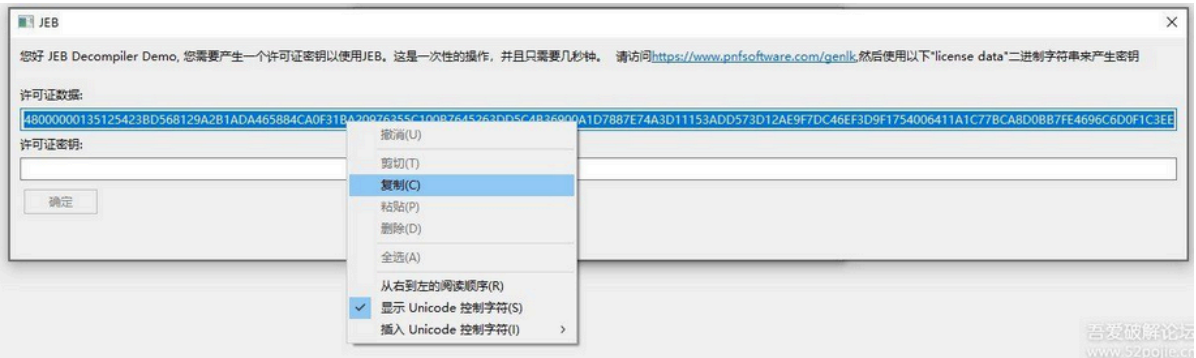
打开jeb调试工具

首次需要激活jeb

破解方法：https://www.52pojie.cn/thread-1598242-1-1.html

激活：移动好后双击jeb/jeb_wincon.bat打开jeb，此时会弹出激活注册界面，选择Manual Key Genaration，复制许可证数据。如果双击jeb_wincon.bat不能打开jeb，需要检查一下自己java环境是否配置好！





得到许可证数据后，复制下面Python代码到python编译器里，推荐到https://tool.lu/coderunner/ 这个在线代码网站上运行，代码格式选择Python，把许可证数据粘贴到

```
licdata = ""
```

的引号里，运行后即可得到KEY值。输入到上图的许可证密匙后即可激活JEB

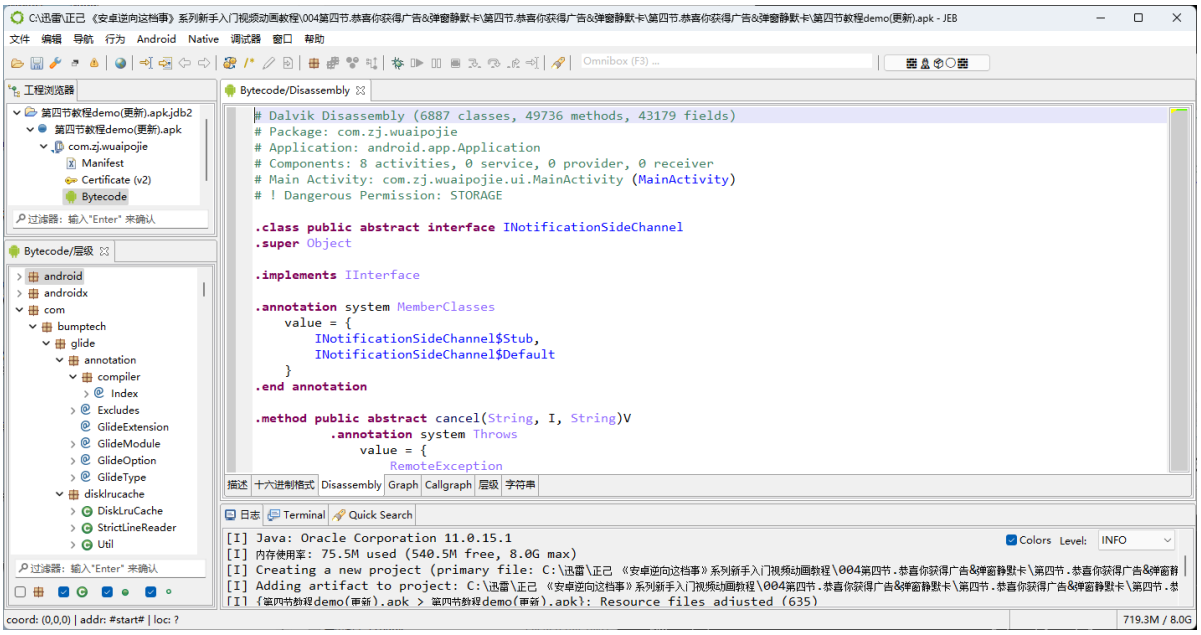
```
#https://bbs.pediy.com/
#!/usr/bin/env python
import os, sys, struct, time, binascii, hashlib

RC4_Key2= 'Eg\xa2\x99_\x83\xf1\x10'

def rc4(Key, inData):
    Buf = ""
    S = range(256)
    K = (map(lambda x:ord(x), Key) * (256 / len(Key) + 1))[:256]
    j = 0
    for i in range(256):
        j = (S[i] + K[i] + j) % 256
        S[i], S[j] = S[j], S[i]
    i, j = 0, 0
    for x in range(len(inData)):
        i = (i + 1) % 256
        j = (j + S[i]) % 256
        S[i], S[j] = S[j], S[i]
        Buf += chr(S[(S[j] + S[i]) % 256] ^ ord(inData[x]))
    return Buf

def Long2Int(longdata):
```

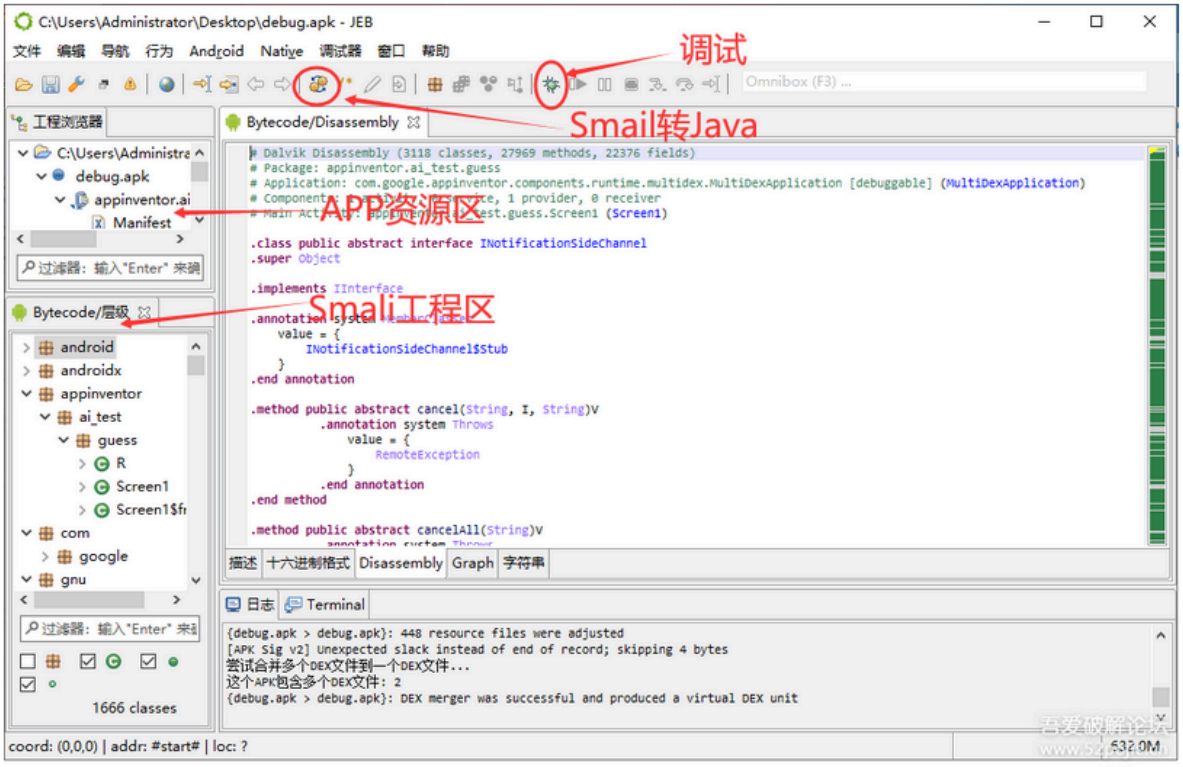
```
    lo = longdata & 0xFFFFFFFF
    hi = (longdata >> 32) & 0x7FFFFFFF
    return hi, lo

def KeygenSN(LicenseSerial, MachineID):
    mhi, mlo = Long2Int(MachineID)
    lhi, llo = Long2Int(LicenseSerial)
    hi_Key = (mhi - lhi + 0x55667788) & 0x7FFFFFFF
    lo_Key = (mlo + llo + 0x11223344) & 0xFFFFFFFF
    Z0, = struct.unpack('<Q', struct.pack('<LL', lo_Key, hi_Key))
    Z1 = int(time.time()) ^ 0x56739ACD
    s = sum(map(lambda x:int(x, 16), "%x" % Z1)) % 10
    return "%dZ%d%d" % (Z0, Z1, s)

def ParsePost(buf):
    Info = struct.unpack('<3L2Q4LQ3L', buf[:0x40])
    flag, CRC, UserSerial, LicenseSerial, MachineID, build_type, \
          Ver_Major, Ver_Minor, Ver_Buildid, Ver_Timestamp, \
          TimeOffset, Kclass, Random2 = Info
    SysInfoData = buf[0x40:]
    assert CRC == binascii.crc32(buf[8:]) & 0xFFFFFFFF
    return Info, SysInfoData

def DecodeRc4Str(buf):
    buf = buf.decode('hex')
    i, s = ParsePost(rc4(buf[:8] + RC4_Key2, buf[8:]))
    return i, s

def GetJebLicenseKey():
    licdata = ""
    if licdata:
        i, MachineID = DecodeRc4Str(licdata)
        SN = KeygenSN(i[3], i[4])
        print "JEB License Key:", SN
        return SN

GetJebLicenseKey()
raw_input("Enter to Exit...")
```

设置中文：在菜单栏里Edit->Language->中文（中文），关闭软件重新打开即可为中文界面。

工具调试app

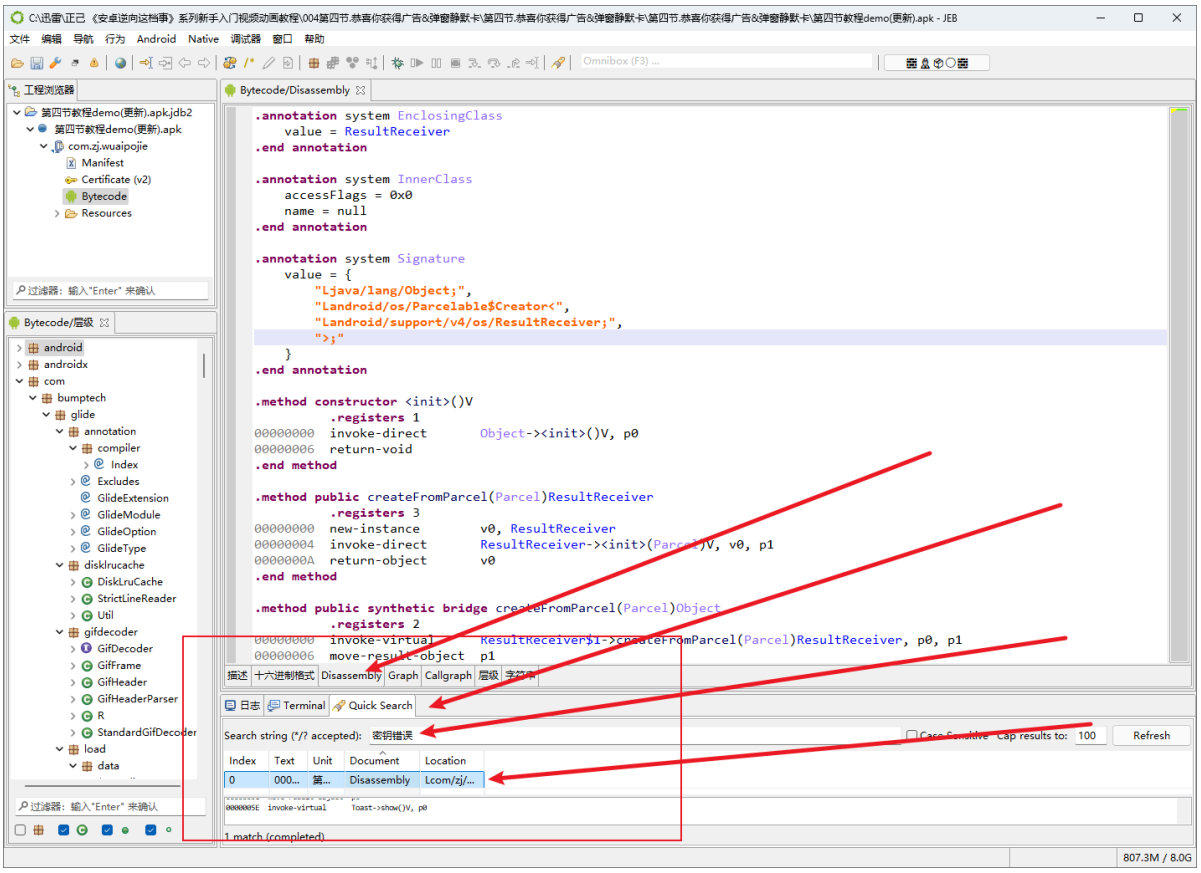
将app拖入jeb工具中


C:\迅雷\正己 《安卓逆向这档事》系列新手入门视频动画教程\004第四节.恭喜你获得广告&弹窗静默卡\第四节.恭喜你获得广告&弹窗静默卡\第四节教程demo(更新).apk - JEB
文件 编辑 导航 行为 Android Native 调试器 窗口 帮助
工程浏览器
第四节教程demo(更新).apk.jdb2
第四节教程demo(更新).apk
com.zj.wuaipojie
Manifest
Certificate (v2)
Bytecode
Bytecode/Disassembly
Dalvik Disassembly (6887 classes, 49736 methods, 43179 fields)
Package: com.zj.wuaipojie
Application: android.app.Application
Components: 8 activities, 0 service, 0 provider, 0 receiver
Main Activity: com.zj.wuaipojie.ui.MainActivity (MainActivity)
! Dangerous Permission: STORAGE
.class public abstract interface INotificationSideChannel
.super Object
.implements IInterface
.annotation system MemberClasses
value = {
INotificationSideChannel$Stub,
INotificationSideChannel$Default
}
.end annotation
.method public abstract cancel(String, I, String)V
.annotation system Throws
value = {
RemoteException
Bytecode/层级
android
androidx
com
bumptech
glide
annotation
compiler
Index
Excludes
GlideExtension
GlideModule
GlideOption
GlideType
disklrucache
DiskLruCache
StrictLineReader
Util
[I] Java: Oracle Corporation 11.0.15.1
[I] 内存使用率: 75.5M used (540.5M free, 8.0G max)
[I] Creating a new project (primary file: C:\迅雷\正己 《安卓逆向这档事》系列新手入门视频动画教程\004第四节.恭喜你获得广告&弹窗静默卡\第四节.恭喜你获得广告&弹窗
[I] Adding artifact to project: C:\迅雷\正己 《安卓逆向这档事》系列新手入门视频动画教程\004第四节.恭喜你获得广告&弹窗静默卡\第四节.恭喜你获得广告&弹窗静默卡\第四节.恭
[I] {第四节教程demo(更新).apk > 第四节教程demo(更新).apk}: Resource files adjusted (635)
coord: (0,0,0) | addr: #start# | loc: ?
719.3M / 8.0G

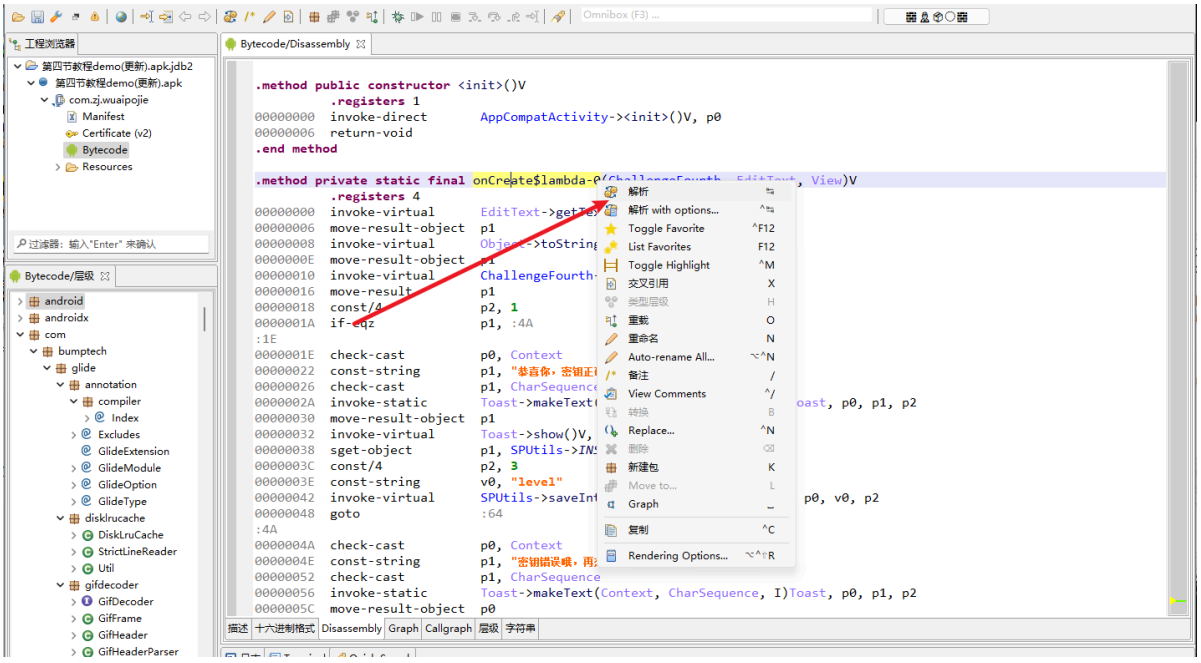

C:\Users\Administrator\Desktop\debug.apk - JEB
文件 编辑 导航 行为 Android Native 调试器 窗口 帮助
调试
Smail转Java
APP资源区
Smali工程区
工程浏览器
debug.apk
appinventor.ai
Manifest
Bytecode/Disassembly
Dalvik Disassembly (3118 classes, 27969 methods, 22376 fields)
Package: appinventor.ai_test.guess
Application: com.google.appinventor.components.runtime.multidex.MultiDexApplication [debuggable] (MultiDexApplication)
.class public abstract interface INotificationSideChannel
.super Object
.implements IInterface
INotificationSideChannel$Stub
.end annotation
.method public abstract cancel(String, I, String)V
.annotation system Throws
value = {
RemoteException
.end annotation
.end method
.method public abstract cancelAll(String)V
Bytecode/层级
android
androidx
appinventor
ai_test
guess
R
Screen1
com
google
gnu
1666 classes
描述 十六进制格式 Disassembly Graph 字符串
日志 Terminal
{debug.apk > debug.apk}: 448 resource files were adjusted
[APK Sig v2] Unexpected slack instead of end of record; skipping 4 bytes
尝试合并多个DEX文件到一个DEX文件...
这个APK包含多个DEX文件: 2
{debug.apk > debug.apk}: DEX merger was successful and produced a virtual DEX unit
coord: (0,0,0) | addr: #start# | loc: ?
632.0M

运行app，找到关键字

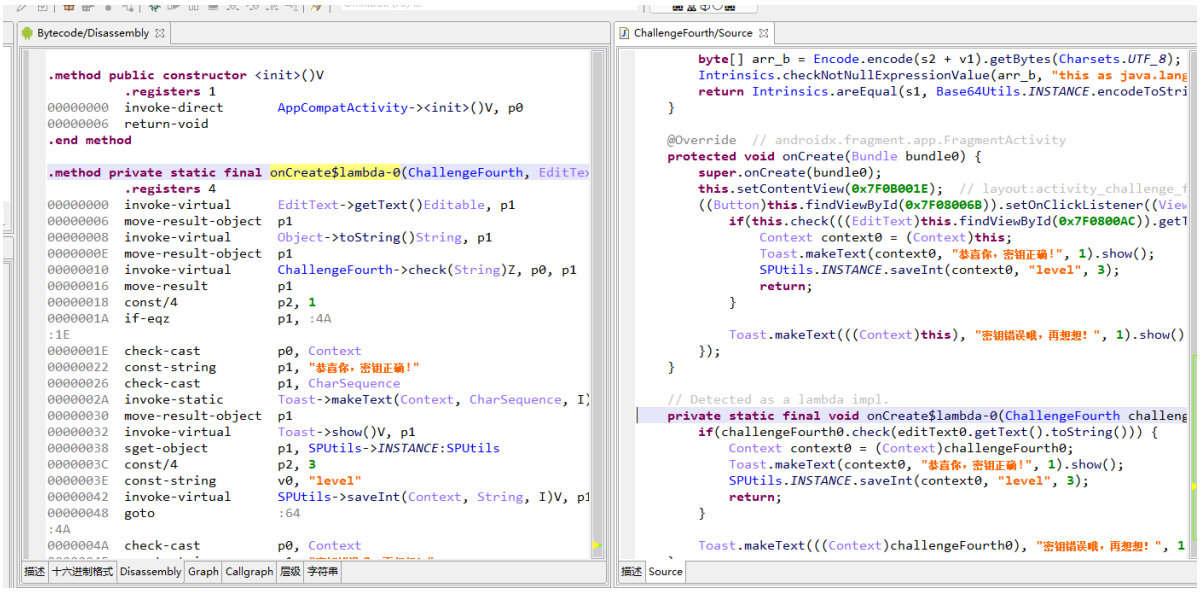
10:45
wuaipojie-布局查看
wuaipojie
Tip:请输入正确的密钥
1
验证
密钥错误哦，再想想！



搜索密钥错误没然后双击最后的箭头



解析为java




```
protected void onCreate(Bundle bundle0) {

    super.onCreate(bundle0); // 调用父类的onCreate方法，初始化活动

    this.setContentView(0x7F0B001E);  // 设置活动的内容视图为
layout:activity_challenge_fourth

    // 为按钮设置点击监听器，按钮ID为0x7F08006B

    ((Button)this.findViewById(0x7F08006B)).setOnClickListener((View view0) -> {

        // 获取EditText中输入的文本，ID为0x7F0800AC


 if(this.check(((EditText)this.findViewById(0x7F0800AC)).getText().toString())) {


            // 如果输入的密钥正确

            Context context0 = (Context)this;

            // 显示Toast提示“恭喜你，密钥正确！”

            Toast.makeText(context0, "恭喜你，密钥正确！",
Toast.LENGTH_SHORT).show();

            // 使用SPUtils保存level为3

            SPUtils.INSTANCE.saveInt(context0, "level", 3);

            return;

        }

        // 如果输入的密钥错误，显示Toast提示“密钥错误哦，再想想！”

        Toast.makeText(((Context)this), "密钥错误哦，再想想！",
Toast.LENGTH_SHORT).show();
```

```
    });

}
```

此处需要跟进check，双击check跳转

ai分析代码

```
@Metadata(
    d1 =
{"\u0000$\n\u0002\u0018\u0002\n\u0002\u0018\u0002\n\u0002\b\u0002\n\u0002\u0010\u
000B\n\u0000\n\u0002\u0010\u000E\n\u0000\n\u0002\u0010\u0002\n\u0000\n\u0002\u001
8\u0002\n\u0000\u0018\u00002\u00020\u0001B\u0005¢
\u0006\u0002\u0010\u0002J\u000E\u0010\u0003\u001A\u00020\u00042\u0006\u0010\u0005
\u001A\u00020\u0006J\u0012\u0010\u0007\u001A\u00020\b2\b\u0010\t\u001A\u0004\u001
8\u00010\nH\u0014¨\u0006\u000B"},
    d2 = {"Lcom/zj/wuaipojie/ui/ChallengeFourth;",
"Landroidx/appcompat/app/AppCompatActivity;", "()V", "check", "", "str", "",
"onCreate", "", "savedInstanceState", "Landroid/os/Bundle;", "app_release"},
    k = 1,
    mv = {1, 7, 1},
    xi = 0x30
)
// 这是一个Kotlin类，继承自AppCompatActivity，用于Android开发
public final class ChallengeFourth extends AppCompatActivity {

    // 定义一个公开的最终方法check，接收一个字符串参数s，返回一个布尔值
    public final boolean check(String s) {
        int v = 0; // 初始化一个计数器v
        Integer integer0 = null; // 初始化一个可空的Integer对象integer0

        // 检查字符串s是否以"flag{"开头，如果不是，则返回false
        if(!StringsKt.startsWith$default(s, "flag{", false, 2, null)) {
            return false;
        }

        // 检查字符串s是否以"}"结尾，如果不是，则返回false
        if(!StringsKt.endsWith$default(s, "}", false, 2, null)) {
            return false;
        }

        // 截取字符串s中"flag{"和"}"之间的部分，并检查截取结果不为null
        String s1 = s.substring(5, s.length() - 1);
        Intrinsics.checkNotNullExpressionValue(s1, "this as java.lang.String…
ing(startIndex, endIndex)");

        // 从SharedPreferences中获取键为"id"的字符串，默认值为空字符串
        String s2 = SPUtils.INSTANCE.getString(((Context)this), "id", "");

        // 如果s2不为null，则获取其长度并赋值给integer0
        if(s2 != null) {
            integer0 = (int)s2.length();
        }

        // 初始化一个变量v1为1000
```

```
        int v1 = 1000;

        // 检查integer0不为null
        Intrinsics.checkNotNull(integer0);

        // 将integer0的值转换为int类型并赋值给v2
        int v2 = (int)integer0;

        // 如果v2大于等于0，则执行以下循环
        if(v2 >= 0) {
            while(true) {
                v1 += -7; // 每次循环v1减7
                if(v == v2) { // 当v等于v2时跳出循环
                    break;
                }
                ++v; // 每次循环v加1
            }
        }

        // 对s2和v1的拼接结果进行编码，并将编码结果转换为UTF-8字节数组
        byte[] arr_b = Encode.encode(s2 + v1).getBytes(Charsets.UTF_8);

        // 检查编码结果不为null
        Intrinsics.checkNotNullExpressionValue(arr_b, "this as
java.lang.String).getBytes(charset)");

        // 将字节数组arr_b进行Base64编码，并检查编码后的字符串是否与s1相等
        return Intrinsics.areEqual(s1,
Base64Utils.INSTANCE.encodeToString(arr_b));
    }
}
```

在smali代码中，找到check方法

```
Bytecode/Disassembly
.end method

.method public final check(String)Z
          .registers 8
00000000  const-string        v0, "flag{"
00000004  const/4             v1, 0
00000006  const/4             v2, 2
00000008  const/4             v3, 0
0000000A  invoke-static       StringsKt->startsWith$default(String, String, Z, I, Ob
00000010  move-result         v0
00000012  if-nez              v0, :18
:16
00000016  return              v1
:18
00000018  const-string        v0, "}"
0000001C  invoke-static       StringsKt->endsWith$default(String, String, Z, I, Obje
00000022  move-result         v0
00000024  if-nez              v0, :2A
:28
00000028  return              v1
:2A
0000002A  const/4             v0, 5
0000002C  invoke-virtual      String->length()I, p1
00000032  move-result         v2
00000034  add-int/lit8        v2, v2, -1
00000038  invoke-virtual      String->substring(I, I)String, p1, v0, v2
0000003E  move-result-object  p1
00000040  const-string        v0, "this as java.lang.String…ing(startIndex, endIndex)
00000044  invoke-static       Intrinsics->checkNotNullExpressionValue(Object, String)
0000004A  sget-object         v0, SPUtils->INSTANCE:SPUtils
0000004E  move-object         v2, p0
```

在base64这行代码，下断点

```
Bytecode/Disassembly
00000092  if-eq               v1, v3, :9C
:96
00000096  add-int/lit8        v1, v1, 1
0000009A  goto                :8E
:9C
0000009C  new-instance        v1, StringBuilder
000000A0  invoke-direct       StringBuilder-><init>()V, v1
000000A6  invoke-virtual      StringBuilder->append(String)StringBuilder, v1, v0
000000AC  move-result-object  v0
000000AE  invoke-virtual      StringBuilder->append(I)StringBuilder, v0, v2
000000B4  move-result-object  v0
000000B6  invoke-virtual      StringBuilder->toString()String, v0
000000BC  move-result-object  v0
000000BE  invoke-static       Encode->encode(String)String, v0
000000C4  move-result-object  v0
000000C6  sget-object         v1, Base64Utils->INSTANCE:Base64Utils
000000CA  sget-object         v2, Charsets->UTF_8:Charset
000000CE  invoke-virtual      String->getBytes(Charset)[B, v0, v2
000000D4  move-result-object  v0
000000D6  const-string        v2, "this as java.lang.String).getBytes(charset)"
000000DA  invoke-static       Intrinsics->checkNotNullExpressionValue(Object, String)
000000E0  invoke-virtual      Base64Utils->encodeToString([B)String, v1, v0
000000E6  move-result-object  v0
000000E8  invoke-static       Intrinsics->areEqual(Object, Object)Z, p1, v0
000000EE  move-result         p1
000000F0  return              p1
.end method

.method protected onCreate(Bundle)V
          .registers 4
00000000  invoke-super        AppCompatActivity->onCreate(Bundle)V, p0, p1
```

雷电模拟器配置adb环境，debug模式启动

```
adb shell am start -D -n
adb shell am start -D -n 包名/类名
am start -n 表示启动一个activity
am start -D 表示将应用设置为可调试模式
```

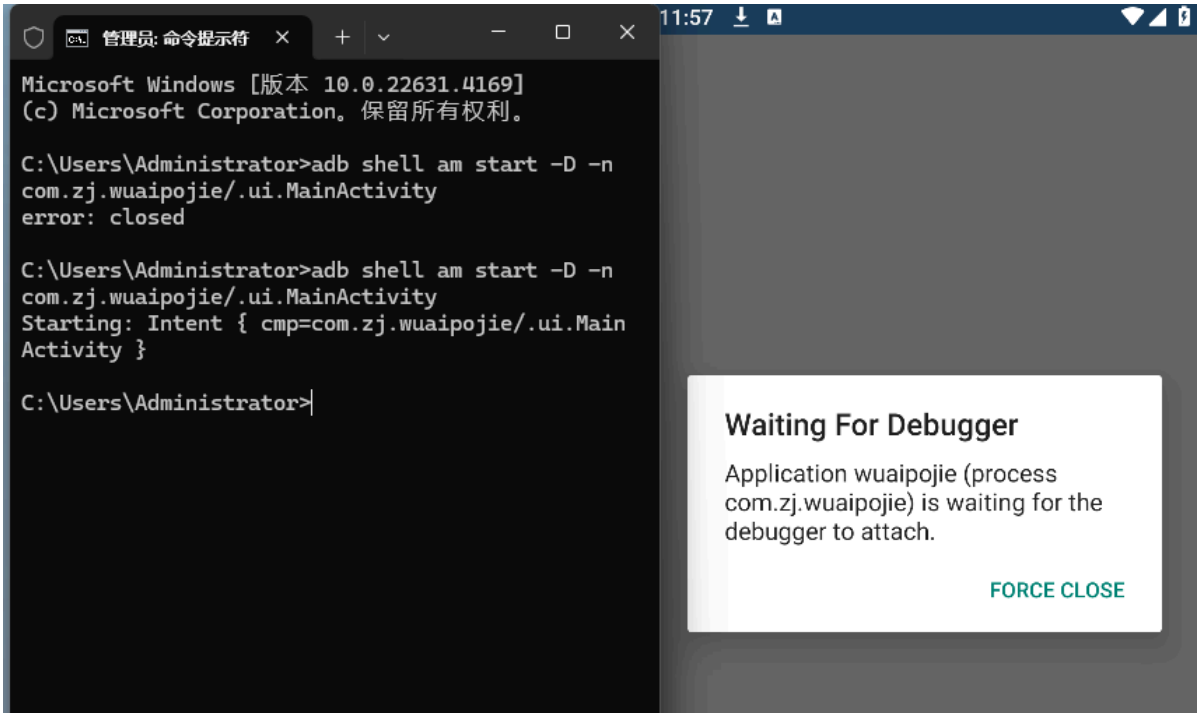
AndroidManifest.xml

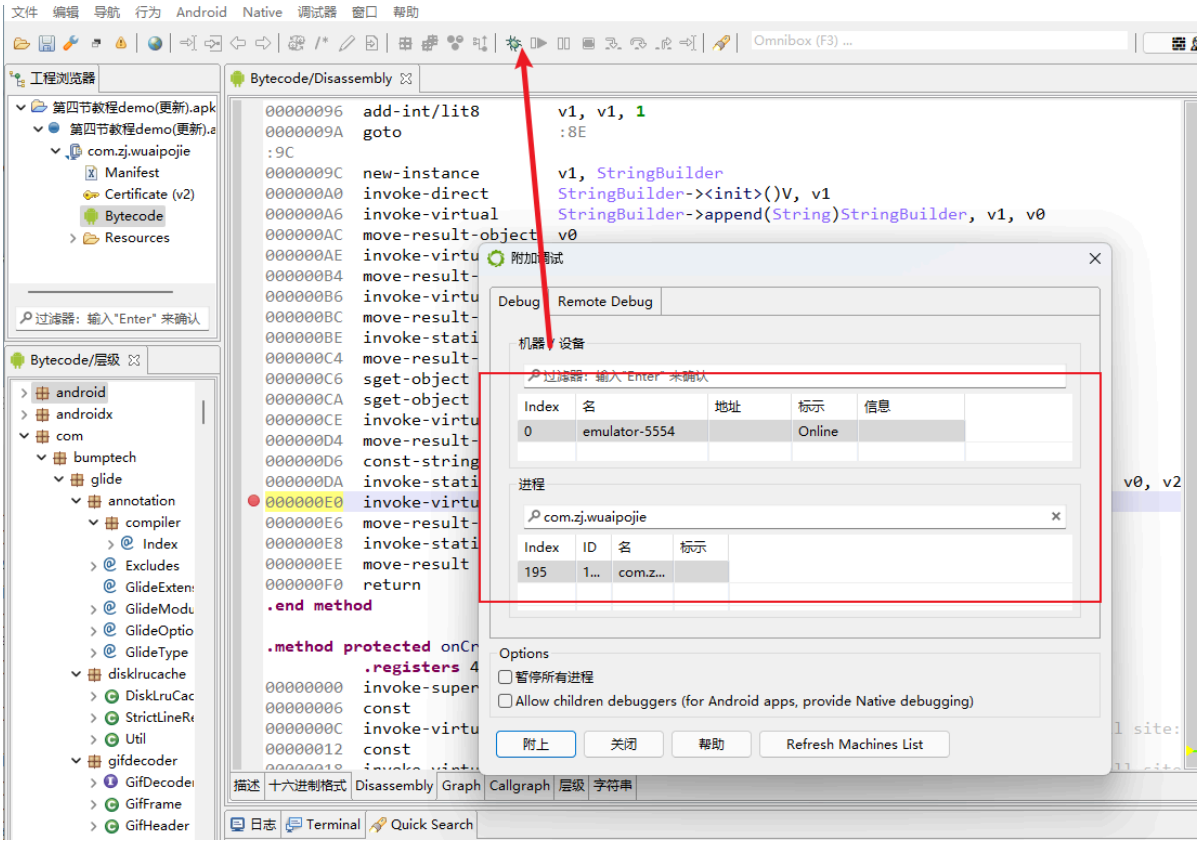
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" android:compileSdkVersion="32" android:co
    <uses-sdk android:minSdkVersion="27" android:targetSdkVersion="32"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <application android:theme="@style/Theme.Wuaipojie" android:label="@string/app_name" android:icon="@mipmap/ic_launcher" android:allowBackup="true" android:s
        <activity android:name="com.zj.wuaipojie.ui.ChallengeSixth" android:exported="false"/>
        <activity android:name="com.zj.wuaipojie.ui.ChallengeFifth" android:exported="true"/>
        <activity android:name="com.zj.wuaipojie.ui.ChallengeFourth" android:exported="true"/>
        <activity android:name="com.zj.wuaipojie.ui.ChallengeThird" android:exported="false"/>
        <activity android:name="com.zj.wuaipojie.ui.ChallengeSecond" android:exported="false"/>
        <activity android:name="com.zj.wuaipojie.ui.AdActivity"/>
        <activity android:label="@string/app_name" android:name="com.zj.wuaipojie.ui.MainActivity" android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity android:name="com.zj.wuaipojie.ui.ChallengeFirst"/>
    </application>
</manifest>
```

```
android:name="com.zj.wuaipojie.ui.MainActivity
```

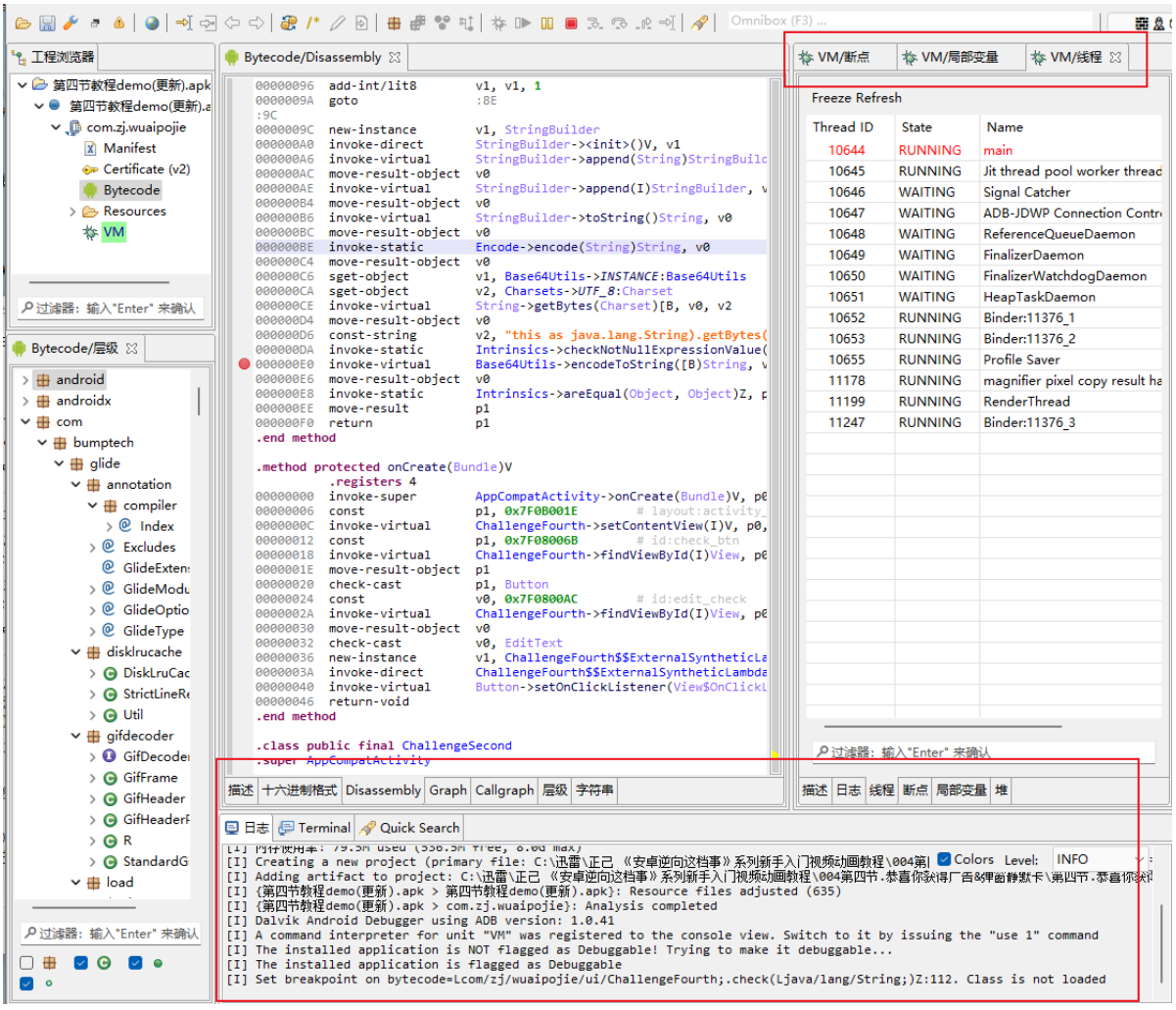
adb运行命令为

```
adb shell am start -D -n com.zj.wuaipojie/.ui.MainActivity
```





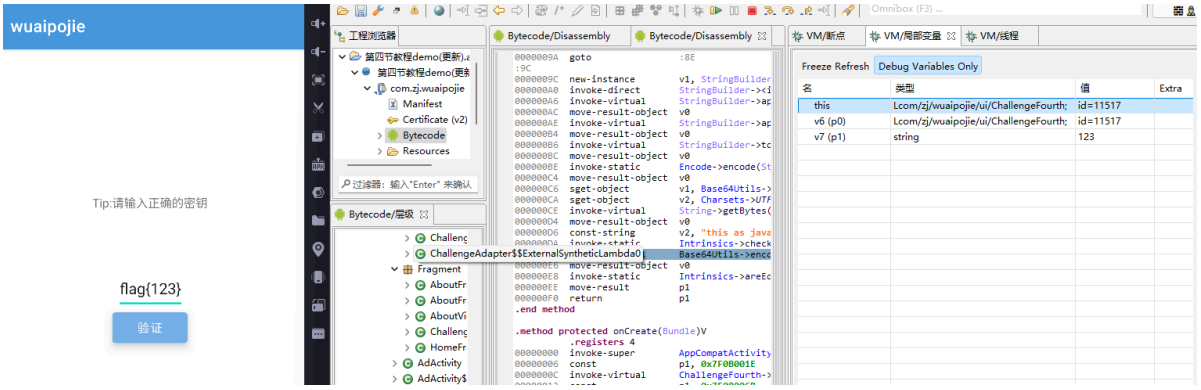
当出现下面这个弹框时说明已经连接好了，这个时候选择“附上”



出现这些信息时说明已经开始调试了，只需要关注局部变量

接下来随便输入数据，让JEB能断下来！

回到模拟器，输入flag{123}

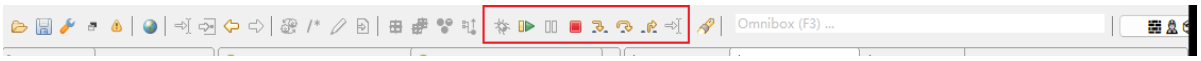


快捷键：

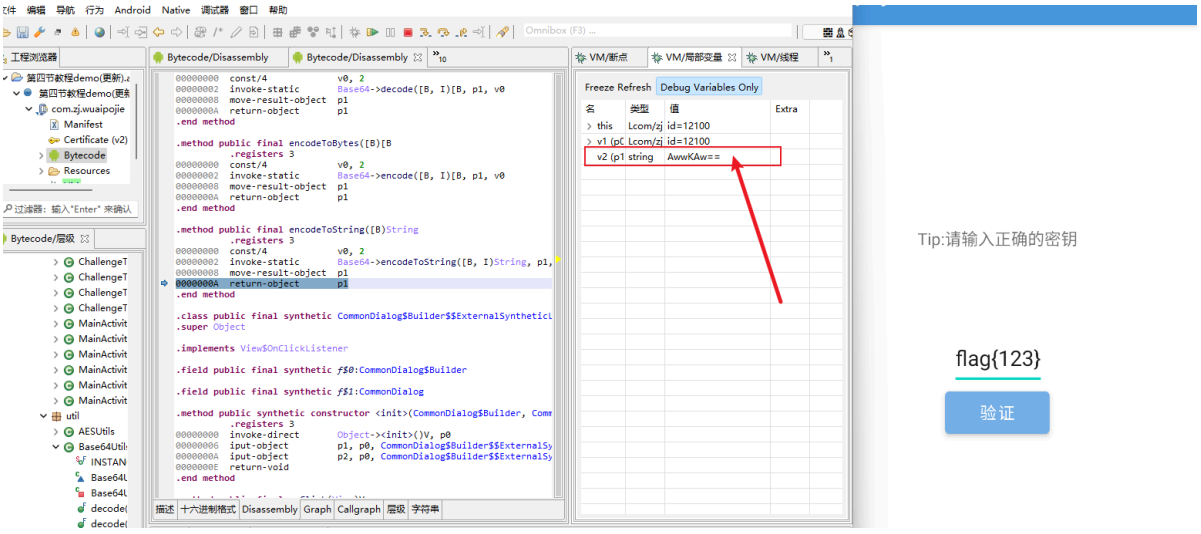
F6进入方法

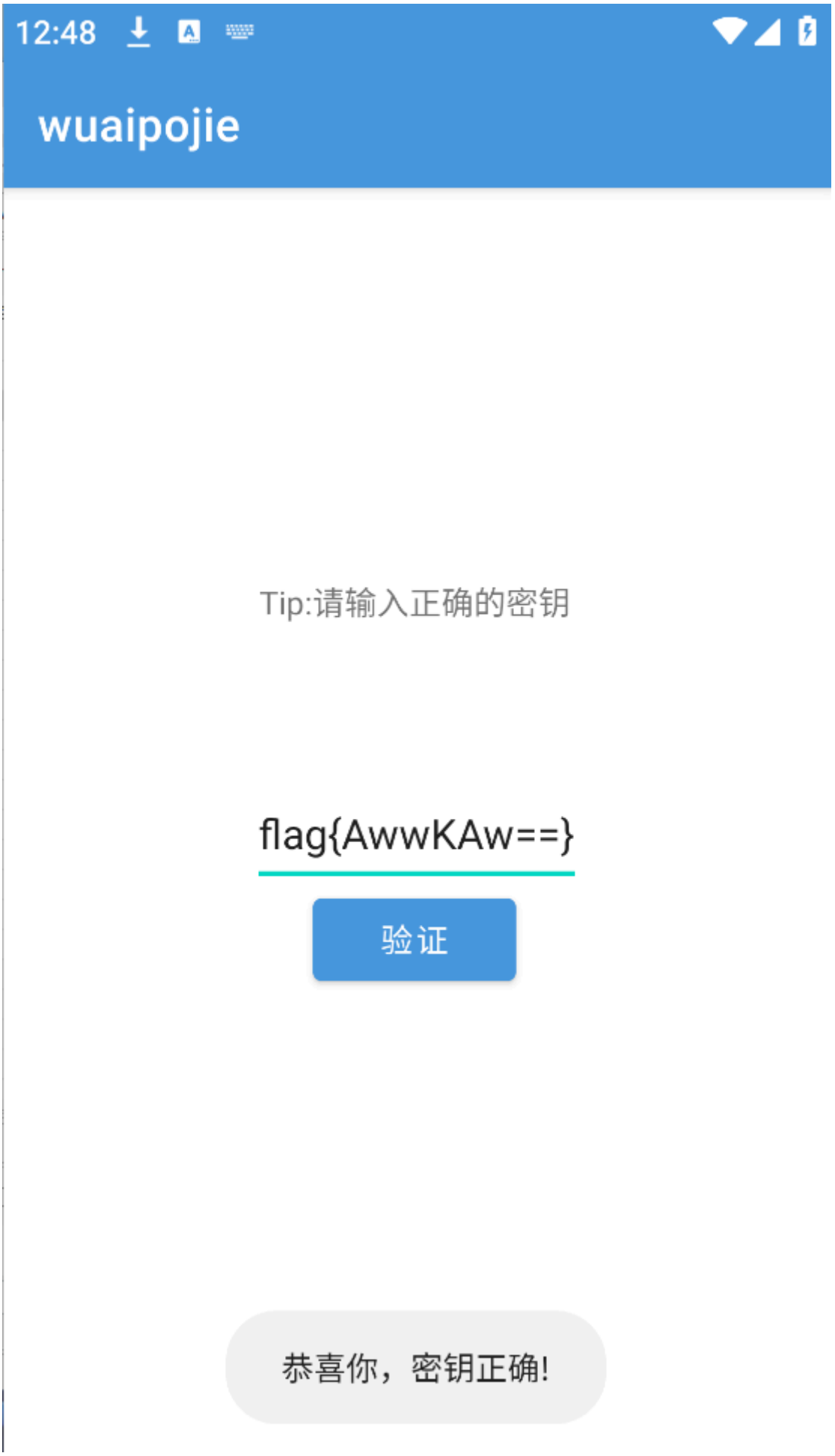
F7从方法中跳出来

R运行到光标处



进入方法调试




12:48
wuaipojie
Tip:请输入正确的密钥
flag{AwwKAw==}
验证
恭喜你，密钥正确!

（2）XappDebug模块hook对应的app

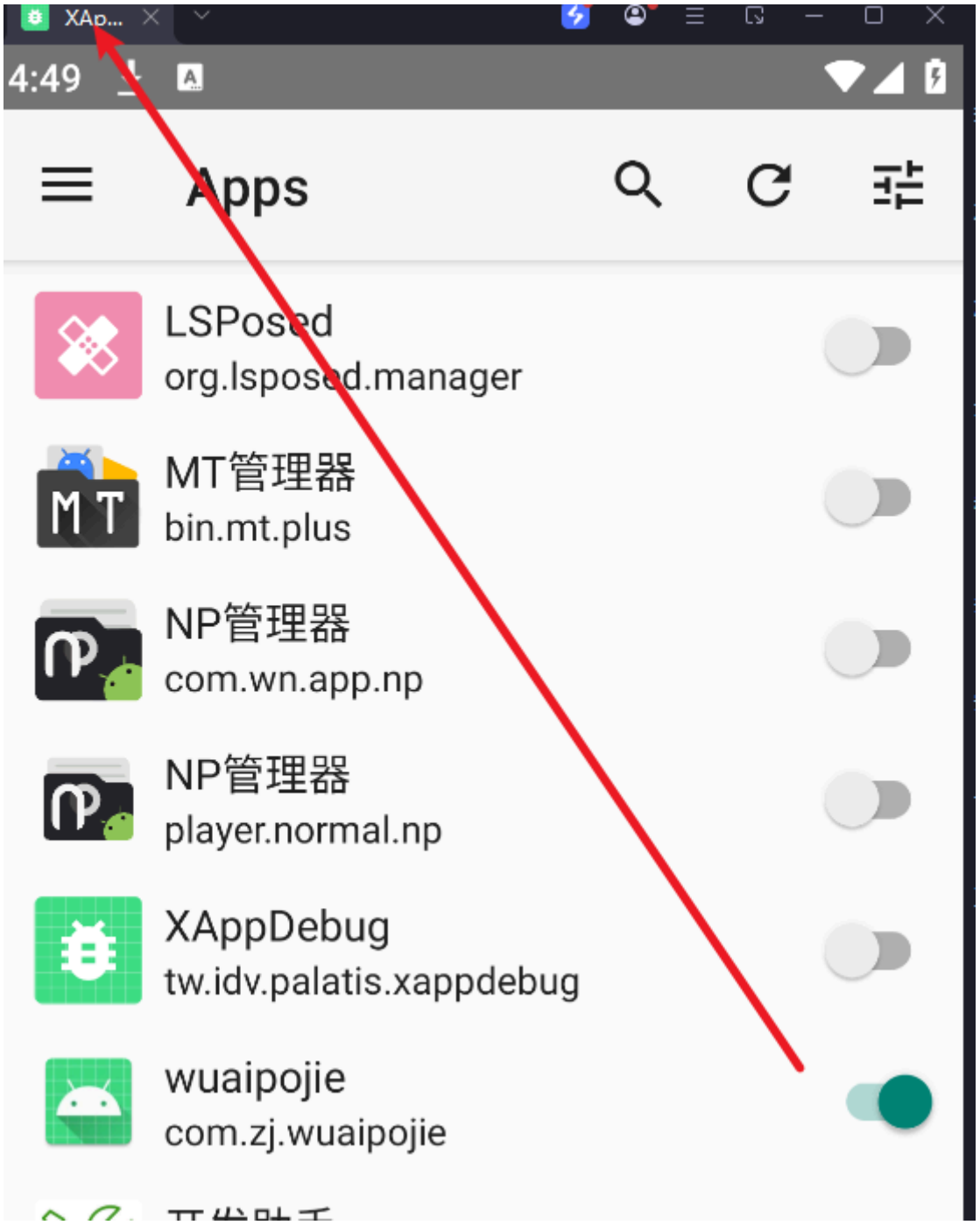
地址：https://github.com/Palatis/XAppDebug.git

安装xappdebug，并使用lsposed框架激活

打开lsposed框架app配置

然后需要重启模拟器

打开xappdebug的app，选择，需要测试的app



此时 即使app的xml文件没有开启android:debuggable="true"同样可以调试

(3) Magisk命令(重启失效)

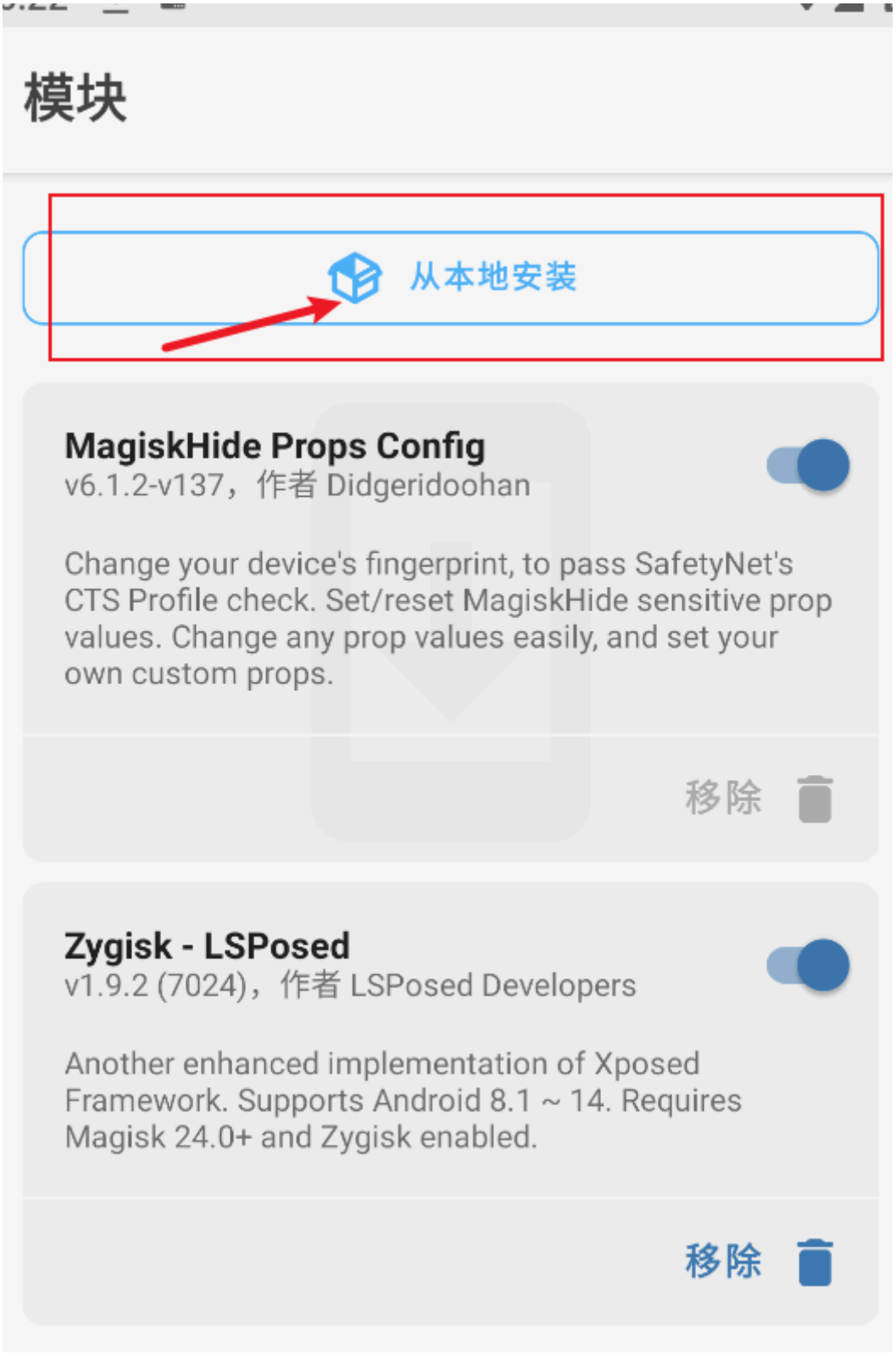
使用电脑的cmd 执行以下命令

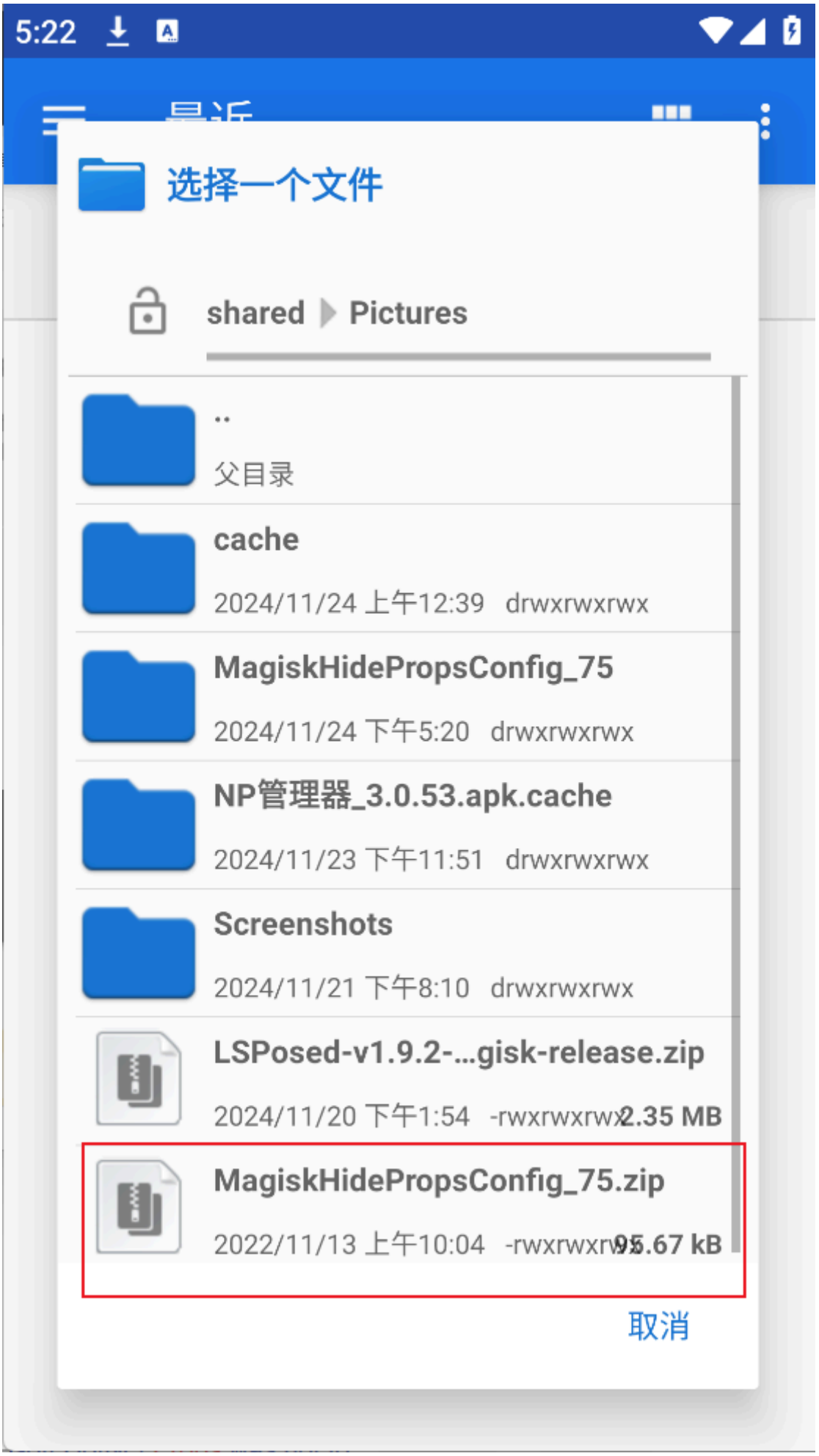
```
adb shell #adb进入命令行模式

su #切换至超级用户

magisk resetprop ro.debuggable 1

stop;start; #一定要通过该方式重启
```

（4）刷入MagiskHide Props Config模块

（(永久有效，但不一定成功）

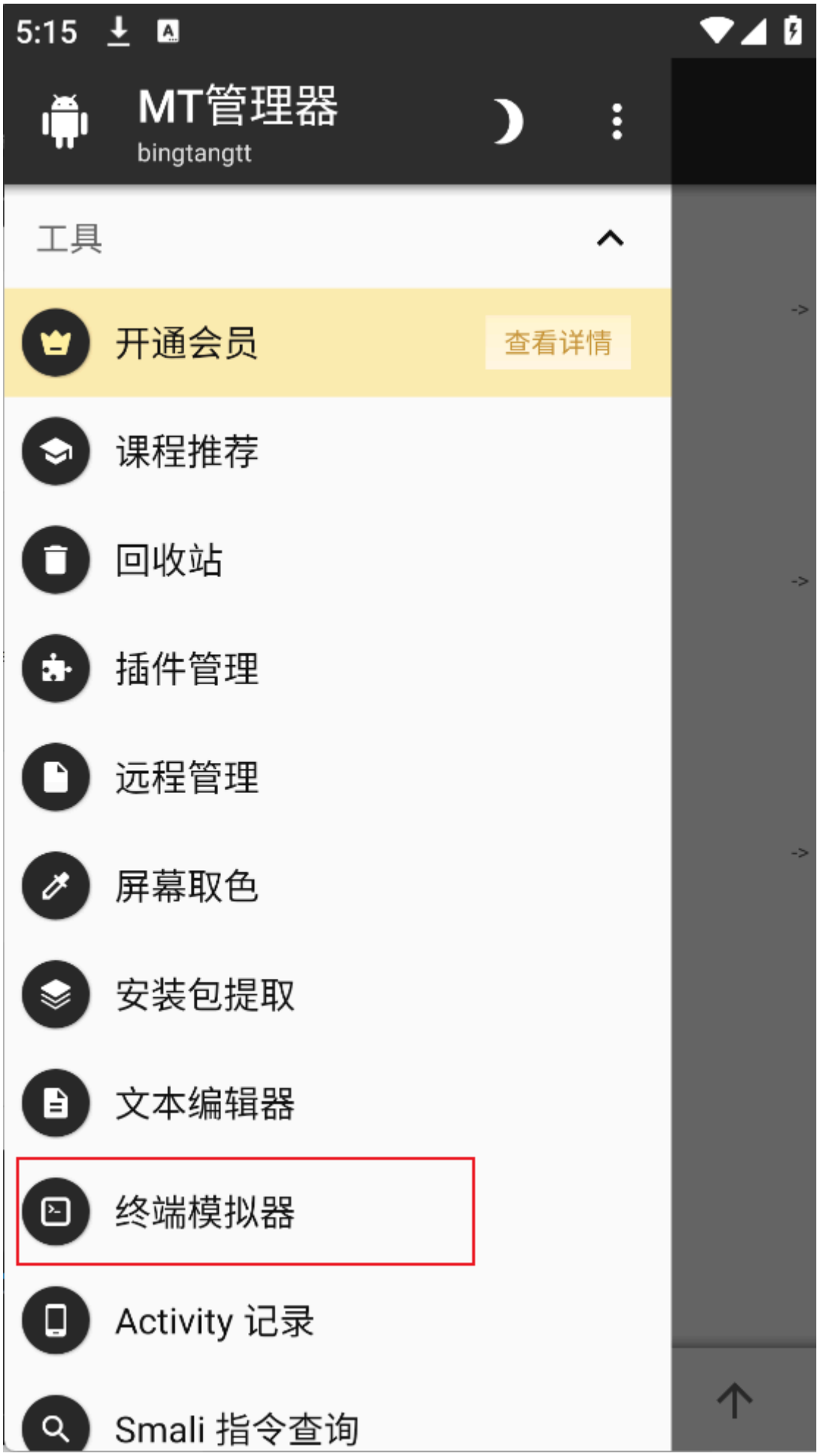
面具安装MagiskHide Props Config模块





选择该文件进行安装，完成后重启

打开mt管理器-终端管理器

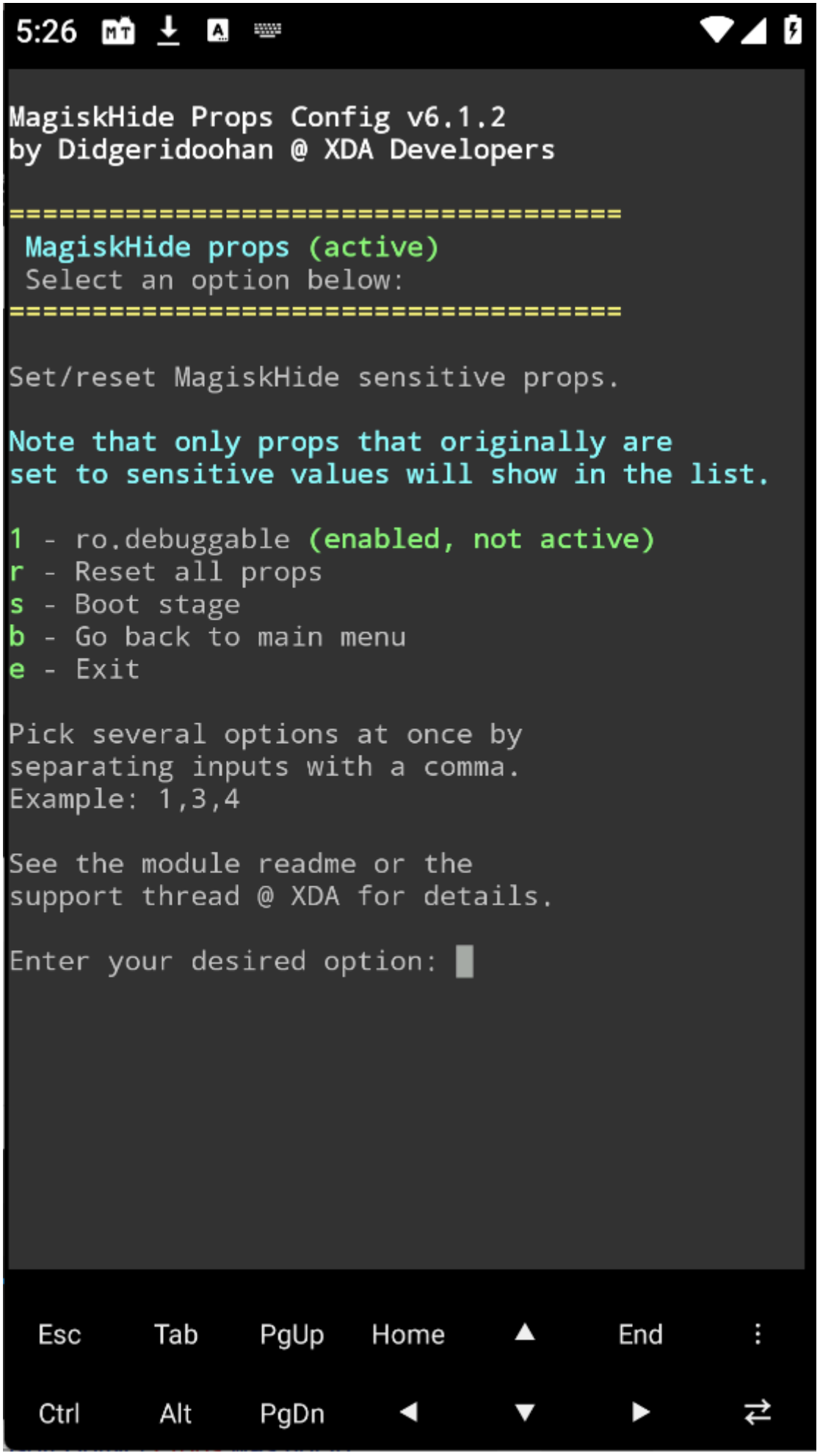

5:15
MT管理器
bingtangtt
工具
开通会员
查看详情
课程推荐
回收站
插件管理
远程管理
屏幕取色
安装包提取
文本编辑器
终端模拟器
Activity 记录
Smali 指令查询

点击输入

```
props
然后确认
```

```
MagiskHide Props Config v6.1.2
by Didgeridoohan @ XDA Developers

========================================
 Select an option below.
========================================

1 - Edit device fingerprint
2 - Force BASIC key attestation
3 - Device simulation (disabled)
4 - Edit MagiskHide props (active)
5 - Add/edit custom props
6 - Delete prop values
7 - Script settings
8 - Collect logs
u - Perform module update check
r - Reset all options/settings
b - Reboot device
e - Exit

See the module readme or the
support thread @ XDA for details.

Enter your desired option:
```

输入4


5:26
MagiskHide Props Config v6.1.2
by Didgeridoohan @ XDA Developers
MagiskHide props (active)
Select an option below:
Set/reset MagiskHide sensitive props.
Note that only props that originally are
set to sensitive values will show in the list.
1 - ro.debuggable (enabled, not active)
r - Reset all props
s - Boot stage
b - Go back to main menu
e - Exit
Pick several options at once by
separating inputs with a comma.
Example: 1,3,4
See the module readme or the
support thread @ XDA for details.
Enter your desired option:
Esc
Tab
PgUp
Home
End
Ctrl
Alt
PgDn

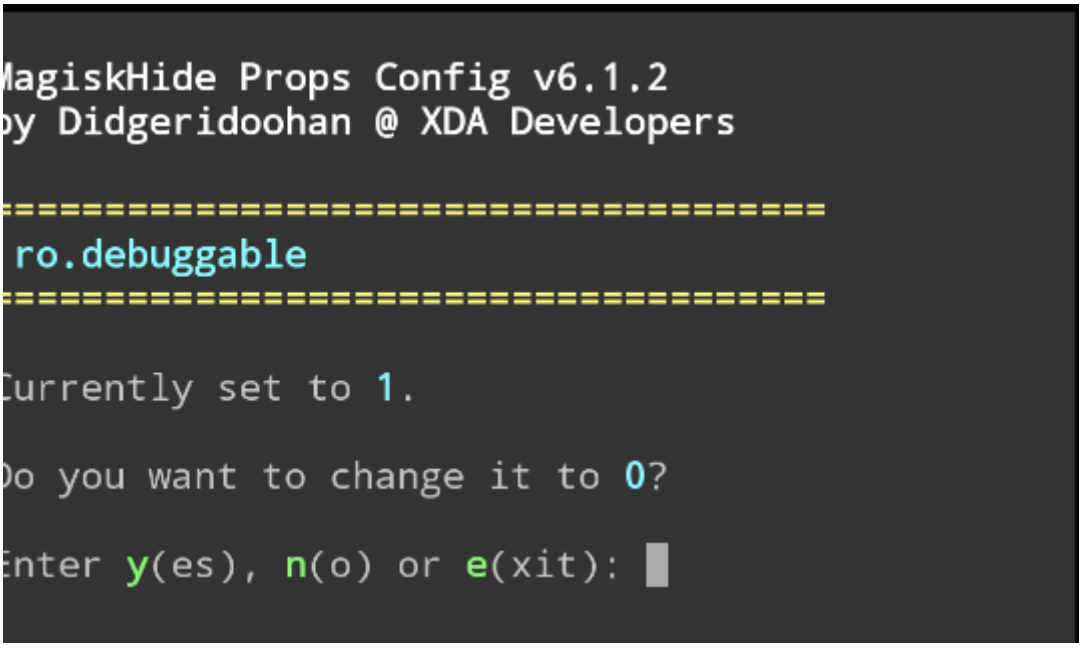
一般来说，在4选项中如果有ro.debuggable那就直接修改
没有的话就选5

- `5` - Add/edit custom props
  - `n` - New custom prop
    - `ro.debuggable`
      - `y` - Yes
        - `1`
          - `y`

吾爱破解论坛
www.52pojie.cn

修改ro.debuggable的值为1

输入1



目前已经是开启状态，所以输入n

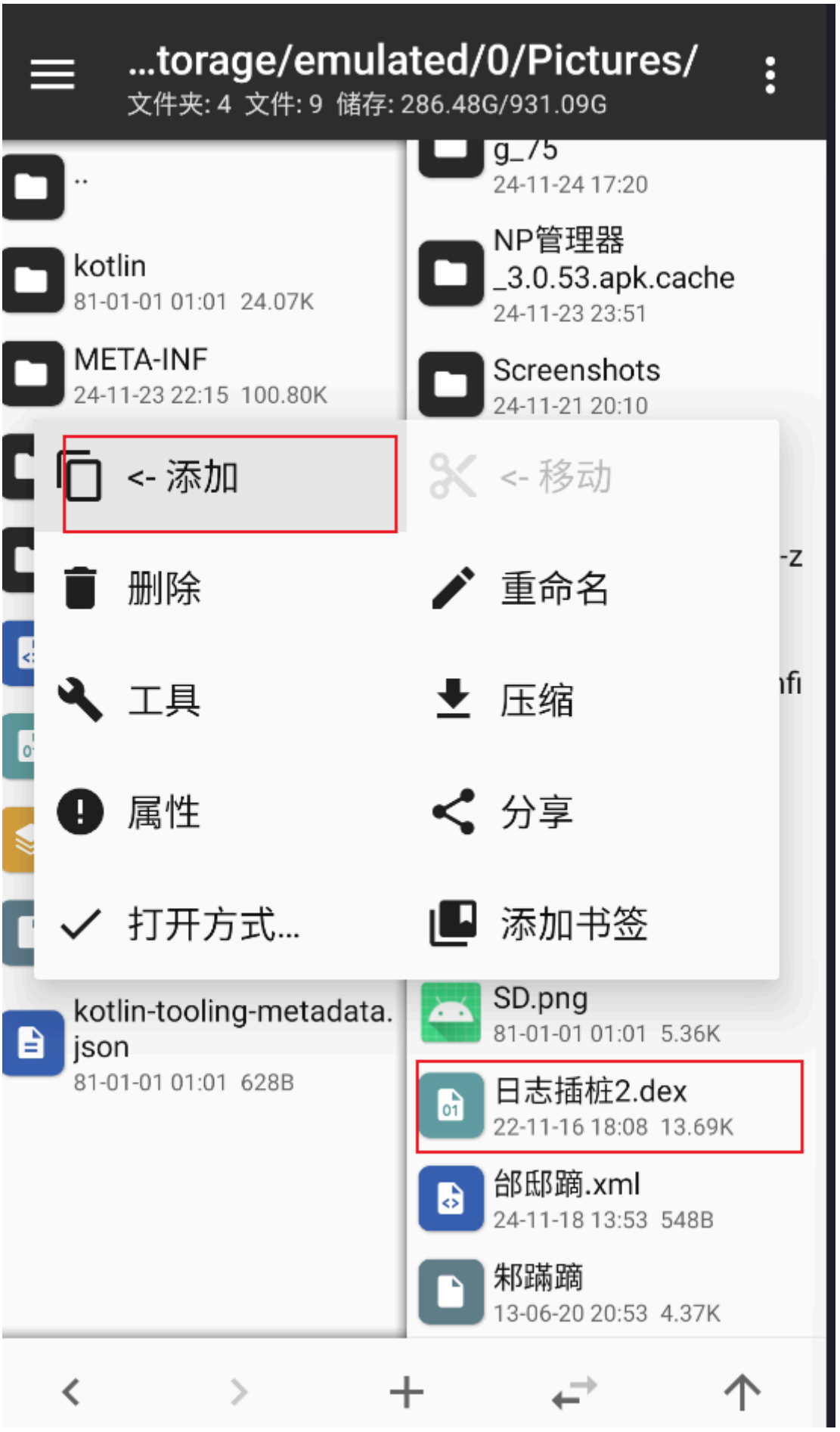
此时会开启永久的debug

2、Log插桩

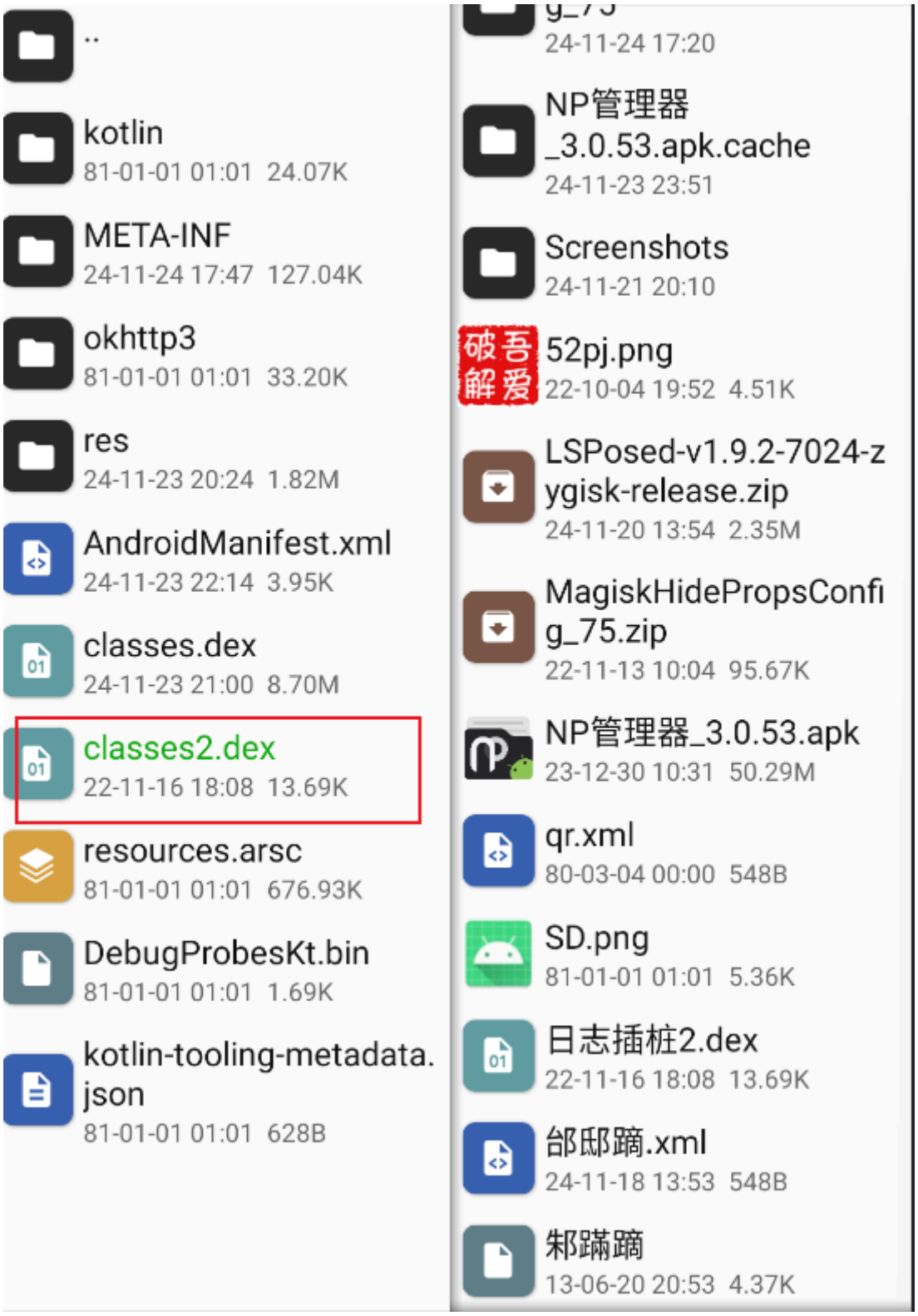
（首先要解决签名校验才能使用此方法）

定义：Log插桩指的是反编译APK文件时，在对应的smali文件里，添加相应的smali代码，将程序中的关键信息，以log日志的形式进行输出。

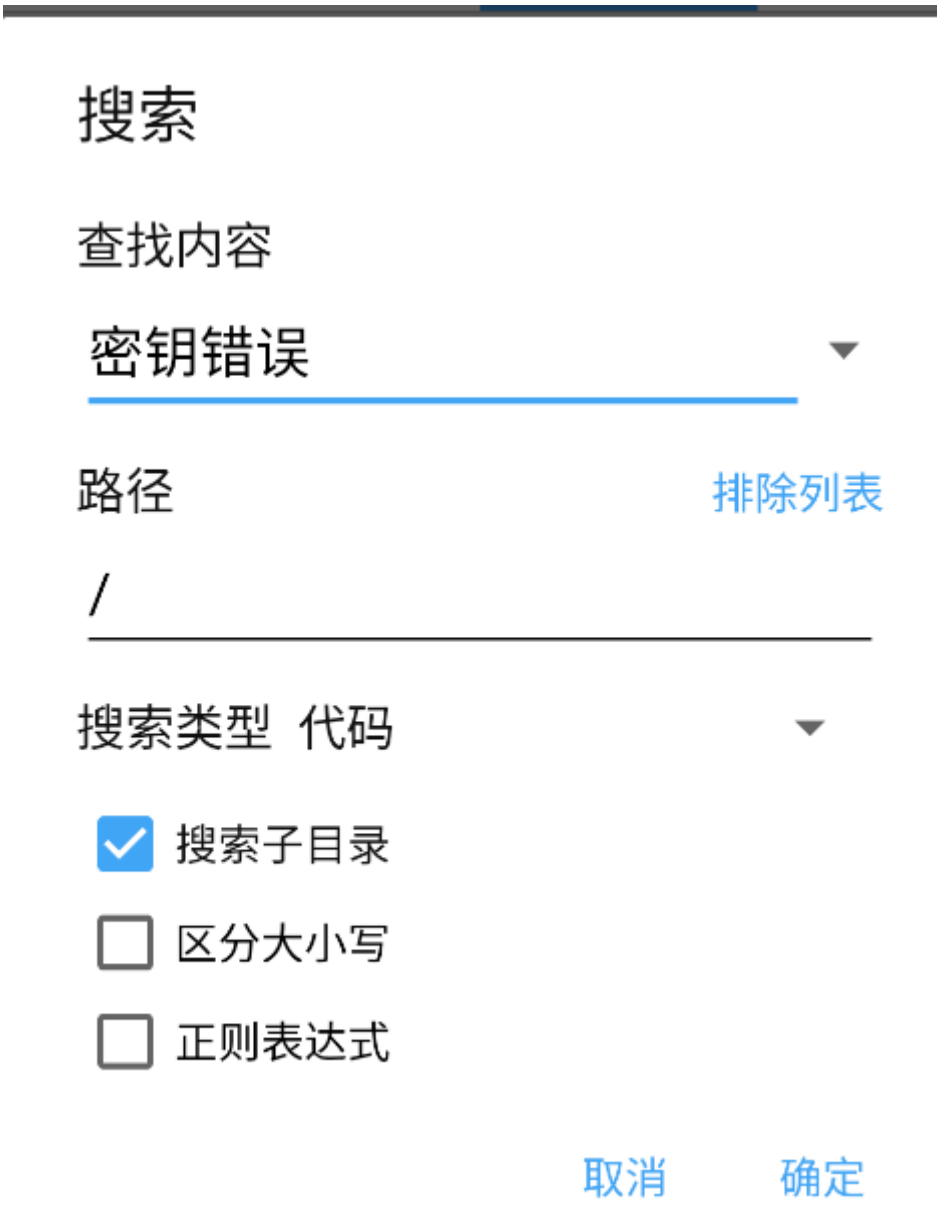
将日志插装dex文件复制到待测试的app包中


...torage/emulated/0/Pictures/
文件夹: 4 文件: 9 储存: 286.48G/931.09G
..
kotlin
81-01-01 01:01 24.07K
META-INF
24-11-23 22:15 100.80K
kotlin-tooling-metadata.
json
81-01-01 01:01 628B
g_75
24-11-24 17:20
NP管理器
_3.0.53.apk.cache
24-11-23 23:51
Screenshots
24-11-21 20:10
<- 添加
<- 移动
删除
重命名
工具
压缩
属性
分享
打开方式...
添加书签
SD.png
81-01-01 01:01 5.36K
日志插桩2.dex
22-11-16 18:08 13.69K
邰邸蹢.xml
24-11-18 13:53 548B
邾蹣蹢
13-06-20 20:53 4.37K

重命名文件



然后在原来classes.dex文件中搜索



找到check函数

```
ChallengeFourth

    invoke-static {v0}, Lcom/zj/wuaipojie/Encode;->encode(Ljava/lang/String;)Ljava/lang/String;

    move-result-object v0

    .line 56
    sget-object v1, Lcom/zj/wuaipojie/util/Base64Utils;->INSTANCE:Lcom/zj/wuaipojie/util/Base64Utils;

    sget-object v2, Lkotlin/text/Charsets;->UTF_8:Ljava/nio/charset/Charset;

    invoke-virtual {v0, v2}, Ljava/lang/String;->getBytes(Ljava/nio/charset/Charset;)[B

    move-result-object v0

    const-string v2, "this.as.java.lang.String).getBytes(charset)"

    invoke-static {v0, v2}, Lkotlin/jvm/internal/Intrinsics;->checkNotNullExpressionValue(Ljava/lang/Object;Ljava/lang/String;)V

    invoke-virtual {v1, v0}, Lcom/zj/wuaipojie/util/Base64Utils;->encodeToString([B)Ljava/lang/String;

    move-result-object v0

    .line 57
    invoke-static {p1, v0}, Lkotlin/jvm/internal/Intrinsics;->areEqual(Ljava/lang/Object;Ljava/lang/Object;)Z

    move-result p1

    return p1
.end method

.method protected onCreate(Landroid/os/Bundle;)V
    .registers 4

    .line 21
```

代码 Smali Simple Fallback Split view

```
// 调用静态方法 checkNotNullExpressionValue，检查 v0（对象引用）是否不为null，如果不为null则继续执行，否则抛出异常。
    // 第二个参数 v2 是一个字符串，通常用于描述如果 v0 为 null 时抛出的异常信息。
    invoke-static {v0, v2}, Lkotlin/jvm/internal/Intrinsics;->checkNotNullExpressionValue(Ljava/lang/Object;Ljava/lang/String;)V

    // 调用虚拟方法 encodeToString 在 v1 指定的对象（假设是 Base64Utils 的实例）上，
    // 将 v0 中的字节数组（[B 表示字节数组类型）编码为 Base64 格式的字符串。
    // 这个方法返回编码后的字符串。
    invoke-virtual {v1, v0}, Lcom/zj/wuaipojie/util/Base64Utils;->encodeToString([B)Ljava/lang/String;

    // 将上一步 invoke-virtual 方法调用的返回值（即编码后的字符串）移动到 v0 寄存器中。
    // 这一步是为了将结果保存到 v0 以便后续使用。
    move-result-object v0
```

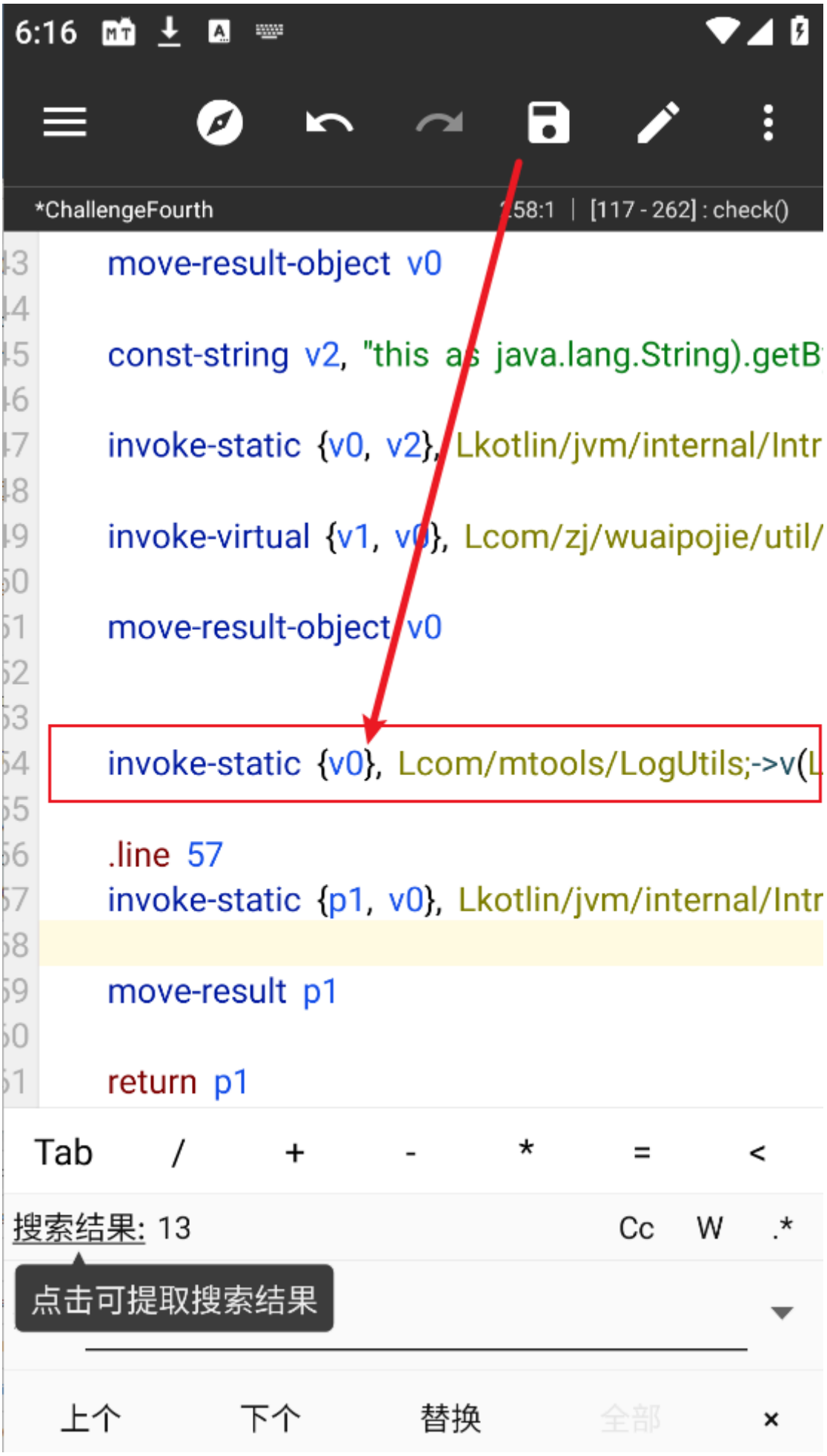
此处最终是将结果的值传给了v0，获取v0的值

调用命令

```
invoke-static {对应寄存器}, Lcom/mtools/LogUtils;->v(Ljava/lang/Object;)V
```

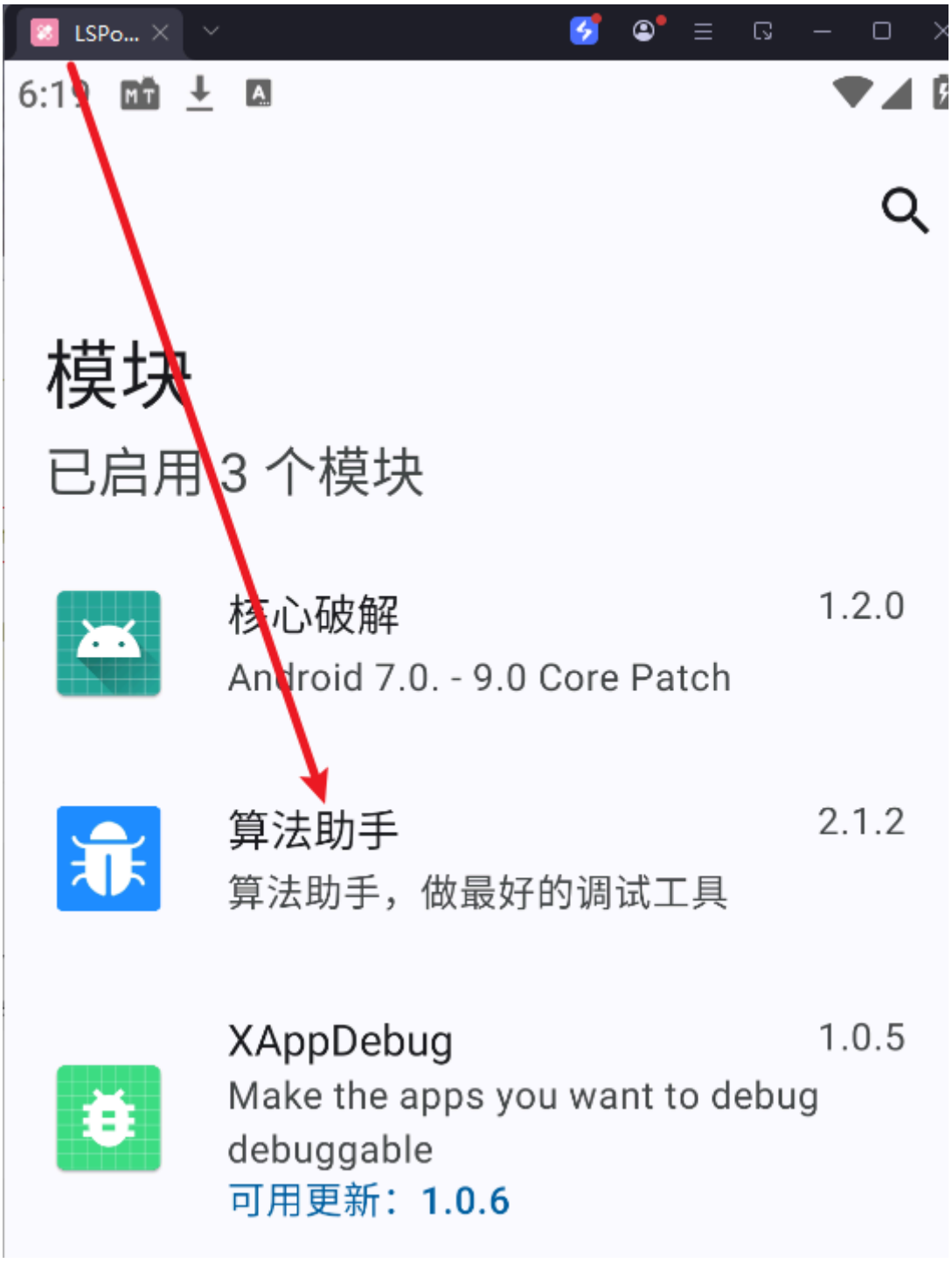
实际改为

```
invoke-static v0, Lcom/mtools/LogUtils;->v(Ljava/lang/Object;)V
```



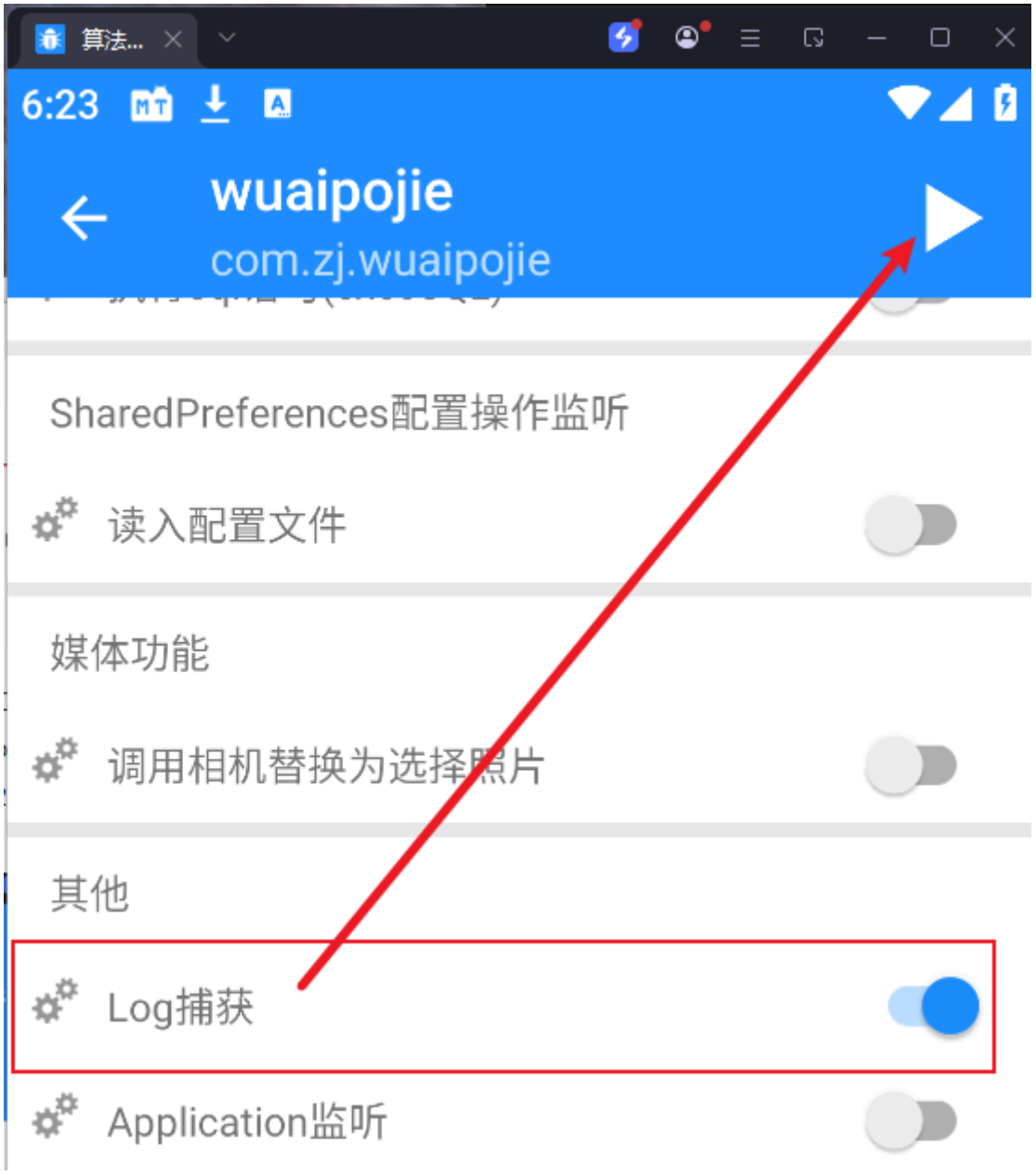
保存后，安装

在LSPosed中 模块安装算法助手

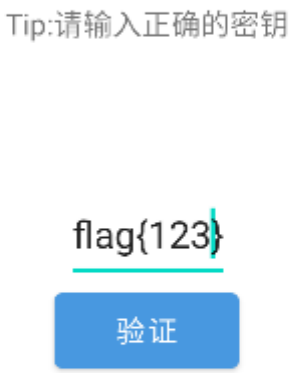

LSPo...
6:19
模块
已启用 3 个模块
核心破解
1.2.0
Android 7.0. - 9.0 Core Patch
算法助手
2.1.2
算法助手，做最好的调试工具
XAppDebug
1.0.5
Make the apps you want to debug debuggable
可用更新：1.0.6

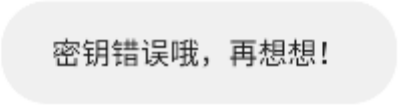
LSPo...
6:21
算法助手
com.junge.algorithmAide
启用模块
系统框架
system
推荐应用
wuaipojie
com.zj.wuaipojie
版本：1.0
开发者助手
com.toshiba_dealin.developer

然后打开算法助手

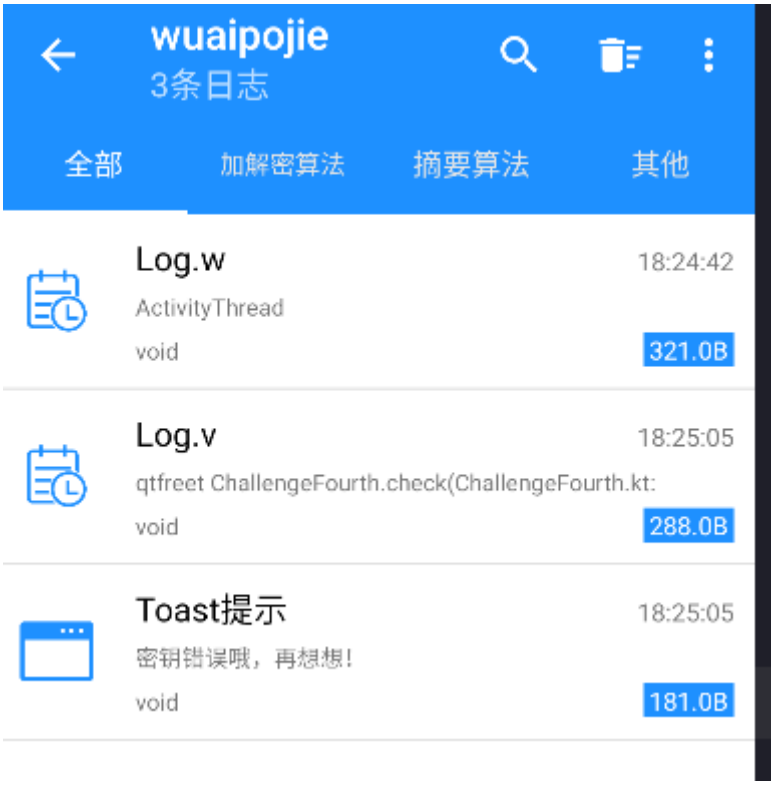


点击后会自动启动app 进入关卡，输入任意密钥





打开算法助手app的日志



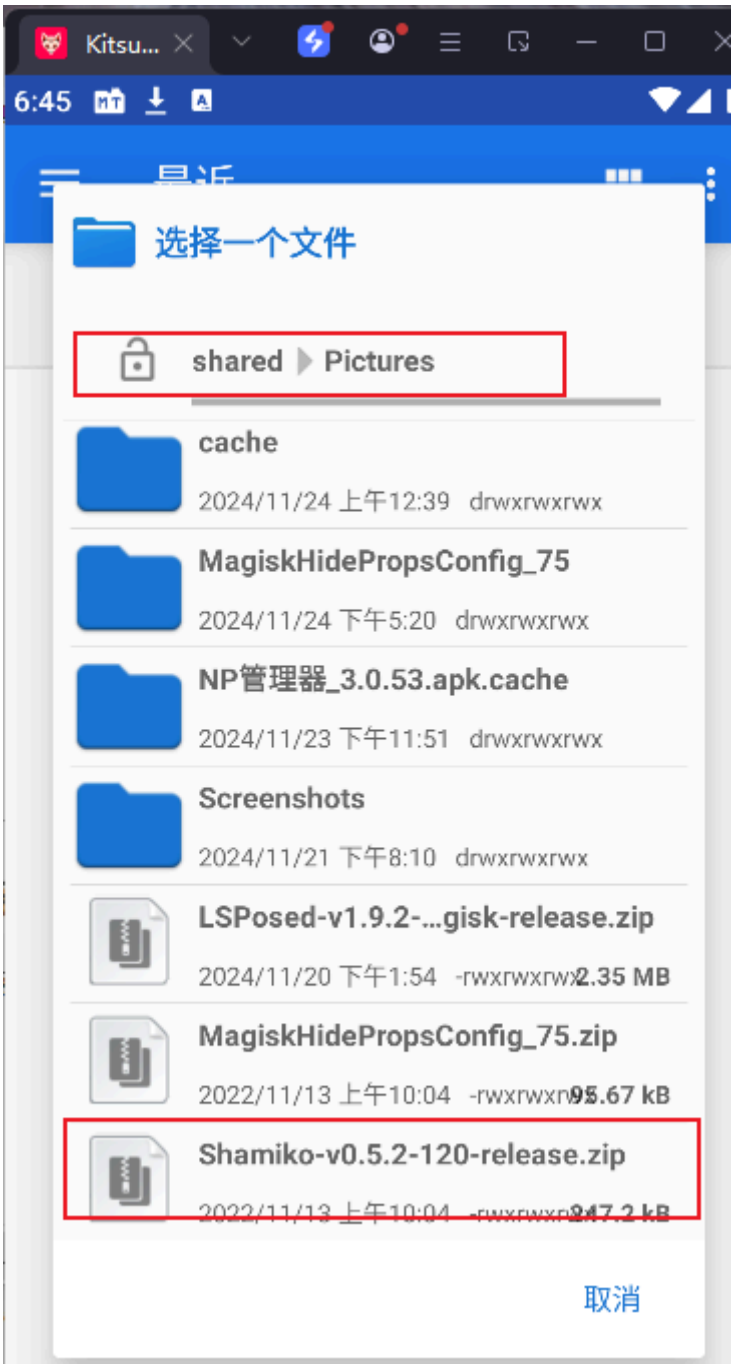
查看log.v

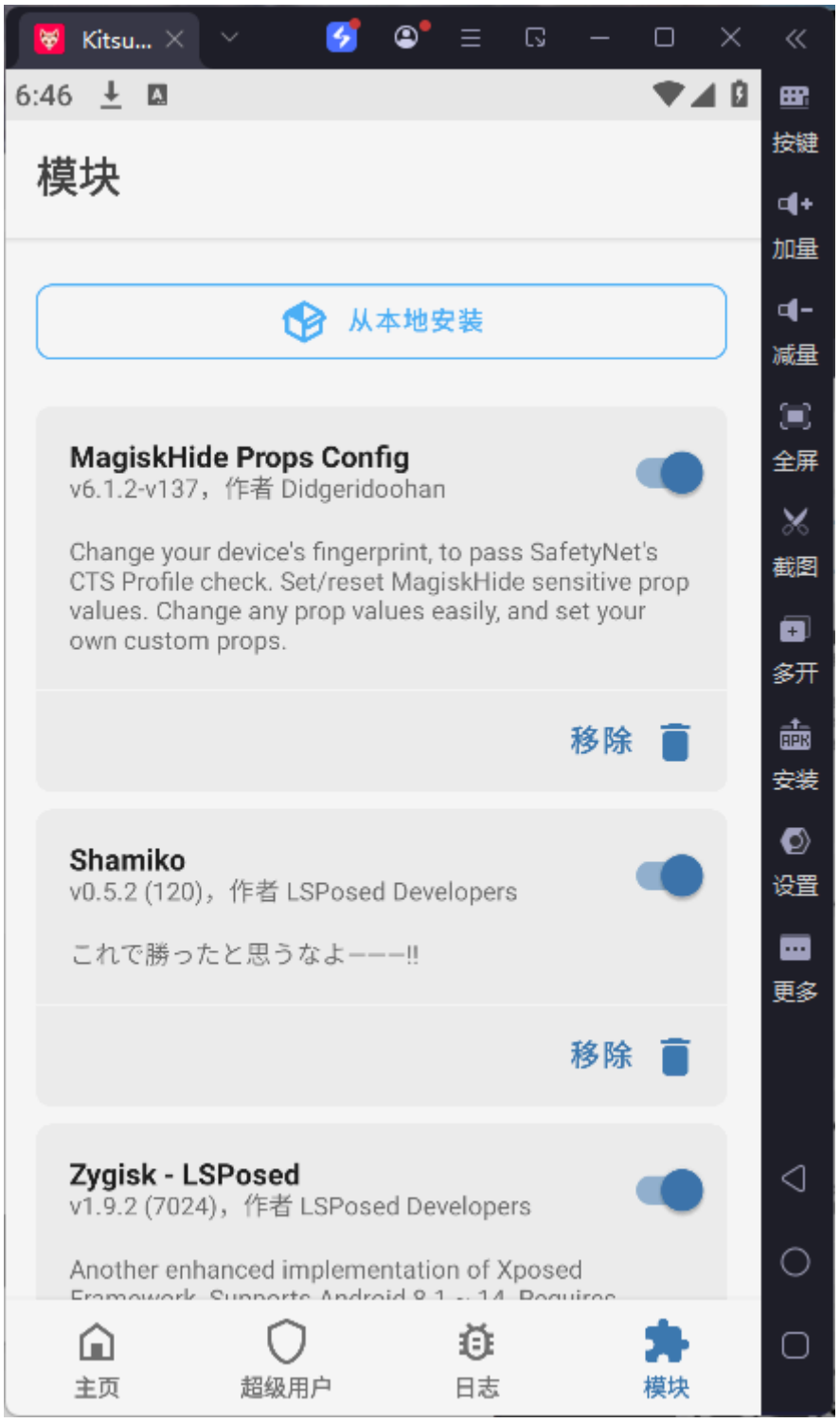
6:29
wuaipojie
Log.v
时间：2024-11-24 18:25:05
类名：android.util.Log
日志名称：Log.v
参数1
参数类型:java.lang.String
参数值：qtfreet
ChallengeFourth.check(ChallengeFourth.kt:56)
参数2
参数类型:java.lang.String
参数值：DAsG
返回结果类型:void
返回结果值：void
调用堆栈：
at F.fETsITVotOdGd.Rf.xtT.XC_MethodReplacement.beforeHookedMethod(Unknown Source:0)
at F.fETsITVotOdGd.Rf.xtT.XposedBridge$LegacyApiSupport.handleBefore(Unknown Source:24)
at J.callback(Unknown Source:179)
at LSPHooker_.v(Unknown Source:11)
at com.mtools.Logger.logString(Logger.java:49)

测试

3、隐藏app的root权限

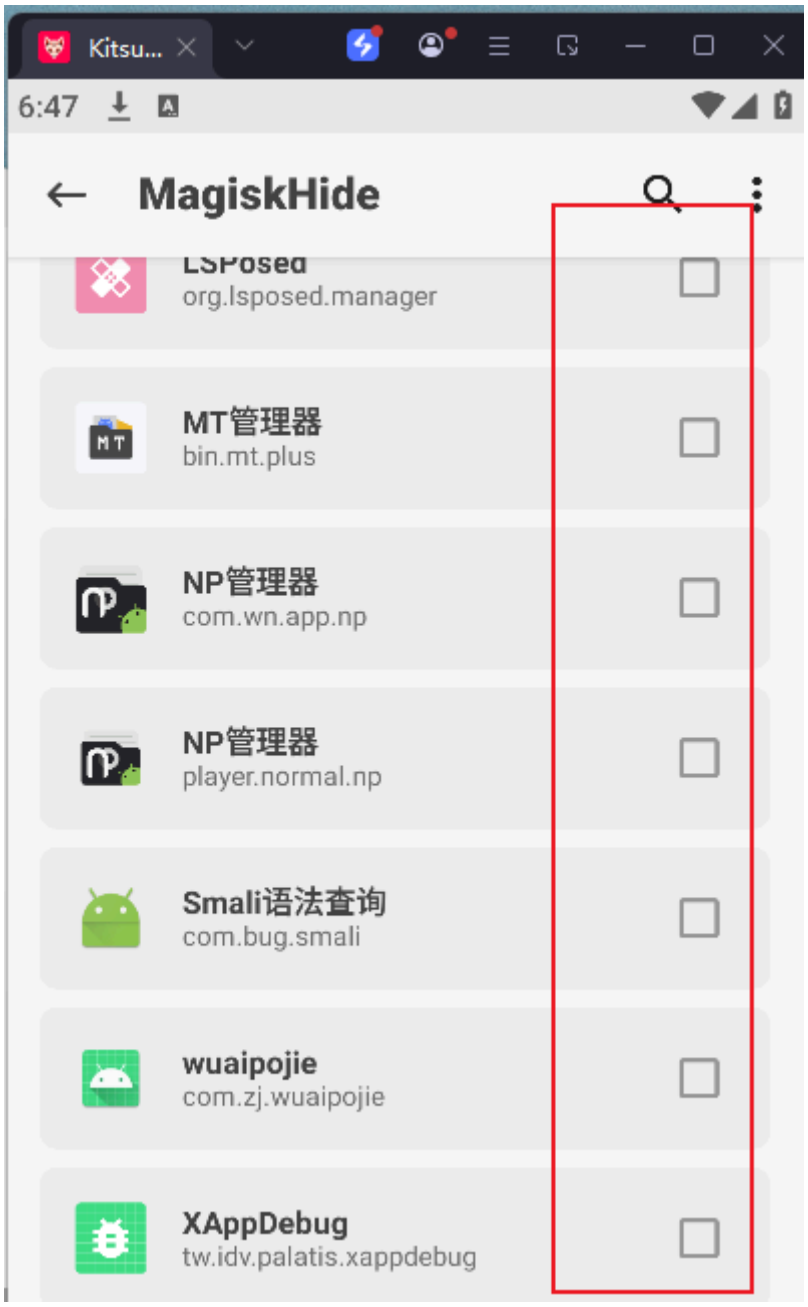
面具安装使用Shamiko


Kitsu...
6:45
选择一个文件
shared
Pictures
cache
2024/11/24 上午12:39 drwxrwxrwx
MagiskHidePropsConfig_75
2024/11/24 下午5:20 drwxrwxrwx
NP管理器_3.0.53.apk.cache
2024/11/23 下午11:51 drwxrwxrwx
Screenshots
2024/11/21 下午8:10 drwxrwxrwx
LSPosed-v1.9.2-...gisk-release.zip
2024/11/20 下午1:54 -rwxrwxrwx 2.35 MB
MagiskHidePropsConfig_75.zip
2022/11/13 上午10:04 -rwxrwxrwx 95.67 kB
Shamiko-v0.5.2-120-release.zip
取消


Kitsu...
6:46
模块
从本地安装
MagiskHide Props Config
v6.1.2-v137，作者 Didgeridoohan
Change your device's fingerprint, to pass SafetyNet's CTS Profile check. Set/reset MagiskHide sensitive prop values. Change any prop values easily, and set your own custom props.
移除
Shamiko
v0.5.2 (120)，作者 LSPosed Developers
これで勝ったと思うなよ−−−!!
移除
Zygisk - LSPosed
v1.9.2 (7024)，作者 LSPosed Developers
Another enhanced implementation of Xposed
主页
超级用户
日志
模块
按键
加量
减量
全屏
截图
多开
安装
设置
更多

Kitsu...
6:46
主页
警告！ 您正在使用 Magisk 的非官方版本，该版本未得到 Magisk 作者的正式支持。 此版本可能包含不稳定的修改，可能会损害您的设备或危及您的安全。 请勿报告任何问题或向 Magisk 官方渠道请求任何帮助。 如果您使用此版本但不知道这是非官方版本，请切换到官方 Magisk github.com/topjohnwu/Magisk
不再显示
Magisk
安装
当前 R6687BB53-kitsune (27001)
Zygisk 是
Ramdisk 否
App
安装
最新 R6687BB53-kitsune (27001)
当前 R6687BB53-kitsune (27001)
包名 io.github.huskydg.magisk
主页
超级用户
日志
模块

勾选app后 会隐藏app的root权限


Kitsu...
6:47
MagiskHide
LSPosed
org.lsposed.manager
MT管理器
bin.mt.plus
NP管理器
com.wn.app.np
NP管理器
player.normal.np
Smali语法查询
com.bug.smali
wuaipojie
com.zj.wuaipojie
XAppDebug
tw.idv.palatis.xappdebug