

目录

前言	1.1
Android逆向动态调试概览	1.2
AS动态调试Smali步骤	1.3
确保安卓已root	1.3.1
调试工具选择	1.3.2
反编译出smali	1.3.3
app可调试	1.3.4
调试方式启动app	1.3.5
AS中导入smali代码	1.3.6
给smali加断点	1.3.7
配置AS项目	1.3.8
adb可获取进程列表	1.3.9
AS调试app进程	1.3.10
其他分析调试工具	1.4
frida	1.4.1
AndBug	1.4.2
redexer	1.4.3
Fino	1.4.4
心得	1.5
附录	1.6
参考资料	1.6.1

Android逆向：动态调试

- 最新版本: v0.8
- 更新时间: 20221027

简介

介绍Android逆向开发期间，如何动态调试apk的smali代码。此处以YouTube为例，介绍主要步骤：确保安卓已root、调试工具选择、反编译出smali代码、确保app可调试、调试方式启动app、Android Studio中导入smali代码、给smali加断点、配置Android Studio项目、确保adb可获取进程列表、Android Studio调试app进程；以及一些心得。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

HonKit源码

- [crifan/android_re_dynamic_debug: Android逆向：动态调试](#)

如何使用此HonKit源码去生成发布为电子书

详见：[crifan/honkit_template: demo how to use crifan honkit template and demo](#)

在线浏览

- [Android逆向：动态调试 book.crifan.org](#)
- [Android逆向：动态调试 crifan.github.io](#)

离线下载阅读

- [Android逆向：动态调试 PDF](#)
- [Android逆向：动态调试 ePUB](#)
- [Android逆向：动态调试 Mobi](#)

版权和用途说明

此电子书教程的全部内容，如无特别说明，均为本人原创。其中部分内容参考自网络，均已备注了出处。如有版权，请通过邮箱联系我 `admin 艾特 crifan.com`，我会尽快删除。谢谢合作。

各种技术类教程，仅作为学习和研究使用。请勿用于任何非法用途。如有非法用途，均与本人无关。

鸣谢

感谢我的老婆陈雪的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

更多其他电子书

本人 `crifan` 还写了其他 100+ 本电子书教程，感兴趣可移步至：

[crifan/crifan_ebook_readme: Crifan的电子书的使用说明](#)

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2022-10-27 16:28:18

Android逆向动态调试概览

关于安卓逆向，之前已通过[安卓应用的安全和破解 \(crifan.org\)](#)进行了一定的介绍。

而安卓逆向期间，除了[静态分析](#)去研究反编译的代码外，想要了解安卓app的底层机制和逻辑，需要通过[动态调试](#)。

而目前安卓的动态调试的主要的方法是：用[Android Studio](#)去调试apk反编译得到的smali代码。

此教程主要就是介绍这方面的详细内容。

crifan.org，使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2022-10-27 14:36:35

AS动态调试Smali步骤

TODO:

【已解决】安卓YouTube逆向：搭建安卓apk动态调试环境

此处介绍， Mac 中如何用 Android Studio 去调试 YouTube 的 apk 的 smali 代码的具体步骤。

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-27 15:48:46

确保安卓已root

确保安卓设备已root:

此处之前买的二手的 Google Pixel 3 , 已root:



自带安装了：

- Magisk Manager
 - 图
 -





10:35 G 100%

主页

刷新 

**Magisk** 安装

当前 25.2 (25200)
Zygisk 否
Ramdisk 是

**App** 安装

最新 25.2 (25200) (33)
当前 25.2 (25200)
包名 com.topjohnwu.magisk

 卸载 Magisk

支持开发

Magisk 将一直保持免费且开源，向开发者捐赠以表示支持。

关注我们

@topjohnwu  

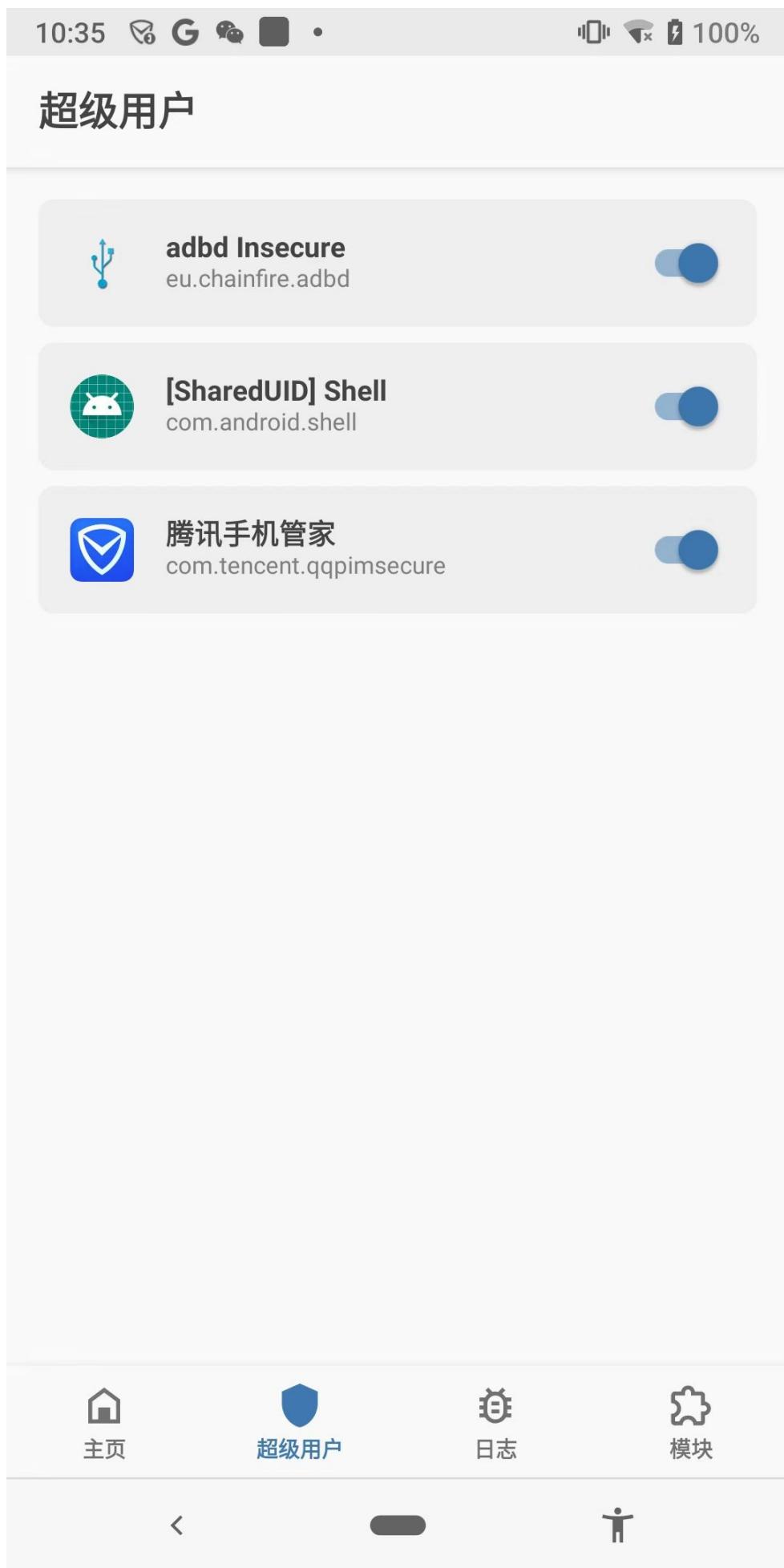
 主页  超级用户  日志  模块

<  >

- 升级后是： Magisk
 - 内部包含：
 - Magisk (的框架)
 - App = Manager : 管理配置界面的app

且已给相关app授权了root权限：

- (对应着) adb shell 中首次 su 后，即可申请 root权限，允许后，此处就会出现：
 - Shell = com.android.shell
 -



调试工具选择

- 调试安卓apk 的工具/环境

- Android Studio + smalidea 插件: 调试 smali
- JD-GUI : 调试 Smali
- IDA : 调试 dex
- Qtrace

此处选择, 相对易用的:

- Android Studio + smalidea 插件: 调试 smali

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-27 14:42:13

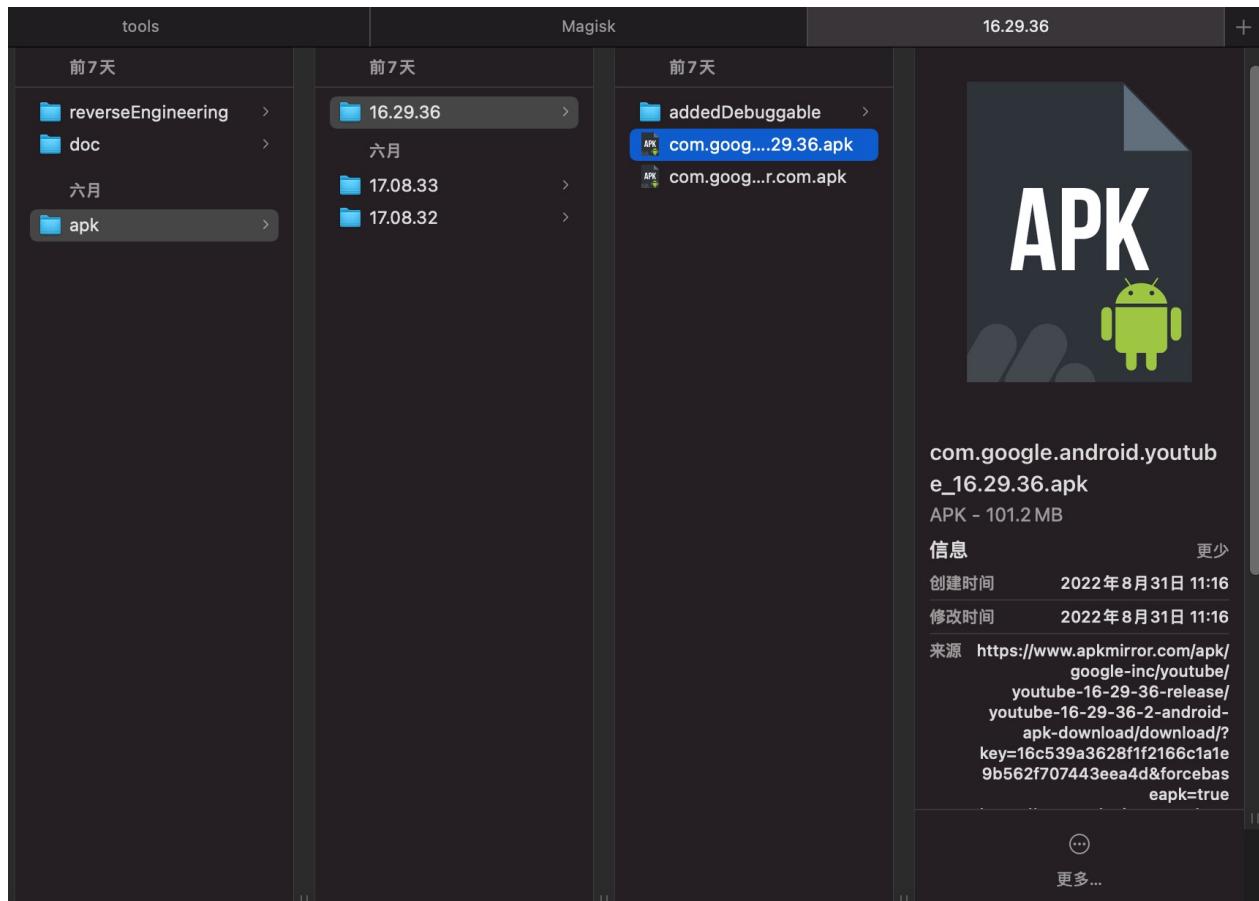
反编译出smali

反编译 apk , 得到 smali 源码

常见反编译工具：

- [Apktool](#)
 - 直接1步：apk to smali
- [baksmali](#)
 - 要2步：先 apk to dex, 再 dex to smali

此处用 apktool 去反编译YouTube的apk：



去反编译得到包含 smali 源码的目录：

```
apktool d --use-aapt2 ./apk/16.29.36/com.google.android.youtube_16.29.36.apk
```

如果要指定输出目录，也可以加上 -o

```
apktool d --use-aapt2 ./apk/16.29.36/com.google.android.youtube_16.29.36.apk -o com.google.android.youtube_16.29.36_aapt2
```

输出的目录的内容：

```

<uses-permission android:name="com.sonyericsson.home.permission.BLUETOOTH_AUDIO"/>
<uses-permission android:name="com.sonymobile.home.permission.PROVIDER_INSERT_BADGE"/>
<uses-feature android:name="android.hardware.camera" android:required="false"/>
<uses-feature android:name="android.hardware.screen.portrait" android:required="false"/>
<uses-feature android:name="android.hardware.telephony" android:required="false"/>
<uses-feature android:name="android.hardware.microphone" android:required="false"/>
<uses-feature android:name="android.hardware.location" android:required="false"/>
<uses-feature android:name="android.hardware.location.gps" android:required="false"/>
<uses-feature android:name="android.hardware.location.network" android:required="false"/>
<permission android:name="com.google.android.youtube.permission.C2D_MESSAGE" android:protectionLevel="signature|signatureOrSystem"/>
<uses-feature android:gEsVersion="0x00020000" android:required="true"/>
<application android:debuggable="true" android:allowBackup="true" android:backupAgent="com.google.android.apps.youtube.config.BuildType$YouTubeBackupAgent"/>
<meta-data android:name="android.max_aspect" android:value="2.1"/>
<meta-data android:name="com.google.android.backup.api_key" android:value="AEdPqrEAAAIXi"/>
<meta-data android:name="to.dualscreen" android:value="true"/>
<meta-data android:name="com.google.android.apps.youtube.config.BuildType" android:value="RELEASE"/>
<activity android:exported="false" android:name="com.google.android.libraries.youtube.youtube.MainActivity"/>
<service android:exported="false" android:foregroundServiceType="dataSync" android:name="com.google.android.apps.youtube.youtube.DownloadService"/>
<receiver android:enabled="false" android:exported="false" android:name="com.google.android.apps.youtube.youtube.DownloadService$DownloadReceiver"/>
</receiver>
<service android:exported="false" android:name="com.google.android.libraries.youtube.playback.PlaybackService"/>
<receiver android:exported="true" android:name="com.google.android.libraries.youtube.playback.PlaybackService$PlaybackReceiver"/>
</receiver>
<activity android:exported="false" android:label="@string/gallery_activity_title" android:name="com.google.android.libraries.youtube.gallery.GalleryActivity"/>
<activity android:exported="false" android:name="com.google.android.libraries.youtube.accessibility.AccessibilityService"/>
<activity android:exported="false" android:name="com.google.android.libraries.youtube.companion.CompanionActivity"/>
<activity android:exported="true" android:launchMode="singleInstance" android:name="com.google.android.apps.youtube.youtube.MainActivity"/>
</activity>
<intent-filter>
<action android:name="android.intent.action.MEDIA_BUTTON"/>
</intent-filter>
<intent-filter>
<action android:name="android.intent.action.VIEW"/>
<category android:name="android.intent.category.DEFAULT"/>
</intent-filter>

```

- 一个或多个 smali 目录
 - 注：每个 smali 目录，对应着 apk 内部的 dex 文件
- AndroidManifest.xml
 - 文本模式的，有 apk 核心的配置和参数

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook 最后更新：2022-10-27 16:27:08

app可调试

确保app 可调试 = debuggable :

让app可调试，有多种方式：

- 针对 特定app 可调试
 - app代码中 `AndroidManifest.xml` 加上 `android:debuggable="true"`
 - 思路：往往是（apktool 等）反编译得到源码，修改后，再重新打包回去
 - 缺点：对于稍微复杂点的apk，往往重新打包的过程中会出错，会很麻烦
- 全局可调：安卓系统中所有app进程都可调试
 - 实现思路：修改系统全局属性 `ro.debuggable`
 - 具体方法
 - 修改 `boot.img`，重新刷入
 - 说明：比较折腾的一种。不推荐，虽然也可以一劳永逸，但是比较折腾，刷机操作失误还容易变砖
 - `mprop`
 - 已基本失效，放弃
 - Xposed 插件
 - xinstaller
 - Xdebbagable
 - BuildProp Enhancer
 - XDebug
 - Magisk
 - Magisk
 - 命令行：`magisk resetprop ro.debuggable 1`
 - 缺点：重启后失效
 - Magisk 插件
 - MagiskHide Props Config
 - 用 props 去修改 `ro.debuggable=1`

此处选择，相对方便操作和效果更好的：

MagiskHide Props Config

- Magisk的插件
 - MagiskHide Props Config
 -

10:36 G 100%

模块

 从本地安装

ADB Root
v1, 作者 Denis Efremov (@evdenis) 

Allows to "adb root" regardless of props settings and skips usb keys auth. Kind of "adbd Insecure" as a Magisk Module. Arm64 only.

移除 

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 

Riru - EdXposed
v0.5.2.2_4683-master, 作者 solohsu, MlgmXyysd 

Another enhanced implementation of Xposed Framework. Supports Android 8.0, 8.1, 9, 10, 11 or above. Requires Riru v23 or above installed. Telegram: @EdXposed

移除 

 主页  超级用户  日志  模块

<  > 

安装后，去安卓设备端的命令行（adb shell 进去）运行：

```
props
```

根据提示，去（先新增再）修改为： ro.debuggable=1

核心的选择是：

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

最后查看属性，确保修改成功：

```
> adb shell getprop ro.debuggable
1
```

TODO：

- 【已解决】安卓逆向：如何让YouTube的apk可以被调试
 - 【已解决】安卓让apk可调试：修改全局系统属性ro.debuggable
 - 【已解决】安卓逆向：用MagiskHide Props Config实现全局apk可调试
 - 【未解决】安卓逆向：让apk可调试之通过Xposed插件
-

AndroidManifest.xml加debuggable属性

顺带介绍另外一种，虽然不推荐，但是大家常提到的方式：

- app代码中 AndroidManifest.xml 加上 android:debuggable="true"

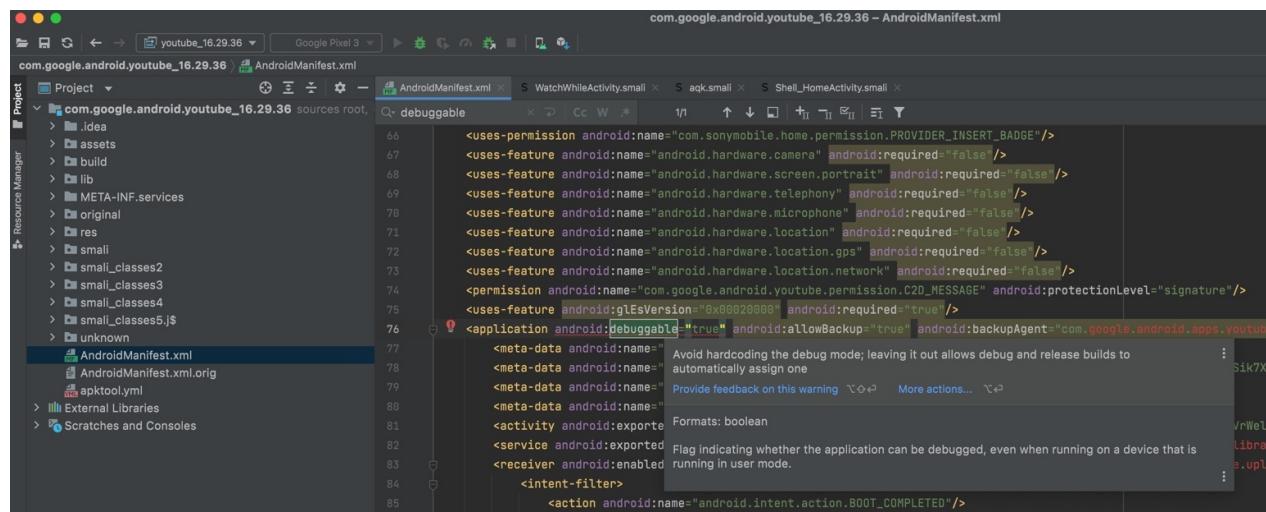
思路：

把apktool反编译后的代码中的

```
AndroidManifest.xml
```

去修改，加上：

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



然后再用apktool重新打包出apk（有失败的风险）

安装到安卓手机中，成为可调试的app

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-27 15:41:34

调试方式启动app

以调试方式启动app：

- 目的：这样启动的app，才可以被调试，方便被调试

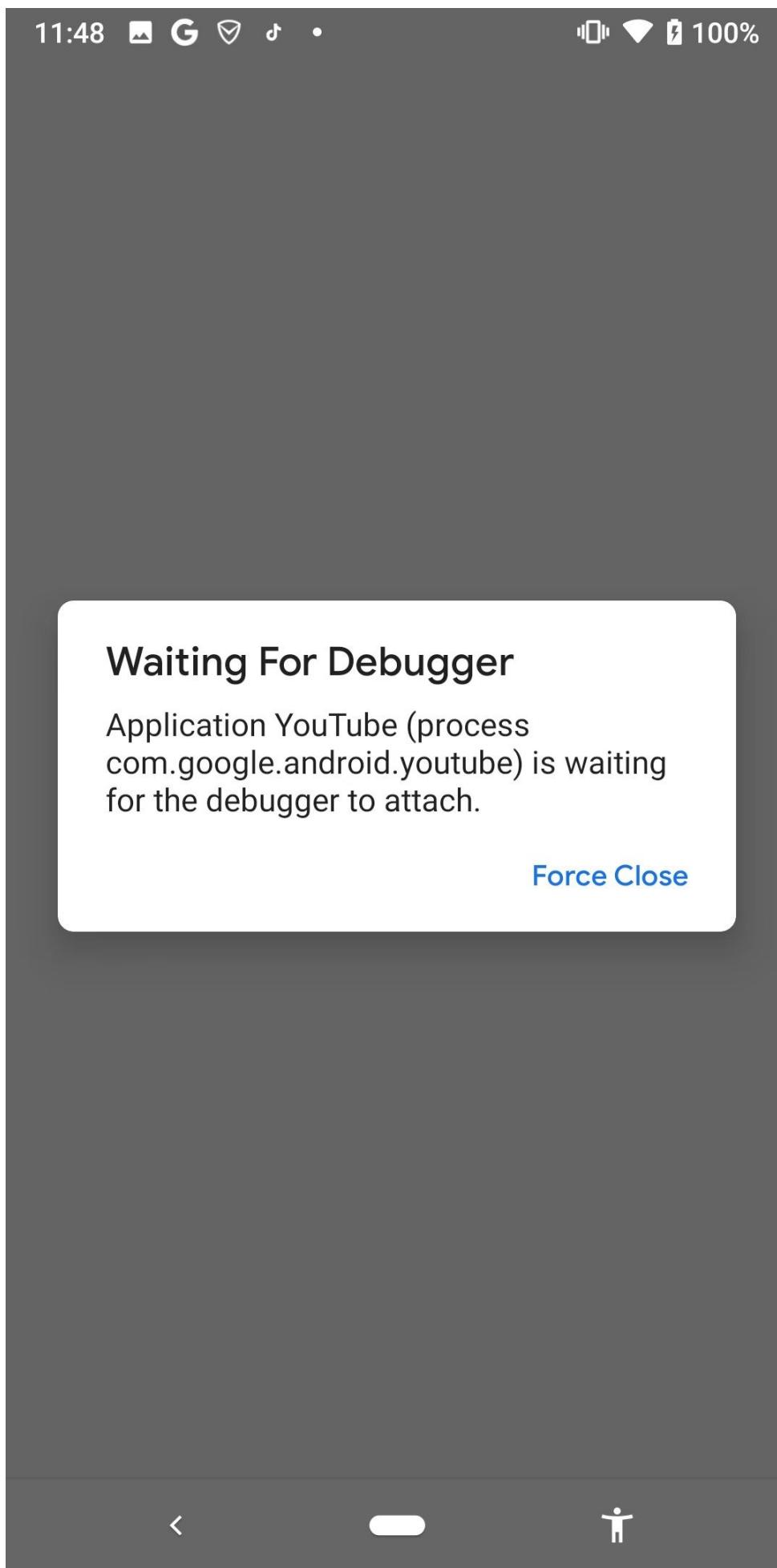
此处以调试方式启动 YouTube 的app：

```
adb shell am start -D -n com.google.android.youtube/com.google.android.apps.youtube.app.application.Shell_HomeActivity
```

说明：

- -D : 表示 debug 模式
- com.google.android.youtube/com.google.android.apps.youtube.app.application.Shell_HomeActivity
 - com.google.android.youtube
 - app包名=package
 - com.google.android.apps.youtube.app.application.Shell_HomeActivity
 - app的主页面=MainActivity

以调试模式启动后的效果：



启动后，app没有立刻正常运行，而是显示 Waiting For Debugger

并且：此界面不会消失，直到对应的debugger调试器连接上，开始调试，此界面才消失，才继续开始运行app

说明：

如何获取app的包名

Youtube 包名： com.google.android.youtube

- 方式1： pm

举例：

```
~ adb shell pm list packages -f | grep youtube  
package:/data/app/com.google.android.youtube-9Nw_99XIZ2jZh7Lyor2SKQ /base.apk com.google.android.youtube
```

- 方式2： aapt

```
aapt dump badging com.google.android.youtube_16.29.36.apk | grep package  
package: name='com.google.android.youtube' versionCode='1522263488' versionName='16.29.36' compileSdkVersion='31' compileSdkVersionCodename='12'
```

如何获取app的MainActivity

app的首页的 activity，一般被叫做 MainActivity

一般情况是： apktool 逆向导出的 xml 格式的 AndroidManifest.xml 中，找到 android.intent.action.MAIN 所属于的 activity，就是 MainActivity

-》此处YouTube的情况稍微特殊点：

AndroidManifest.xml

```
<activity exported="true" name="com.google.android.apps.youtube.app.application.Shell_HomeActivity"  
theme="@style/Theme.YouTube.Launcher"/>  
<activity-alias exported="true" name="com.google.android.youtube.HomeActivity" targetActivity="  
com.google.android.apps.youtube.app.application.Shell_HomeActivity"/>  
<activity-alias exported="true" name="com.google.android.youtube.app.application.Shell$HomeActivity"  
targetActivity="com.google.android.apps.youtube.app.application.Shell_HomeActivity"/>  
<activity-alias exported="true" name="com.google.android.youtube.app.honeycomb.Shell$HomeActivity"  
targetActivity="com.google.android.apps.youtube.app.application.Shell_HomeActivity">  
    <intent-filter>  
        <action name="android.intent.action.MAIN"/>  
        <category name="android.intent.category.DEFAULT"/>  
        <category name="android.intent.category.LAUNCHER"/>  
    </intent-filter>
```

找到：

- android.intent.action.MAIN

所属的activity是：

- com.google.android.youtube.app.honeycomb.Shell\$HomeActivity

其是个别名 alias，对应着真正的activity = MainActivity 是：

- com.google.android.apps.youtube.app.application.Shell_HomeActivity

TODO：

- 【已解决】获取安卓apk应用的app的主界面activity即MainActivity

AS中导入smali代码

导入带 smali 源码的目录到 Android Studio 作为新项目：

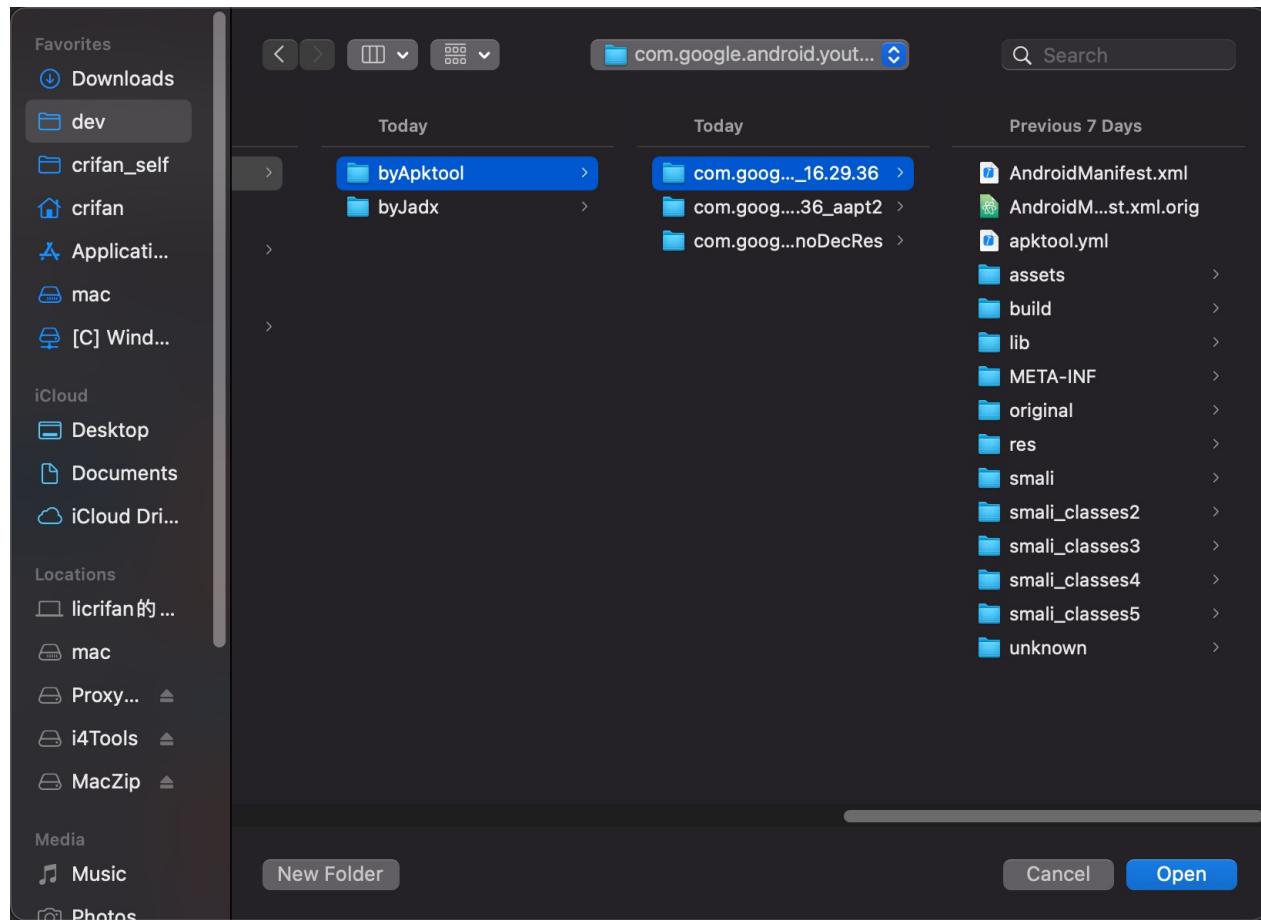
导入smali的源码：

新版 Android Studio 的 Welcome to Android Studio 欢迎对话框中，点击：

- Open

◦

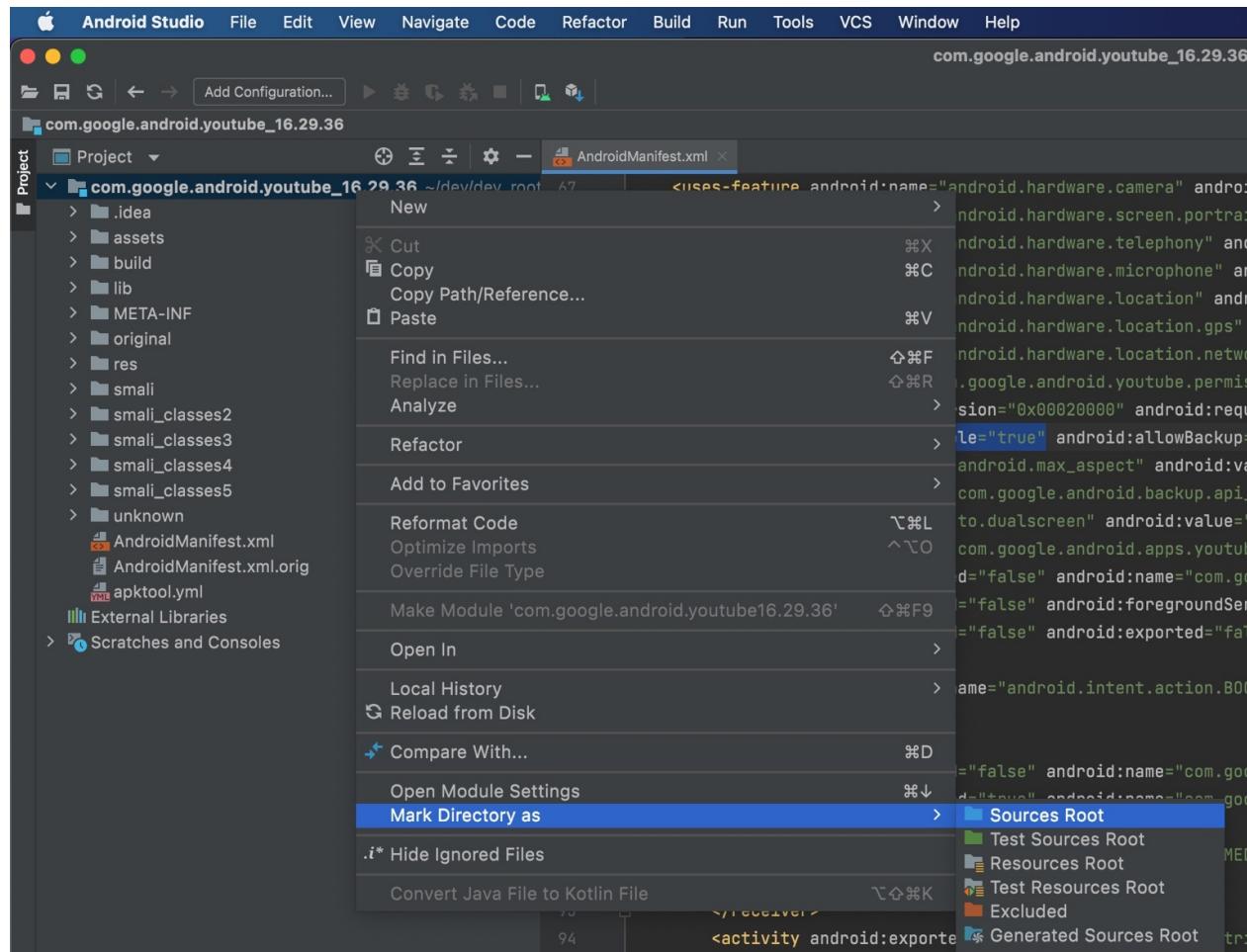
去选择对应的， Apktool 反编译后的输出的目录：



导入后，即可：

然后把根目录设置为源代码根目录：

Mark Directory as Source Root



注：其实不需要像别人说的：

- 一定要：
 - 只导入smali代码到项目中

- 只能把smali代码所在目录去mark as source root

而是：

- 只要当前项目中包含了smali代码

即可。

TODO:

- 【已解决】安卓AS调试apk的smali：导入apktool反编译的源码作为项目代码

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2022-10-27 15:40:37

给smali加断点

给Smali代码加断点：

- 先(重点)是：找到要加断点的地方
- 再去加断点

如何找到要加断点的位置？

要在哪加断点？

才能实现，app运行时触发到断点而停下来，供自己调试

一般逻辑是：

- 找到最有可能运行到的代码的逻辑对应的地方
 - 入口的 MainActivity
 - 前面的：如何获取app的 MainActivity，已经解释过
 - 此处 YouTube 就是：com.google.android.apps.youtube.app.application.Shell_HomeActivity
 - 或者是你调试时看到的顶层页面

获取顶层页面的activity

```
adb shell dumpsys activity top | grep --color always ACTIVITY
...
ACTIVITY com.google.android.youtube/com.google.android.apps.youtube.app.watchwhile.WatchWhileActivity 781cc07 pid:20720
```

-»

- 此处 YouTube 的顶层页面是：
 - com.google.android.apps.youtube.app.watchwhile.WatchWhileActivity

然后就可以去，此处 Apktool 反编译后的，smali 的源码所在位置了：

- com.google.android.youtube_16.29.36
 - smali_classes2/com/google/android/apps/youtube/app/application/Shell_HomeActivity.smali
 - smali_classes2/com/google/android/apps/youtube/app/watchwhile/WatchWhileActivity.smali

找到了smali文件位置

给哪些函数打断点？

再说说：具体给哪些函数打断点

-» 代码运行才能，才容易触发到断点停下来（我们才好调试）

这部分涉及到安卓的正向开发知识

概述是：

- onResume
- onStart
- getLifecycle
- 等等

如此，去找：

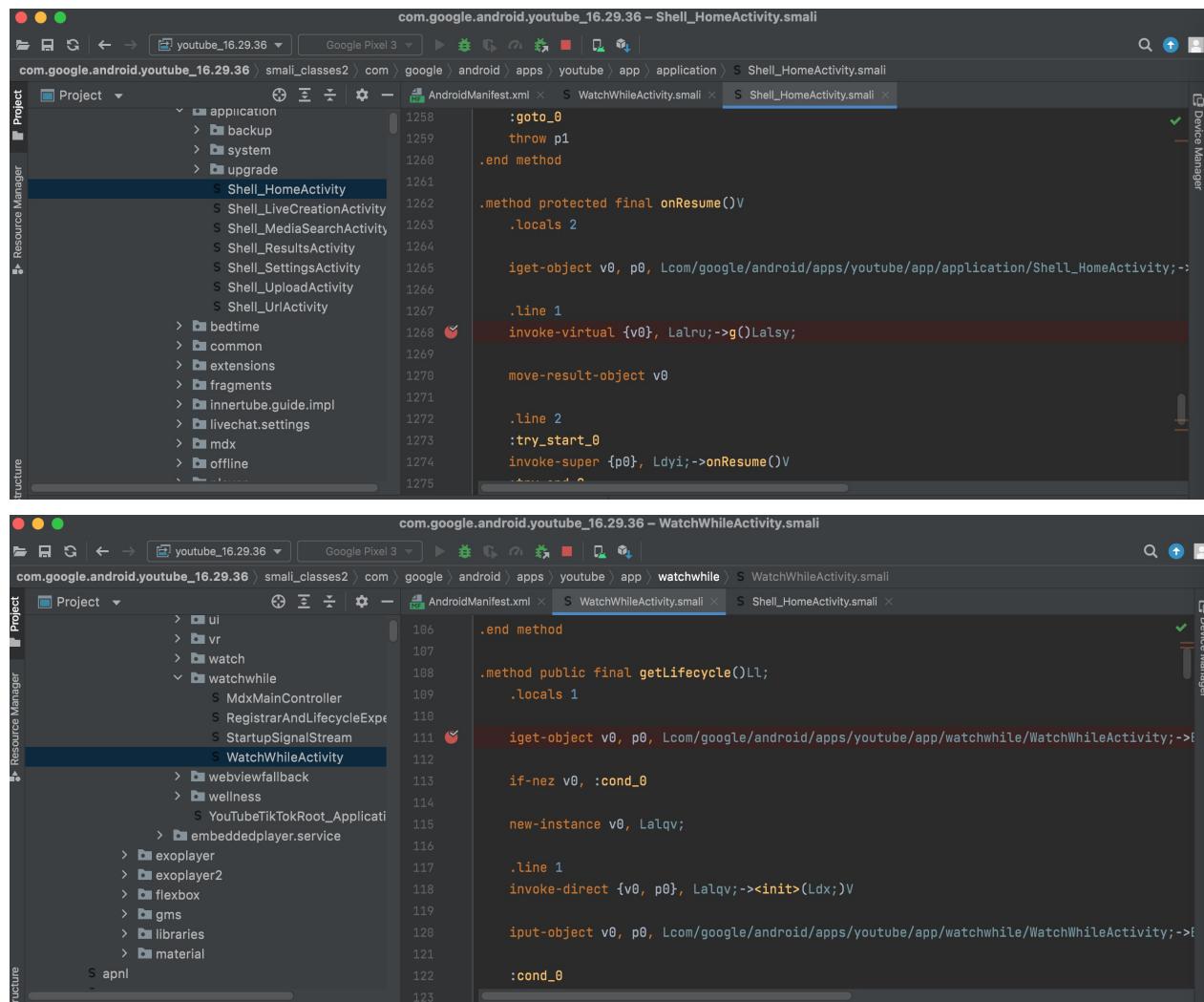
- com.google.android.youtube_16.29.36

- smali_classes2/com/google/android/apps/youtube/app/application/Shell_HomeActivity.smali
 - smali_classes2/com/google/android/apps/youtube/app/watchwhile/WatchWhileActivity.smali

的

- `onResume`
 - `onStart`
 - `getLifecycle`

等函数，去打断点：



TODO:

【已解决】Android Studio调试Smali: 给YouTube的smali代码加断点

cifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-27 15:20:21

配置AS项目

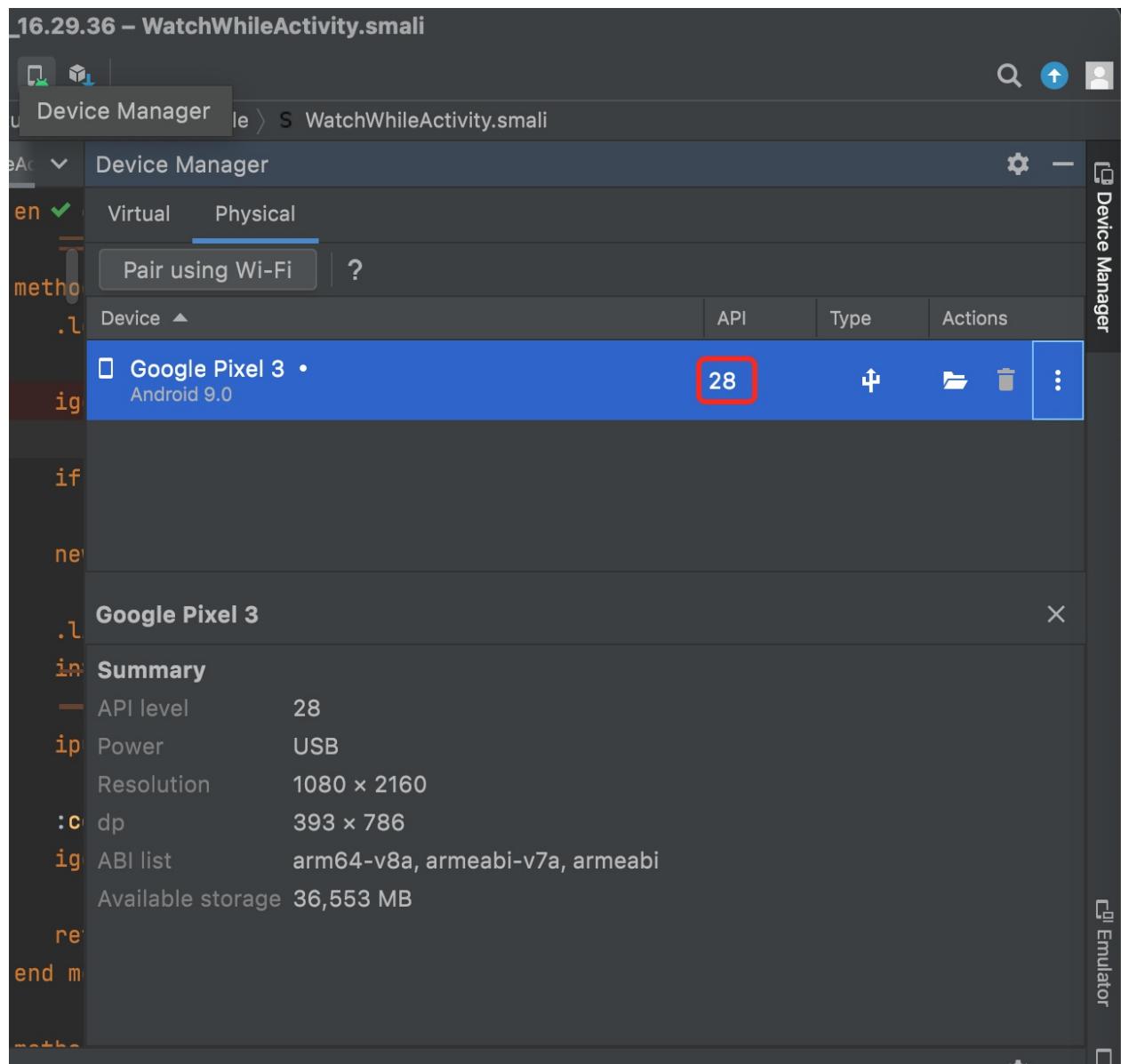
Android Studio项目初始化配置

主要是：

File -> Project Structure -> Project Settings -> Project SDK :

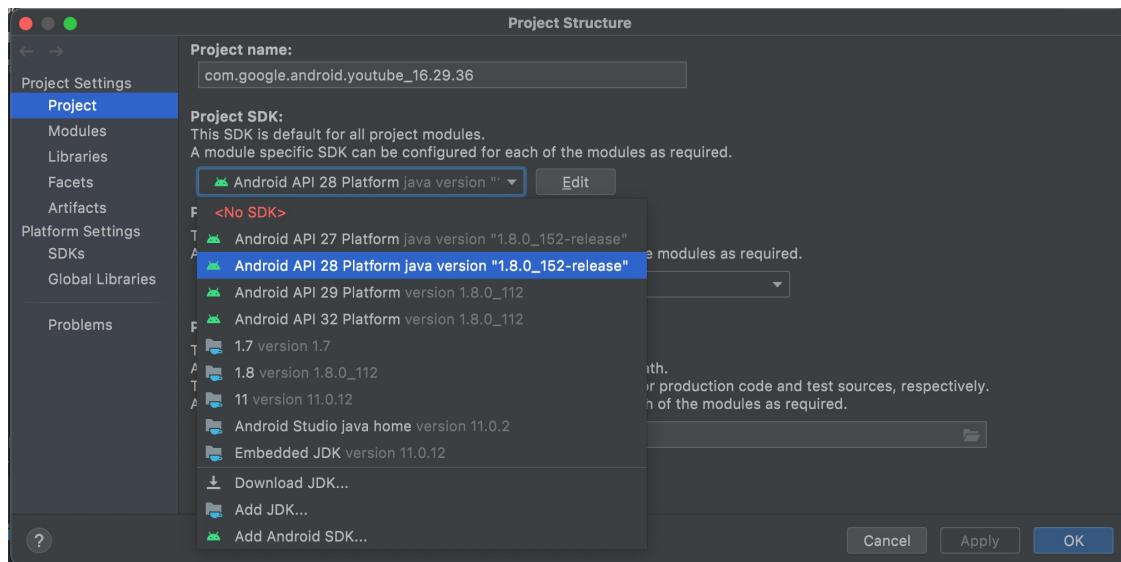
设置为和安卓设备的 Android API Level = sdkVersion 一致的值：

此处：Android设备是 Google Pixel 3，API 是 Android 9 = Android API 28



所以设置为：

- Android API 28 Platform



TODO:

- 【基本解决】安卓AS调试apk的smali: 新建和设置远程调试配置
- 【已解决】安卓AS调试apk的smali: 初始化配置AS调试环境

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-27 15:24:27

adb可获取进程列表

确保adb可以获取到安卓设备的进程列表

确保：

```
adb shell ps
```

可以获取到：

- 安卓设备中的进程的列表

目的：防止后续无法看到进程，无法调试

比如，我此处的遇到的特殊情况：

`adb shell` 的子命令，包括 `ps`，无法直接运行（要么报错，要么没权限，要么没返回结果）

则意味着后续 Android Studio 中 `Attach Debugger to Android Process` 的 `Choose Process` 中，无法看到设备的进程列表，就无法继续调试。

TODO：

- 【已解决】Android Studio和DDMS中看不到安卓设备中的所有进程
- 【已解决】Mac中运行adb shell无需su超级用户即可正常运行输出结果

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2022-10-27 15:27:19

AS调试app进程

Android Studio中调试设备端的app进程

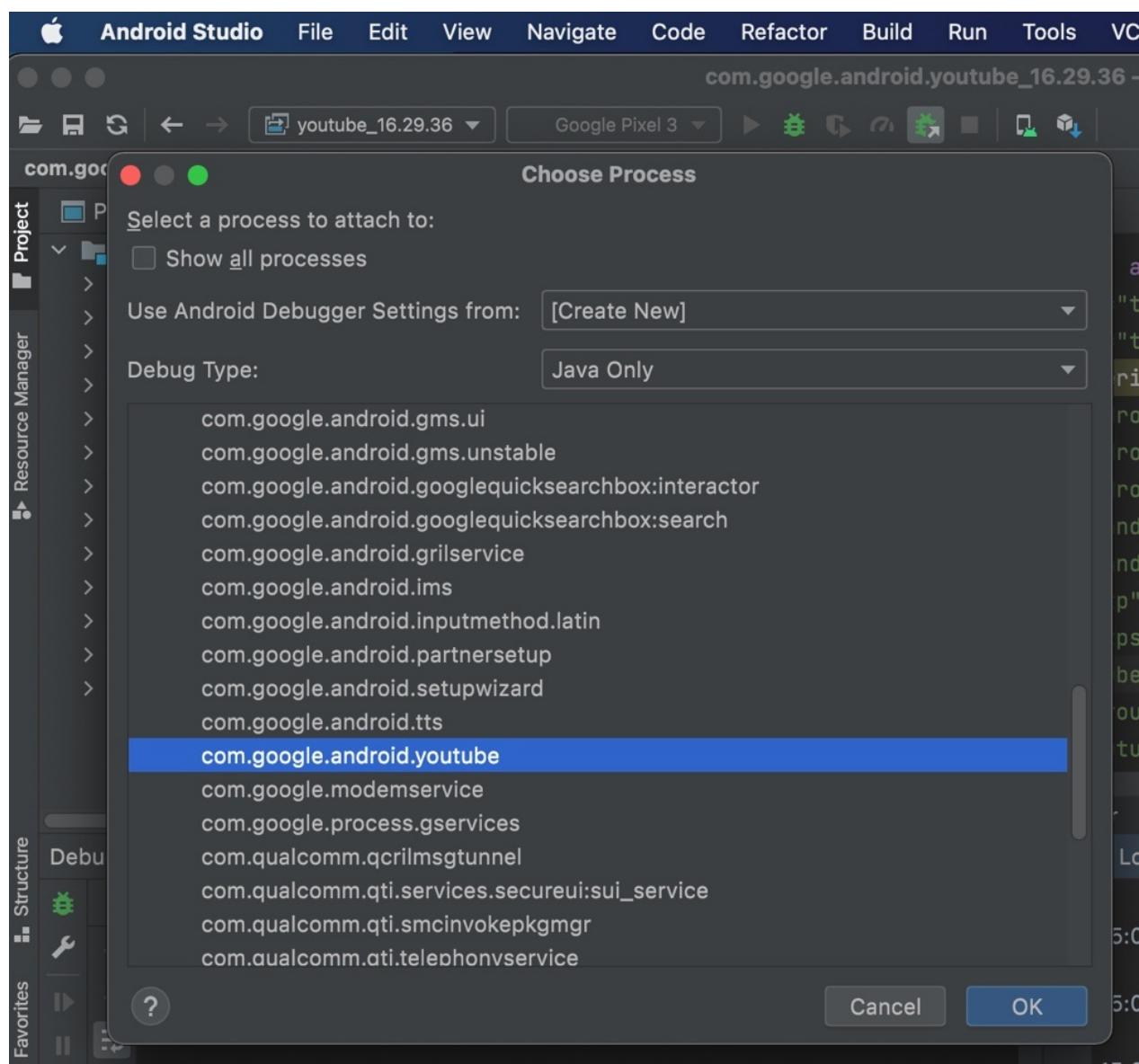
- Attach Debugger to Android Process

◦

进入 Choose Process 弹框页面

正常会显示出安卓设备，且会列出设备中可调试的众多进程

选择对应的要调试（此处是 YouTube）的进程：



即可顺利启动调试

并触发之前加的断点了：

The screenshot displays two instances of the Android Studio IDE. Both instances are connected to the same application, "com.google.android.youtube_16.29.36", running on a "Google Pixel 3" device.

Top Window (WatchWhileActivity.onResume()):

- Project View:** Shows the package structure: com.google.android.youtube_16.29.36 > smali_classes2 > com.google.android.apps.youtube.app.watchwhile > S WatchWhileActivity.smali.
- Code Editor:** Displays the smali code for the onResume() method. The cursor is at line 3375, which contains the instruction ".method protected final onResume()V".
- Variables Tab:** Shows local variables for the current frame. One variable, "this", is expanded to show its fields: A=null, B={alqv@22855}, C=false, x={nwd@22856}, y={alru@22857}, z=false, nuh.x=true, nwk.x=false, IH={alqx@22858}, njuy={Object@22859}, II={ArrayList@22860} size=4, h={ydi@22861}, I={awrh@22862}.
- Event Log:** Shows an error message: "17:18 Error running 'youtube_16.29.36': Unable to open debugger port (0.0.0:8700): java.io.IOException "handshake failed - connection prematurely closed"".

Bottom Window (WatchWhileActivity.getLifecycle()):

- Project View:** Shows the package structure: com.google.android.youtube_16.29.36 > smali_classes2 > S aqk.smali.
- Code Editor:** Displays the smali code for the getLifecycle() method. The cursor is at line 111, which contains the instruction ".method public final getLifecycle()Ll;".
- Variables Tab:** Shows local variables for the current frame. One variable, "mMainThread", is expanded to show its fields: mManagedCursors={ArrayList@22910} size=0, mManagedDialogs=null, mMenuInflater=null, mParent=null, mReferrer="com.google.android.youtube", mRestoredFromBundle=false, mresultCode=0, mResultData=null, Activity.mResumed=false, mSearchEvent=null, mSearchManager=null, mStartedActivity=false, Activity.mStopped=false.
- Event Log:** Shows an error message: "17:18 Error running 'youtube_16.29.36': Unable to open debugger port (0.0.0:8700): java.io.IOException "handshake failed - connection prematurely closed"".

TODO:

- 【基本解决】安卓AS调试apk的smali：新建和设置远程调试配置
- 【已解决】安卓AS调试apk的smali：初始化配置AS调试环境

其他调试工具

此处整理和安卓逆向的动态调试有关的，其他的分析和调试方面的工具。

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2022-10-27 16:05:46

frida

- 主页
 - Github
 - [Frida](#)
 - 官网
 - [Frida • A world-class dynamic instrumentation framework | Inject JavaScript to explore native apps on Windows, macOS, GNU/Linux, iOS, Android, and QNX](#)
- 简介
 - 一款基于 `python + javascript` 的 hook 框架
 - 可运行在Android/iOS/Linux/Win/OSX等各平台
 - 主要使用动态二进制插桩(DBI)技术
 - 将外部代码注入到现有的正在运行的二进制文件中，从而让它执行额外操作
 - 支持哪些额外操作
 - 访问进程内存
 - 在应用程序运行时覆盖函数
 - 从导入的类调用函数
 - 在堆上查找对象实例并使用
 - Hook、跟踪和拦截函数等
 - 注：调试器也能完成相应工作，不过非常麻烦，比如各种反调试
- 功能和特点
 - Scriptable
 - Inject your own scripts into black box processes. Hook any function, spy on crypto APIs or trace private application code, no source code needed. Edit, hit save, and instantly see the results. All without compilation steps or program restarts.
 - Portable
 - Works on Windows, macOS, GNU/Linux, iOS, Android, and QNX. Install the Node.js bindings from npm, grab a Python package from PyPI, or use Frida through its Swift bindings, .NET bindings, Qt/Qml bindings, or C API.
 - Free
 - Frida is and will always be free software (free as in freedom). We want to empower the next generation of developer tools, and help other free software developers achieve interoperability through reverse engineering.
 - Battle-tested
 - We are proud that NowSecure is using Frida to do fast, deep analysis of mobile apps at scale. Frida has a comprehensive test-suite and has gone through years of rigorous testing across a broad range of use-cases.
- 文档
 - [Welcome | Frida • A world-class dynamic instrumentation framework](#)
- 截图

◦

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-27 16:00:01

AndBug

- 主页
 - [swdunlop/AndBug: Android Debugging Library](#)
- 作用
 - 在没有源代码的情况下，调试android上的java程序
- 功能
 - 支持断点、call stack查看、查看class、method等信息
- 截图

◦

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-27 16:04:53

redexer

- 主页
 - [plum-umd/redexer: The Redexer binary instrumentation framework for Dalvik bytecode](#)
 - [The Redexer Dalvik bytecode instrumentation platform](#)
- 文档
 - [Tutorial: using redexer to implement logging in Android apps](#)
- 功能
 - Dalvik 字节码（用于安卓APP）分析框架
 - 它是一套基于OCaml的实用工具
 - 帮助程序员解析、操作Dalvik虚拟机
- 作者
 - Redexer由来自马里兰大学帕克分校的PLUM组织开发完成
 - 主要作者是：Jinseong Jeon, Kristopher Micinski以及Jeff Foster

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2021-07-18 09:55:45

Fino

- `Fino = FINO`
- 主页
 - [sysdream/fino: Android small footprint inspection tool](#)
- 功能
 - An Android Dynamic Analysis Tool
- 特点
 - 资源占用少

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2021-07-18 09:55:45

心得

安卓逆向期间，折腾Android Studio去调试smali代码，其实更麻烦的不在于搞懂本身的流程，而在于：

找到一个好用的调试设备：已root好的，可以顺利调试的安卓手机

因为期间遇到Google Pixel 3的adb异常，导致adb shell各种命令无法正常运行，包括adb的ps无法正常获取进程列表

从而导致后续调试时AS中看不到进程，折腾了很长时间，才确认，就是adb方面的问题导致的。

解决了该问题了，AS中能attach到进程，后续就顺利多了。

并且另外Google Pixel 3本身不稳定，导致虽然开始能调试，但是后来出现开发者选项崩溃的问题，无法进入设置了。对于正常安卓逆向开发，也有很大影响。

总之：还是要找个靠谱的root后的安卓手机，才能保证后续安卓逆向的顺利。

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2022-10-27 15:57:46

附录

下面列出相关参考资料。

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-27 14:20:47

参考资料

- 内容相对最完整
 - 安卓逆向13 --- *AndroidStudio + Smalidea 动态调试 smali 代码【APK可调试】、gradle 配置擒贼先擒王的博客-CSDN博客_smalidea*
- DDMS adb 调试 端口 映射 关系
 - Android调试系列—使用android studio调试smali代码 - Gordon0918 - 博客园 (cnblogs.com)
- Android Studio 动态调试 apk 反编译出的 smali 代码 - yhjoker - 博客园 (cnblogs.com)
- Android逆向破解：使用Android Studio调试反编译后的smali代码 - 简书 (jianshu.com)
- Android逆向 | AndroidStudio的两种动态调试技巧 - 腾讯云开发者社区-腾讯云 (tencent.com)
- 超详细的android so库的逆向调试 - 掘金 (juejin.cn)
- 学习笔记：Android studio 调试smali_深秋黄金甲的博客-CSDN博客_androidstudio smali
- Android Studio 3.6 调试 smali的全过程 - 腾讯云开发者社区-腾讯云 (tencent.com)
- Smali动态调试之Android Studio - Blog - teisyogun (bushrose.github.io)
- 基于Android studio动态调试smali全过程 - 简书 (jianshu.com)
- Android逆向笔记-使用Android Studio调试Smali代码（方式一）_IT1995的博客-CSDN博客
- android 动态调试 - 简书 (jianshu.com)
- Android Applications Reversing 101 (evilsocket.net)
- 安卓逆向-从环境搭建到动态调试apk - FreeBuf网络安全行业门户
- 安卓逆向分析中常用动态调试方法总结Denny_Chen的博客-CSDN博客_安卓动态调试
- Android逆向破解：使用Android Studio调试反编译后的smali代码 - 简书 (jianshu.com)
- Android Studio 动态调试 apk 反编译出的 smali 代码 - yhjoker - 博客园 (cnblogs.com)
- [免费专栏] Android安全之Android Studion 动态调试APK的两种方法菠萝橙留香的博客-CSDN博客_安卓apk调试
- [原创]修改Nexus5的boot.img - 打开系统调试-Android安全-看雪论坛-安全社区|安全招聘|bbs.pediy.com

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-10-27 15:36:57