**Lab 2. Android App Reverse II**

* **Task A**

(1) Your Answer

[If you choose to write an app to send the Intent, please submit the APK to the Git server and copy the Intent-sending code here.]

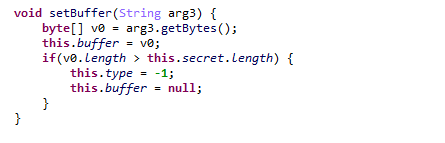
.\adb shell am start -n com.example.lab2/.MainActivity --ei type 1 --es name Mingxiao --es password 12345

(2) Writeup

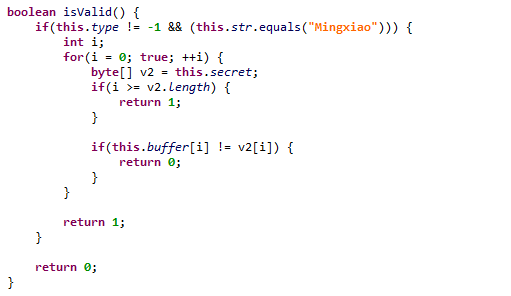
[Record how you solve this task here.]



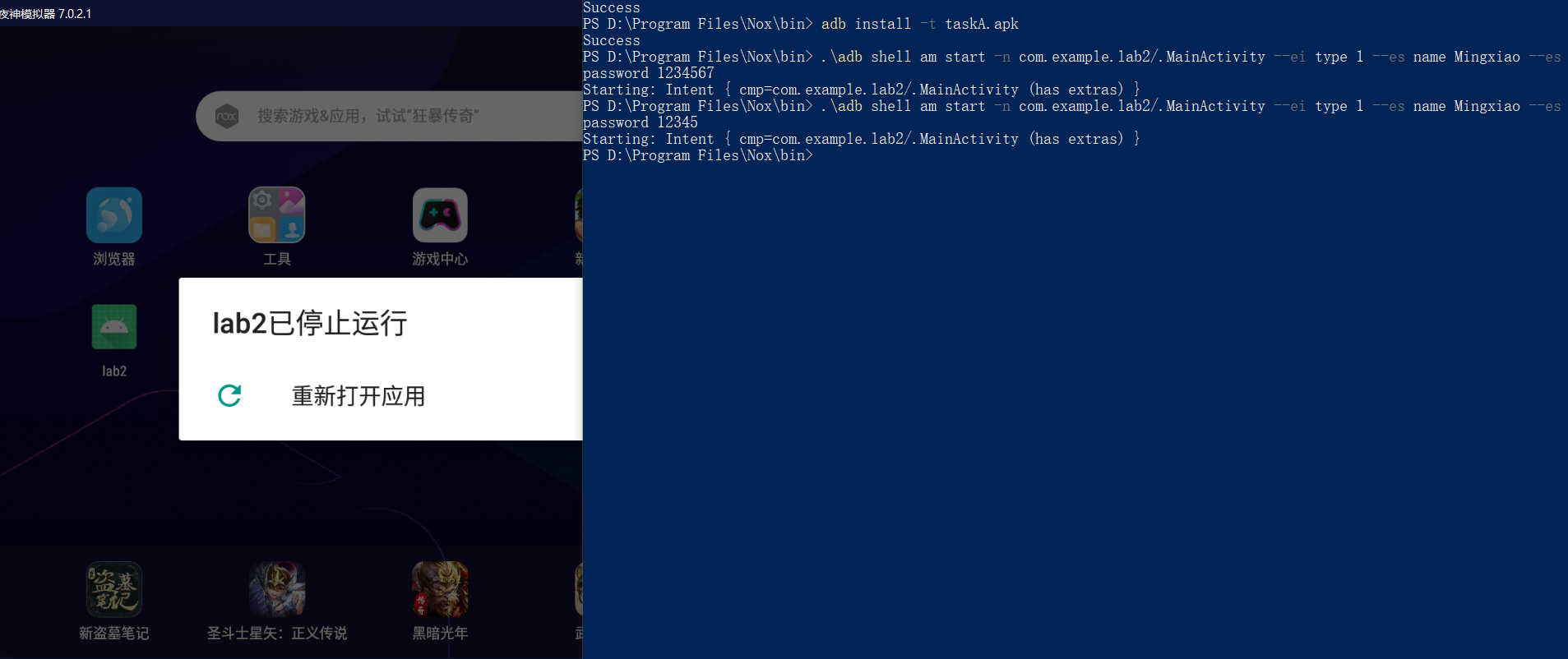
上面这段代码 前面的一大段有try catch语句，说明这一段内容不会报错，所以需要通过，而下面有一个isValid函数，这不在范围内，是crush的目标



看一下setbuffer函数 只要输入的密码位数小于等于secret的大小就能通过 所以输入<=7



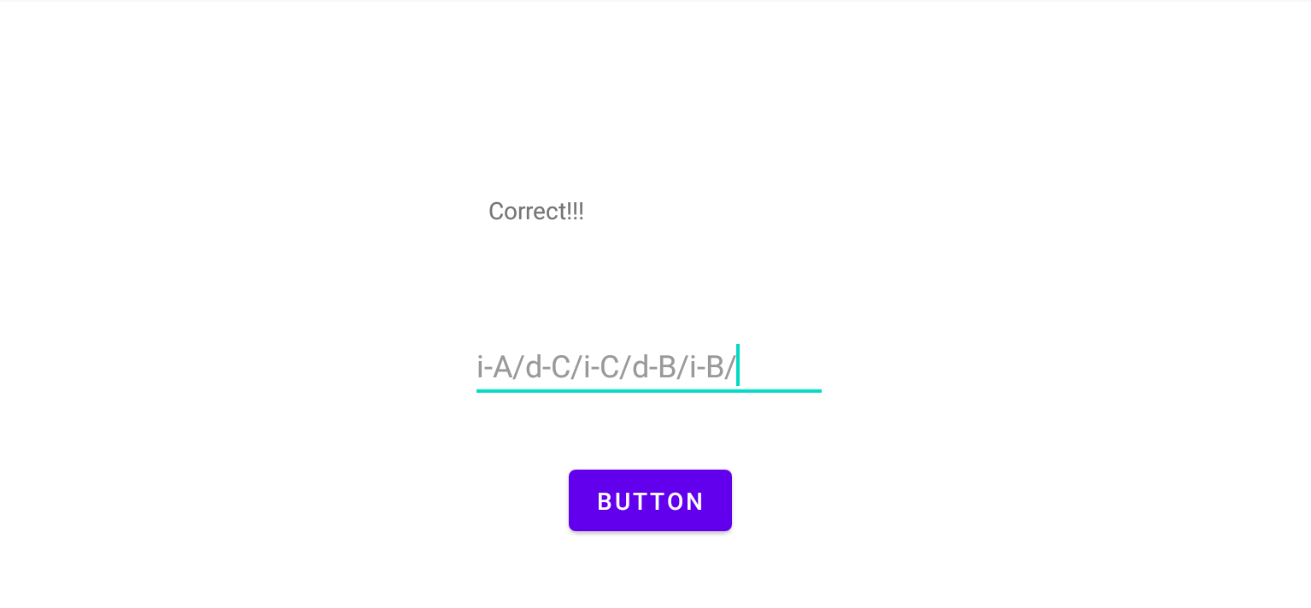
看需要crush的isValid 发现它通过判断i >= this.secret.length来退出循环，之后会访问两个byte串i的内容，这就有突破点。因为i是固定的7，所以只要输入的buffer<7 在访问内容的时候就会越界出错。



* **Task B**

(1) Your Answer

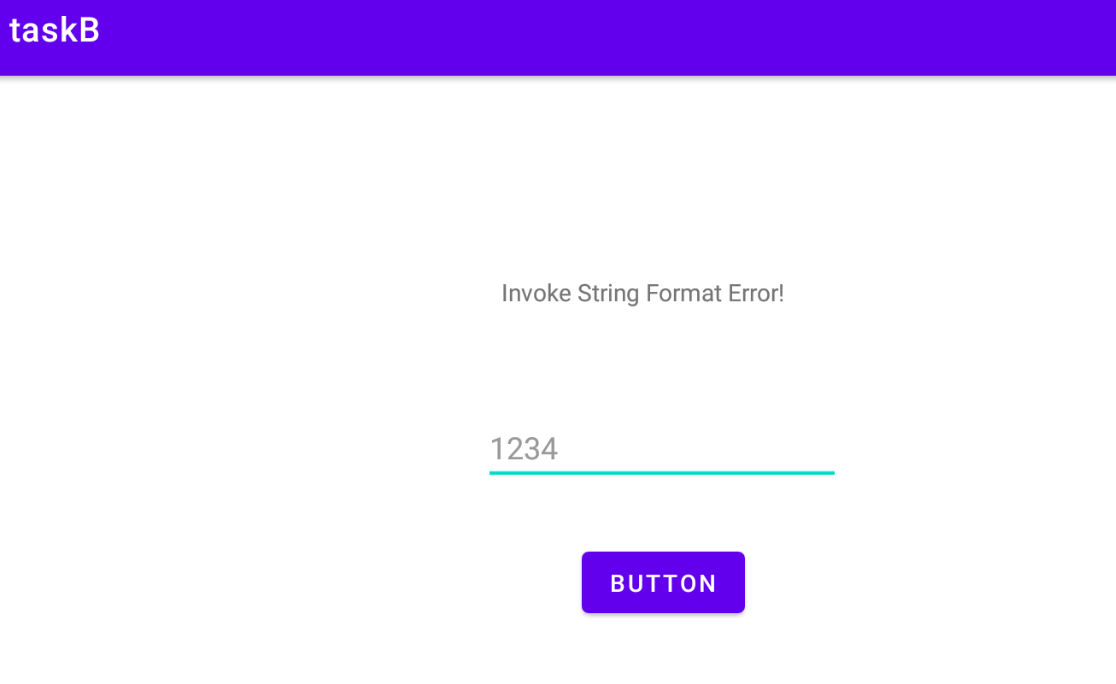
i-A/d-C/i-C/d-B/i-B/



(2) Writeup

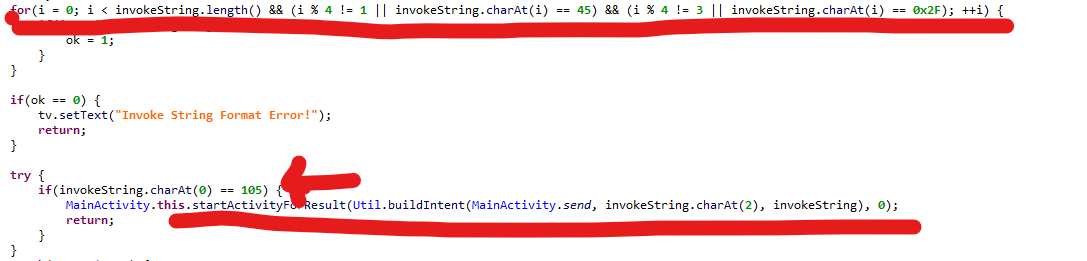
[Record how you solve this task here.]

先随便输入看看

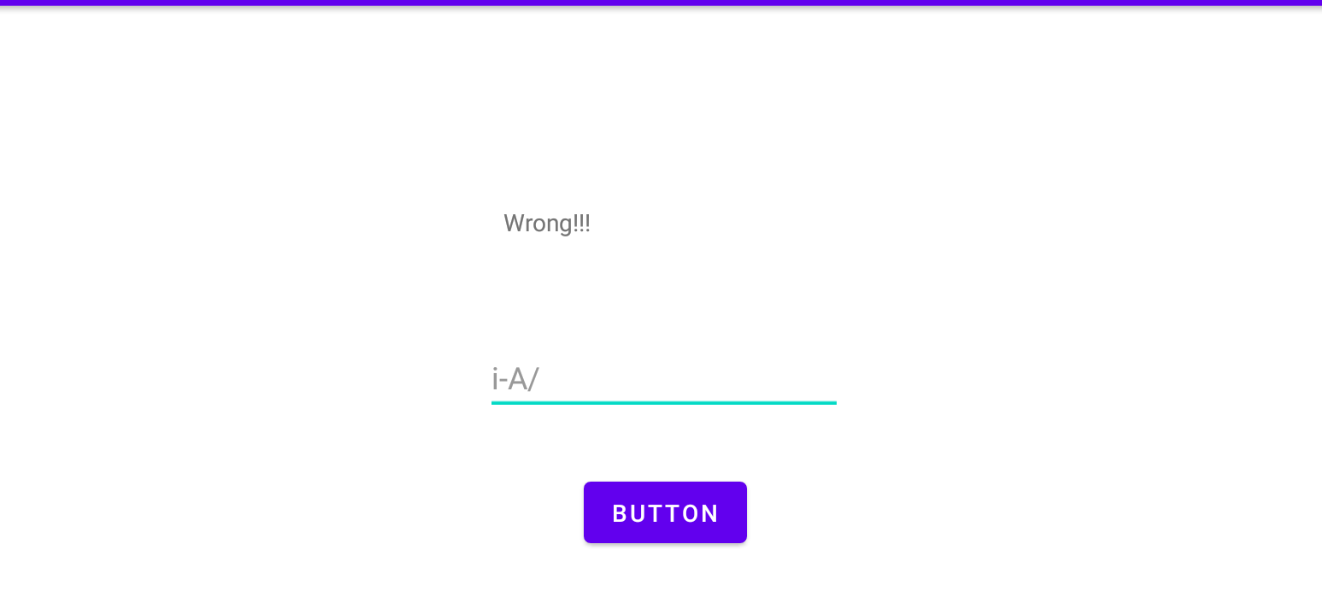


ok，那我就先把apk用jeb逆向分析一下代码

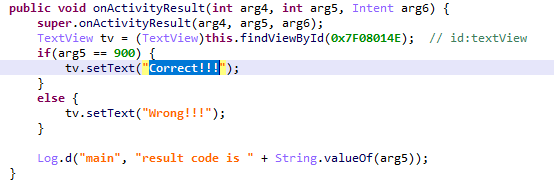
首先先看进入的MainActivity的输入要求



第一个判断，说明了字符串四个为一组，而且字符串的格式是□-□/, 而且字符串第零个是i，并且为i后，start一个以第二个字符为名字的activity，并传入MainActivity的send值100。所以我先输入一个以此为格式的字符串。



那么如何correct呢？这里的函数给出了



字符串的判断回调函数onActivityResult()实现，如果启用的其他进程返回了900，那么输入的字符串就是正确的。

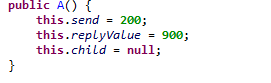
如何收到调用进程的返回值呢，通过有final()函数的地方就能收到了

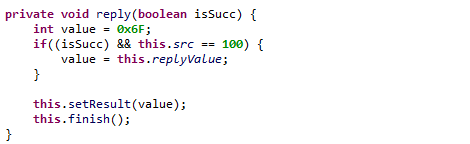
ok，MainActivity完毕，进入Util模块



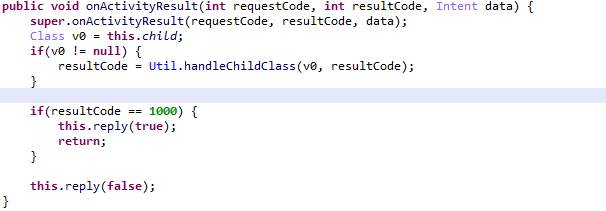
其中有三个函数，第一个是每四个字符截取，第二个是返回一个以输入字符为名和输入数字为src的进程，第三个函数是将输入的子进程进行get名字为“handleResponse“这个函数调用，使用resultcode的值，返回进程的返回值

所以综上所述，现在第一步需要调用一个能返回900的函数，去A、B、C三个进程找，发现A有。 **第一部分 i-A/**

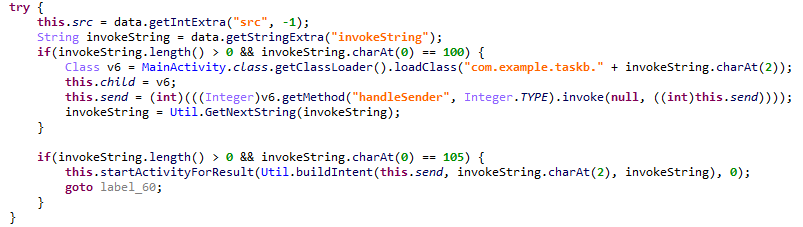




但是返回有两个先决条件，一是reply的参数是True，二是src==100。当然恰好，主进程中输入的就是100，所以第二个是满足的，那么接下来看



resultcode是通过其他进程的final()后返回的，是1000，所以接下来，需要启动一个新进程并且其返回值是1000



chr(100) == d; chr(105) == i;

d中的函数，将一个子进程（子类）赋给了child，将进程调用handleSender（this.send）的返回值赋给该进程的send。

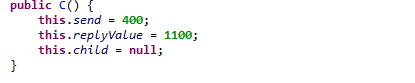
i中的，就是和之前的main一样，创建新进程，得到返回值。

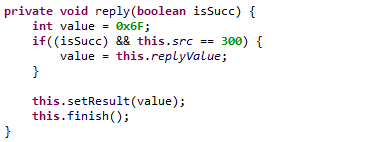
ok，那让我们找找哪里返回1000 发现C有，不过是在



这里，所以需要d进入Util.handleSender **第二部分 d-C/**

进入C进程后，需要返回1000的条件，是respond == 1100，需要一个进程的返回值来返回1100，是谁呢？是C自己呀！



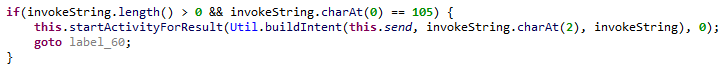


所以它需要启动一个自己子进程的finish() **第三部分i-C/**

就和上面一样，C的reply()也要满足两个条件 其中第二个条件依然是直接满足的，但我之前没解释。首先，从A调用d-C时，会将他的200传送到C的HandleSender函数，C的HandleSender函数

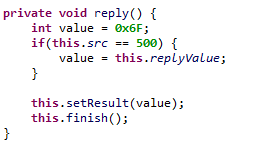


恰巧给d-C进程赋值了了个300，而d-C调用i-C时



正好又把自己的300给了i-C的src，所以很巧

接下来就和上面的步骤一样，找到了**第四第五d-B/i-B/，**一样的所以就不在解释了



但由于B的reply()不需要参数，只要满足src，所以就结束了

所以payload就是i-A/d-C/i-C/d-B/i-B/