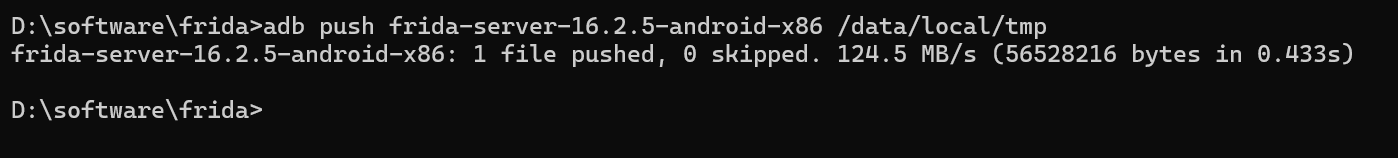
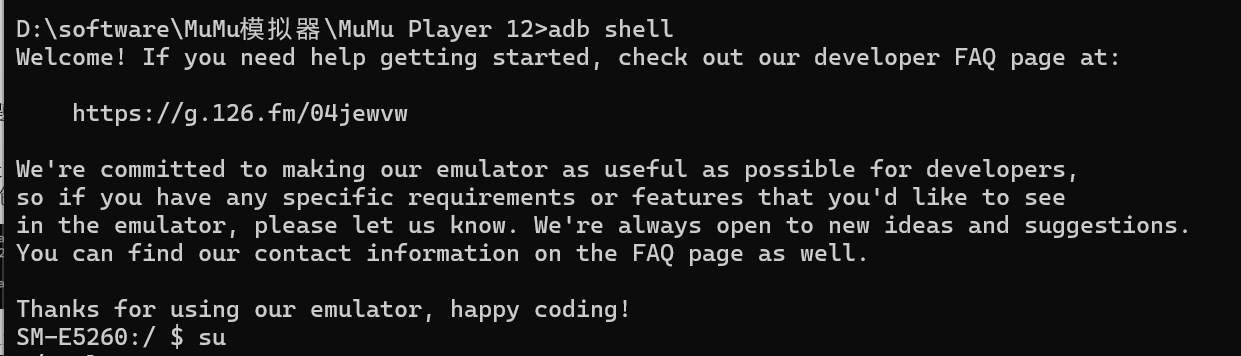
# 配置环境

因为使用的是MuMu模拟器，所以在这里首先尝试下载frida-server-16.2.5-android-x86包，在操作机上将解压后的包发送到android模拟器中



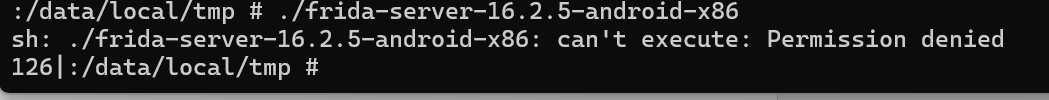
在终端2中通过adb获取到android机的shell



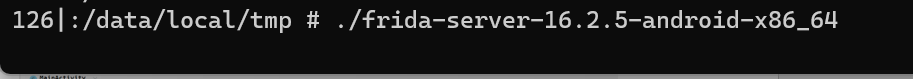
进入到data/local/tmp目录下



运行上一步中发送过来的frida-server-16.2.5-android-x86包

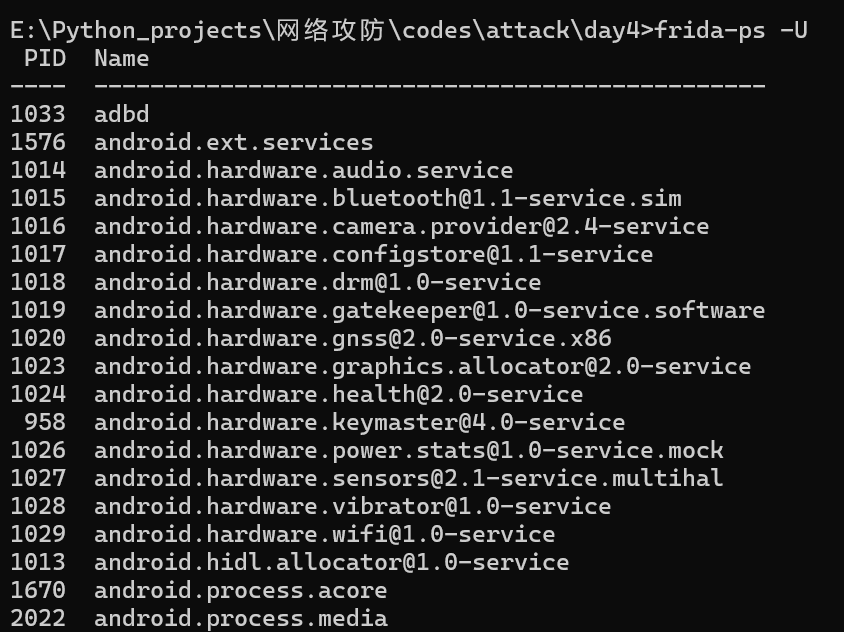


发现不能够识别，由此断定应该下载frida-server-16.2.5-android-x86\_64包，重复上面的步骤之后运行frida-server-16.2.5-android-x86\_64包



如果不会输出任何回显则说明运行正常

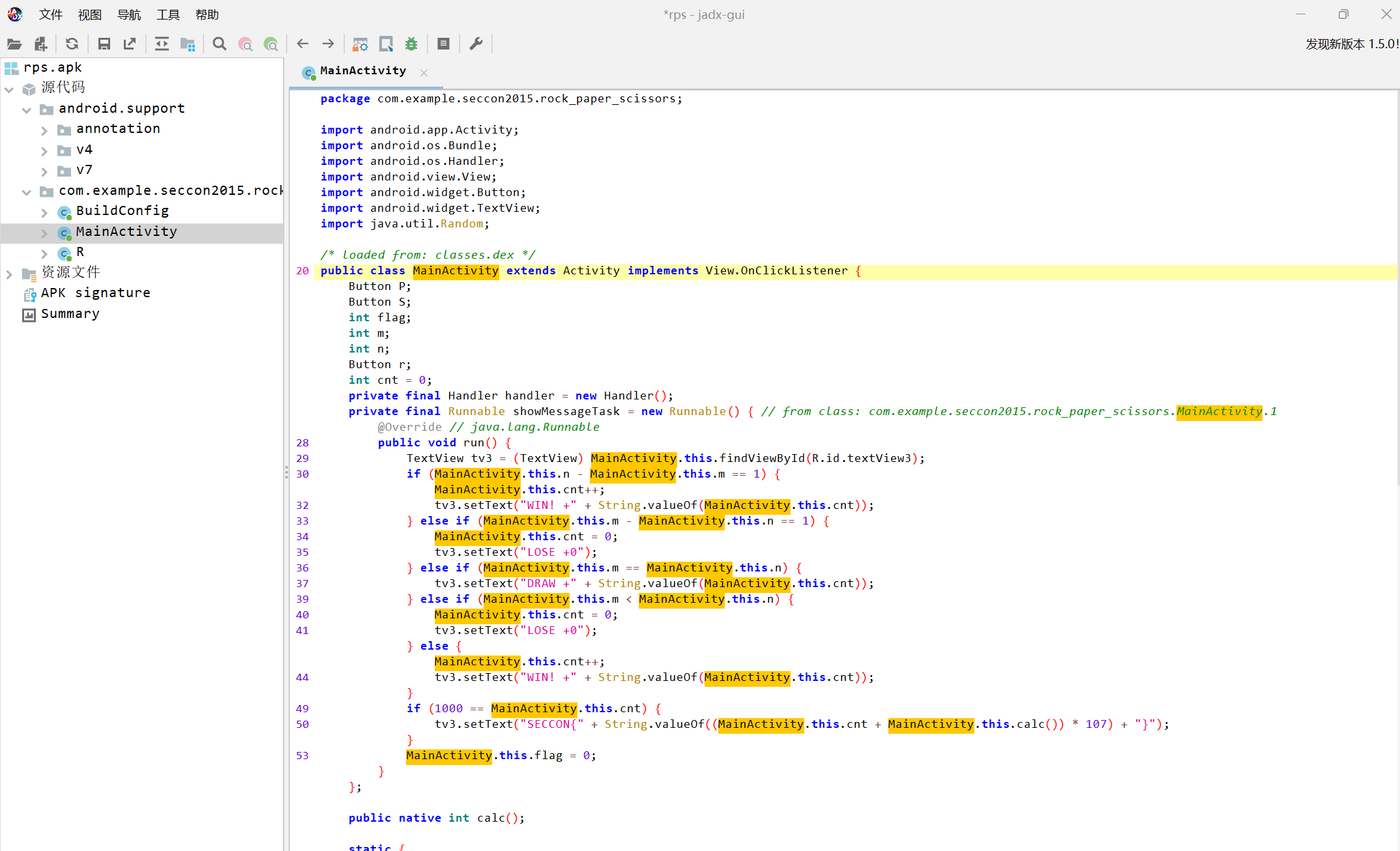
接下来进入到终端1中执行frida-ps -U,可以显示出android机正在运行的所有的pid以及应用包。



在这里frida的环境已经全部配置完毕

# 分析代码

将apk包使用jadx工具打开



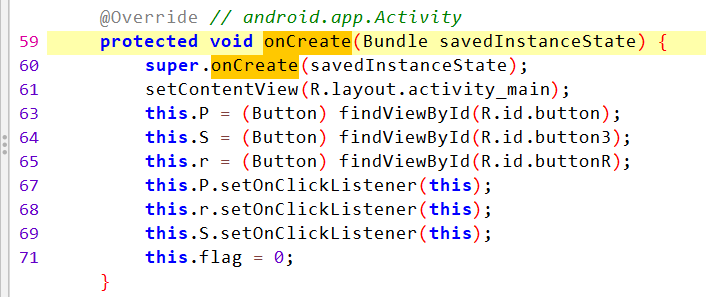
可以找到onCreate方法如下

该方法首先设置布局内容，通过setContentView(R.layout.activity\_main)设置了Activity所使用的布局文件为activity\_main.xml。

然后初始化UI元素，通过findViewById()方法，onCreate()会初始化Activity中的UI元素，初始化了Button和TextView等元素。

之后设置事件监听器，为UI元素设置事件监听器，以便在用户与界面交互时响应相应的动作。通过setOnClickListener()方法为按钮设置了点击事件的监听器。

最后初始化变量和状态，在onCreate()中，还可以进行一些变量的初始化或状态的设置。初始化了一些整型变量flag、m、n和cnt，并将flag初始化为0。



找到MainActivity方法发现可以找到onClice方法如下

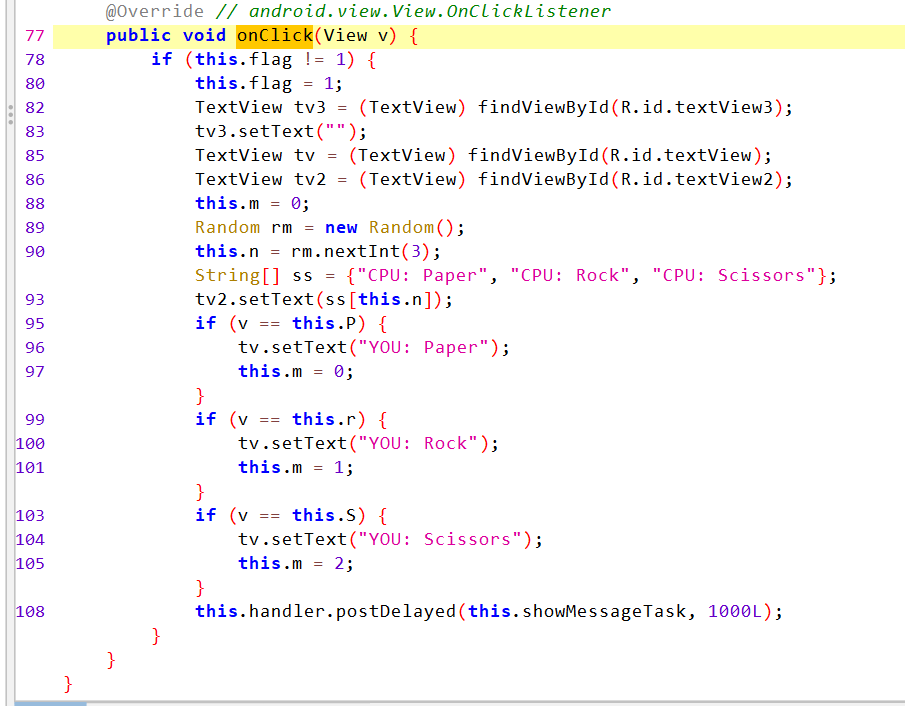
onClick()方法是一个事件处理方法，用于处理用户与界面交互时的点击事件。当用户点击与onClick()方法相关联的UI元素时（比如按钮），系统会自动调用onClick()方法来响应用户的操作。onClick()方法被实现在MainActivity类中，并且该类实现了View.OnClickListener接口，这意味着MainActivity类负责处理点击事件。

onClick()方法的作用如下：

首先处理点击事件： 当用户点击与onClick()方法相关联的UI元素时，系统会自动调用onClick()方法，通过判断点击的是哪个UI元素（通过View v参数传入），可以执行相应的逻辑操作。

之后执行业务逻辑： 在onClick()方法中，可以编写业务逻辑来响应用户的点击操作。根据用户点击的不同按钮，设置了相应的文本内容，并触发了一段延迟执行的任务（通过handler.postDelayed()方法），最终更新了文本视图的显示内容。

最后处理多个UI元素的点击事件： 由于onClick()方法是通过实现View.OnClickListener接口来实现的，因此可以用于处理多个UI元素的点击事件。通过在onCreate()方法中为各个UI元素设置相同的点击监听器（setOnClickListener(this)），当用户点击任意一个按钮时，都会调用onClick()方法，从而执行相应的逻辑。

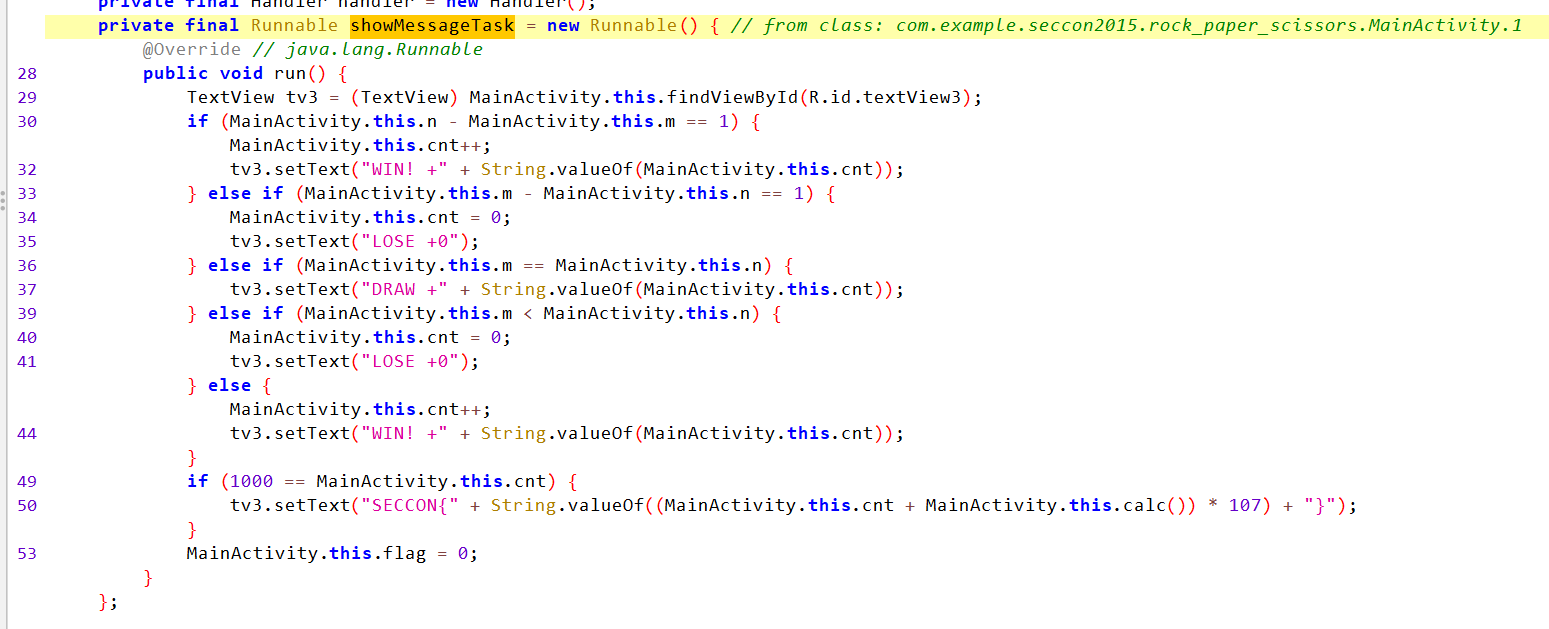


继续查看文件的showMessageTask方法

在run()方法中定义了游戏结果的显示逻辑。根据玩家和计算机的选择，更新了文本视图tv3的显示内容，显示出游戏结果，比如"WIN"、"LOSE"或者"DRAW"，同时根据不同结果更新了计分器cnt的值。

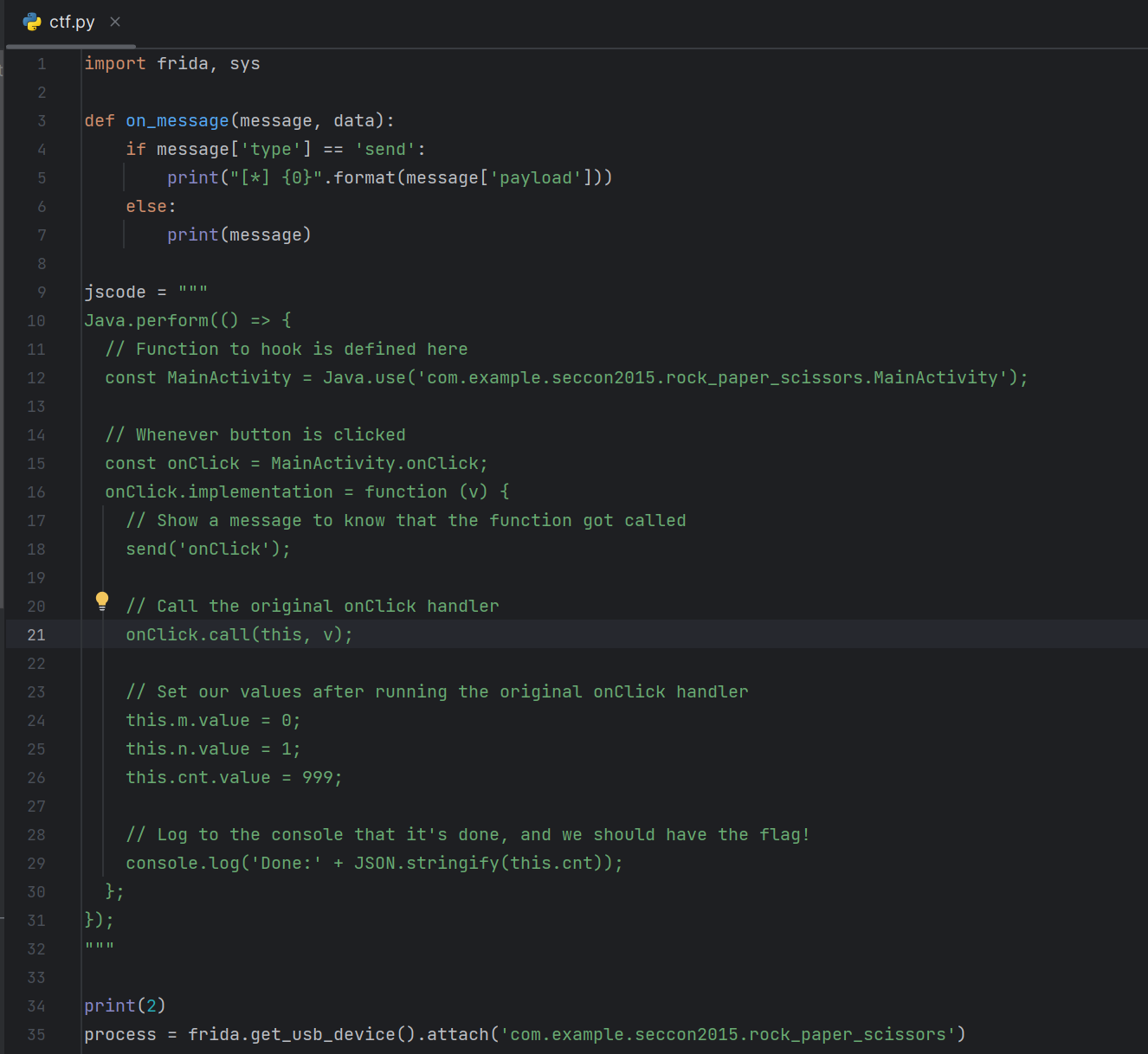
当计分器cnt达到一定条件时（这里是达到1000），显示了最终的游戏结果，即一串带有特定格式的字符串，作为游戏的最终胜利条件。

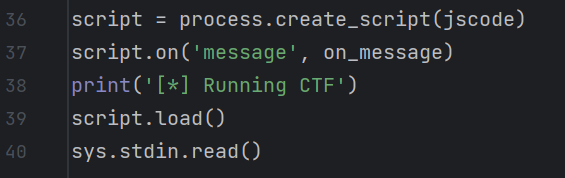
在处理完游戏结果后，将flag置为0，表示游戏结果已经处理完毕，可以进行下一轮游戏。



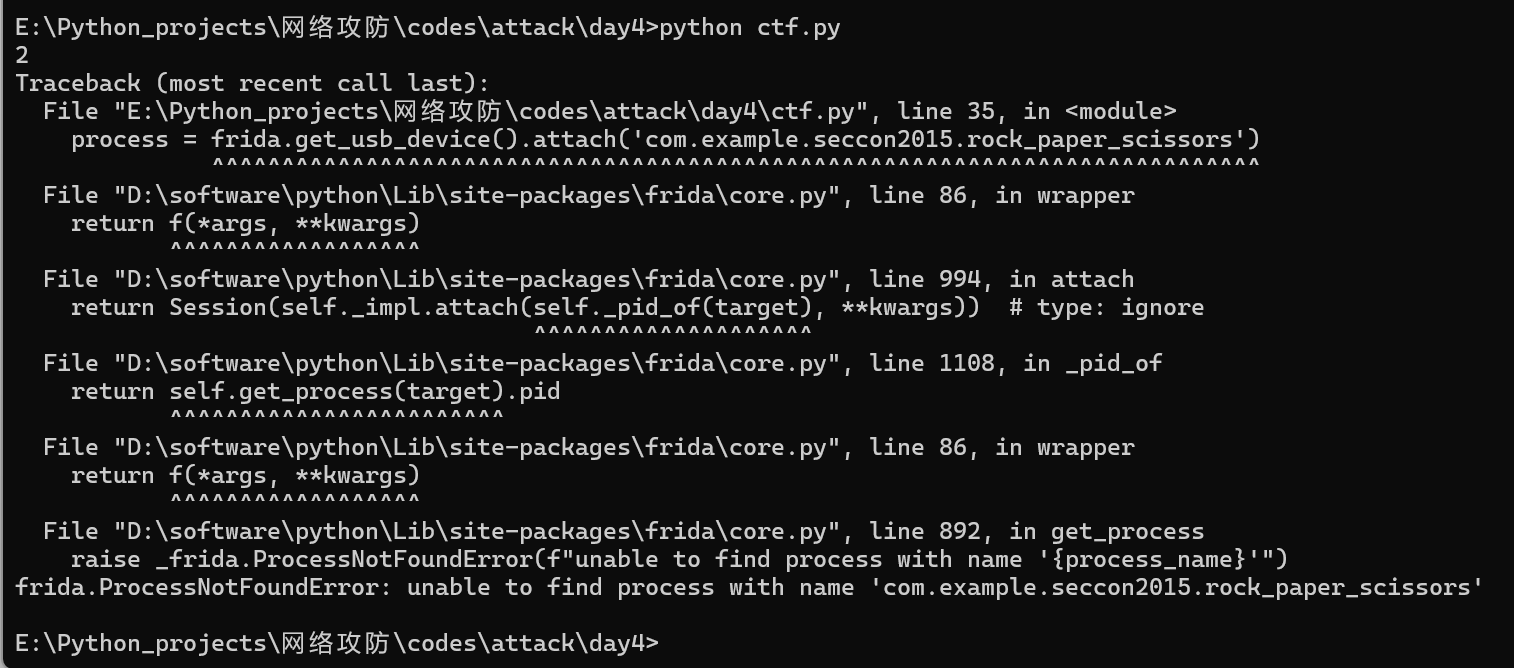
# 动态hook

在官方网站上下载ctf.py文件如下

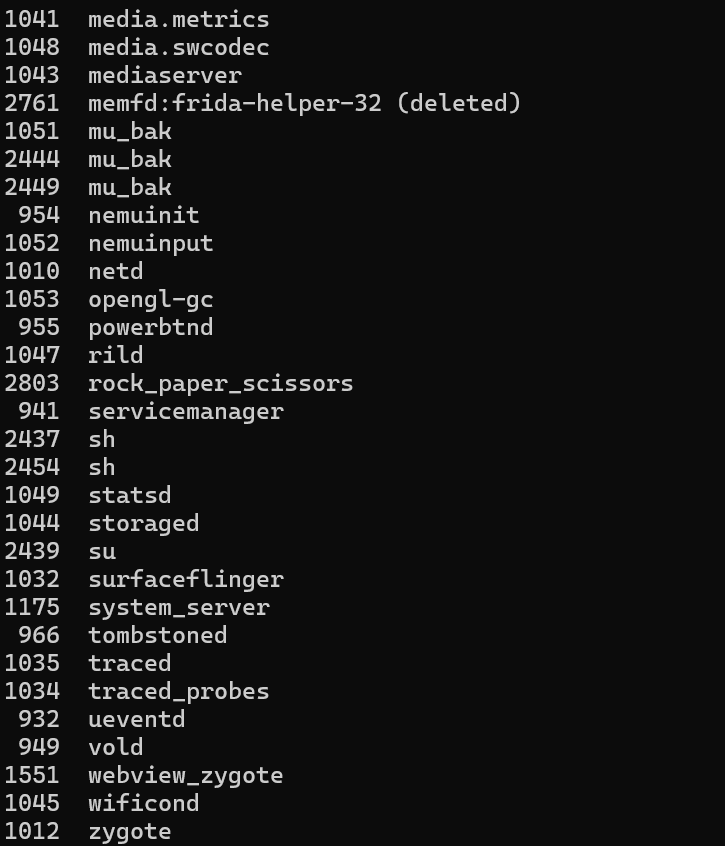




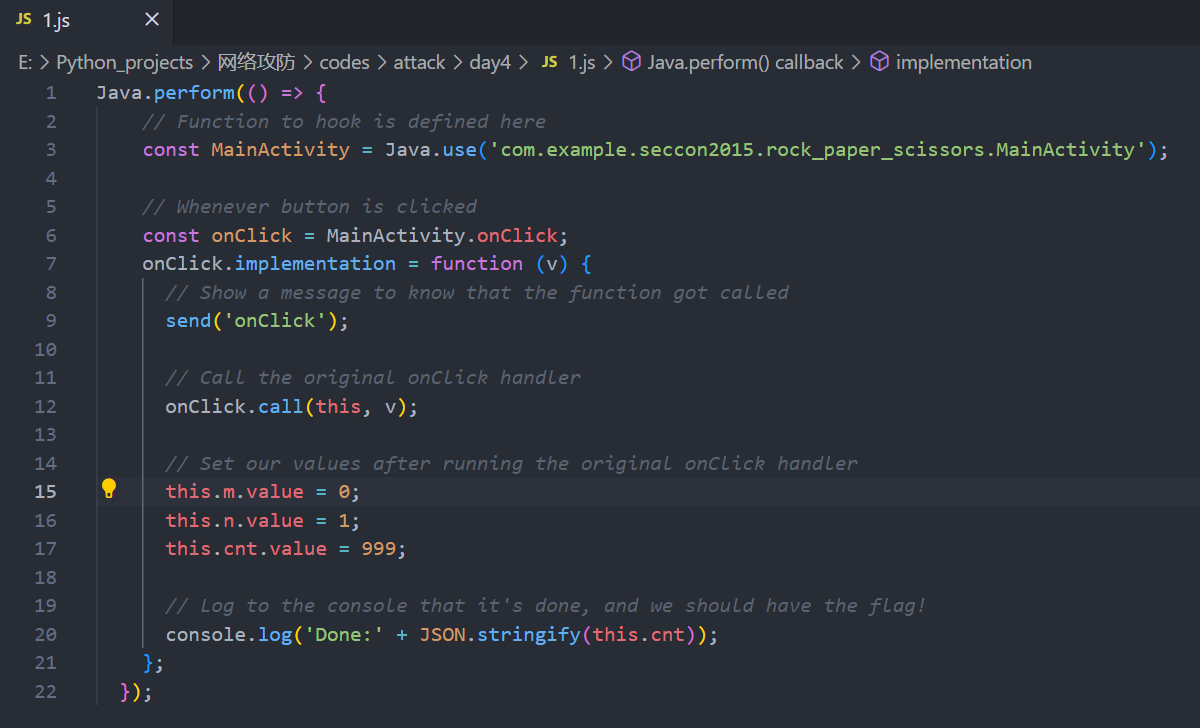
在模拟器中打开该程序后在终端1中直接运行该程序，发现并不能与应用包相连



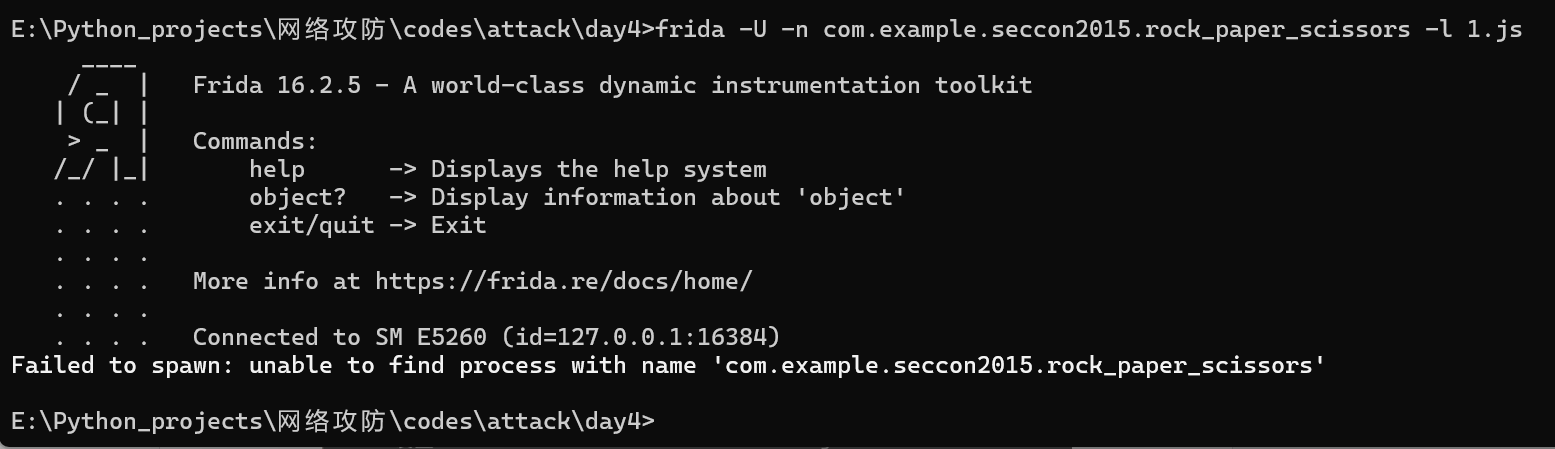
查看当前正在运行的进程发现目标进程rock\_paper\_scissors有回显



在经过多种方式的尝试之后在这里放弃使用python脚本，绕过python直接将官方给出的python中的JavaScript代码拿出保存到1.js文件中如下



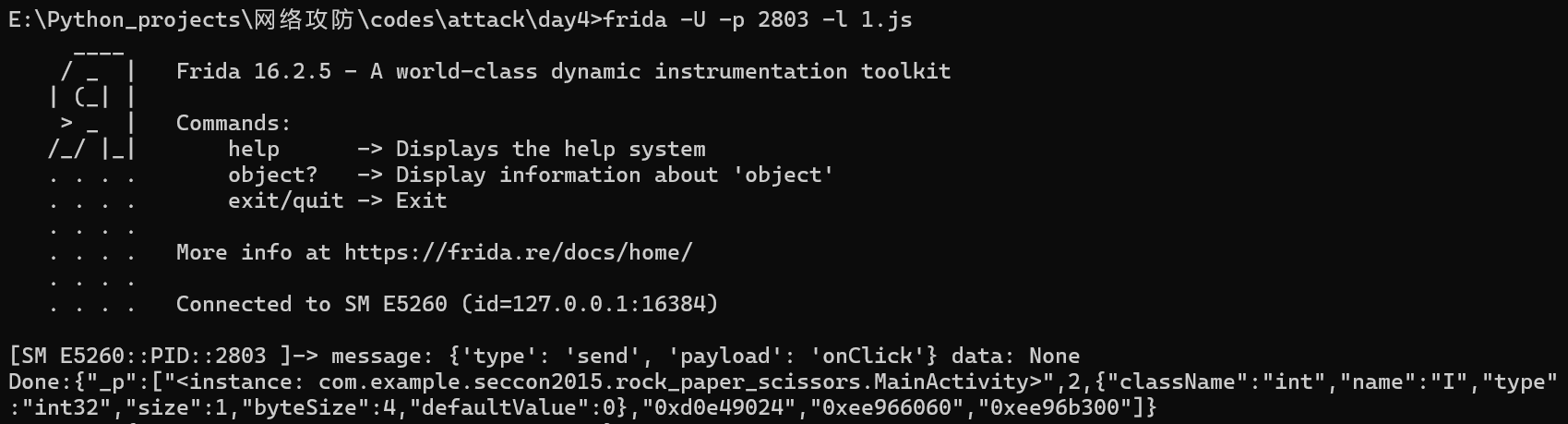
直接使用js代码通过包名对目标程序进行hook如下，发现与使用python的结果一样，同样是不能够识别包名



在网上查找资料发现可以使用进程id代替包名来进行hook的方法。首先查看目标进程的pid



直接通过pid对目标程序使用JavaScript脚本进行hook，点击任意按钮之后发现终端中有正常的输出



进入到模拟器中查看目标应用发现成功将flag输出

