

XCTF mobile新手区解题记录(WP)以及一些总结和思考

原创

windy_ll 于 2020-03-28 17:46:46 发布 654 收藏 3

版权声明：本文为博主原创文章，遵循 [CC 4.0 BY-SA](#) 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/qq_41374107/article/details/105165212

版权

XCTF mobile新手区解题记录以及一些总结和思考

前言

题目：app3

题目：easy-apk

题目：easy-java

题目：easy-jni

题目：easy-so

题目：app1

题目：Ph0en1x-100

题目：RememberOther

题目：app2

题目：黑客精神

题目：easy-dex

题目：我是谁

一些总结

一些博客链接

前言

疫情当下，都已经耍了好几个月了，都不知道干啥了，好不容易等到我四川宣布开学时间了，结果四川高校一点动静都没有，无聊中，那就写一下解题记录吧，有些题先前已经在我个人博客上发过了，就不再重复写了，贴个链接就行了!!!

题目：app3

此题wp已经在个人博客写过，不在详细说明，详情请看链接：<https://www.52pojie.cn/thread-1082706-1-1.html>

题目：easy-apk

此题wp已经在个人博客写过，不在详细说明，详情请看链接：<https://www.cnblogs.com/aWxvdmVseXc0/p/11955006.html>

题目：easy-java

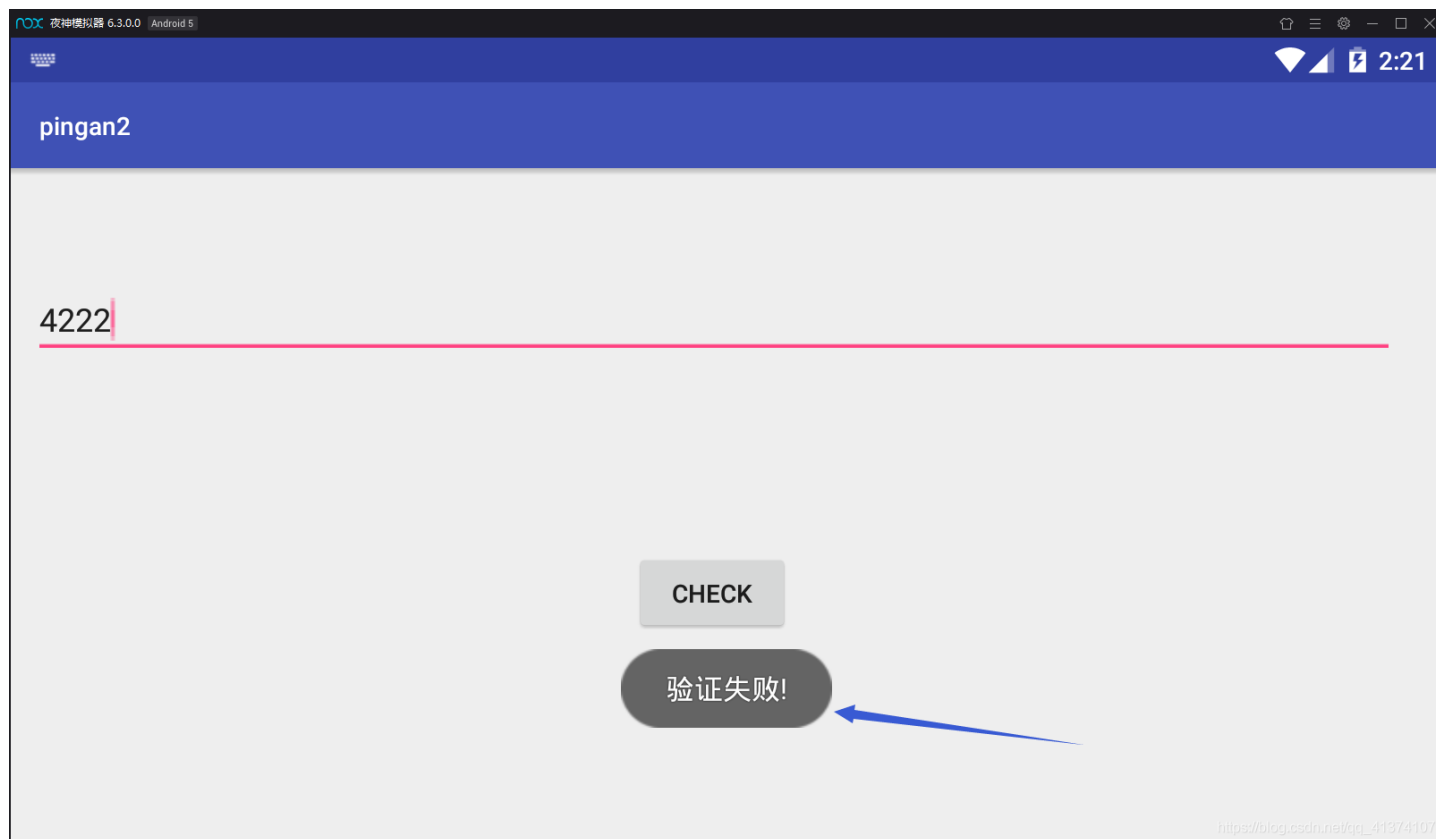
此题wp已经在个人博客写过，不在详细说明，详情请看链接：<https://www.cnblogs.com/aWxvdmVseXc0/p/12207697.html>

题目：easy-jni

此题wp已经在个人博客写过，不在详细说明，详情请看链接：<https://www.cnblogs.com/aWxvdmVseXc0/p/12198459.html>

题目：easy-so

1、下载好题目，拖入夜神中，打开如下所示：



2、查壳，发现无壳，用jeb反编译后，找到关键字 **验证失败** 所在类，发现调用了so层的 **CheckString** 函数进行了验证，传进去的参数为我们在输入框中输入的字符串，如下图所示：



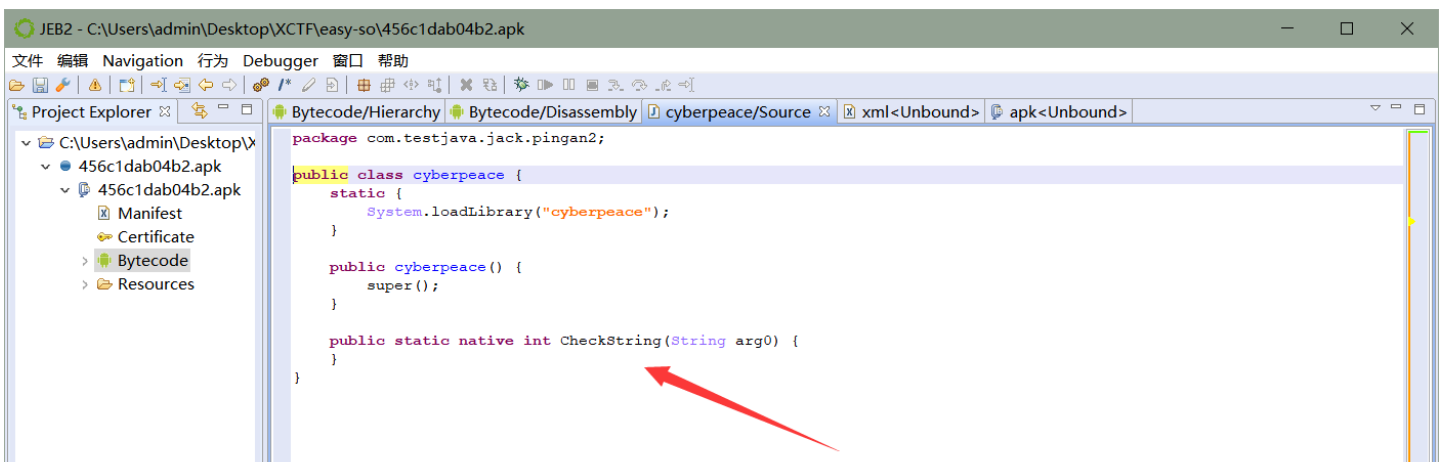
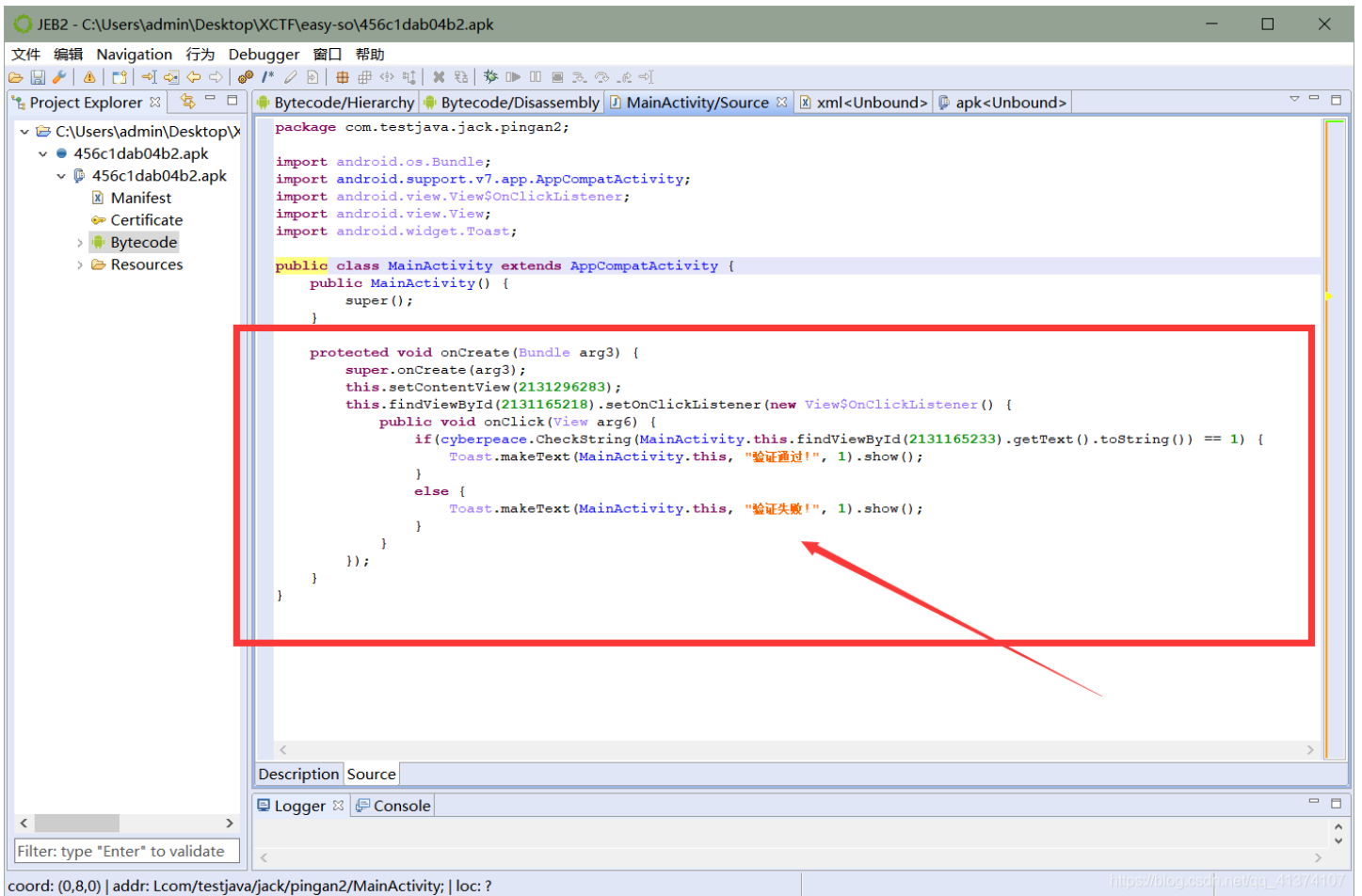
百度加固、阿里加固、腾讯加固、盛大加固、瑞星加固、网秦加固、
国信灵通加固、apkprotect加固、几维安全加固、顶像科技加固、
网易易盾 等常规厂商
更多请访问:www.legendsec.org

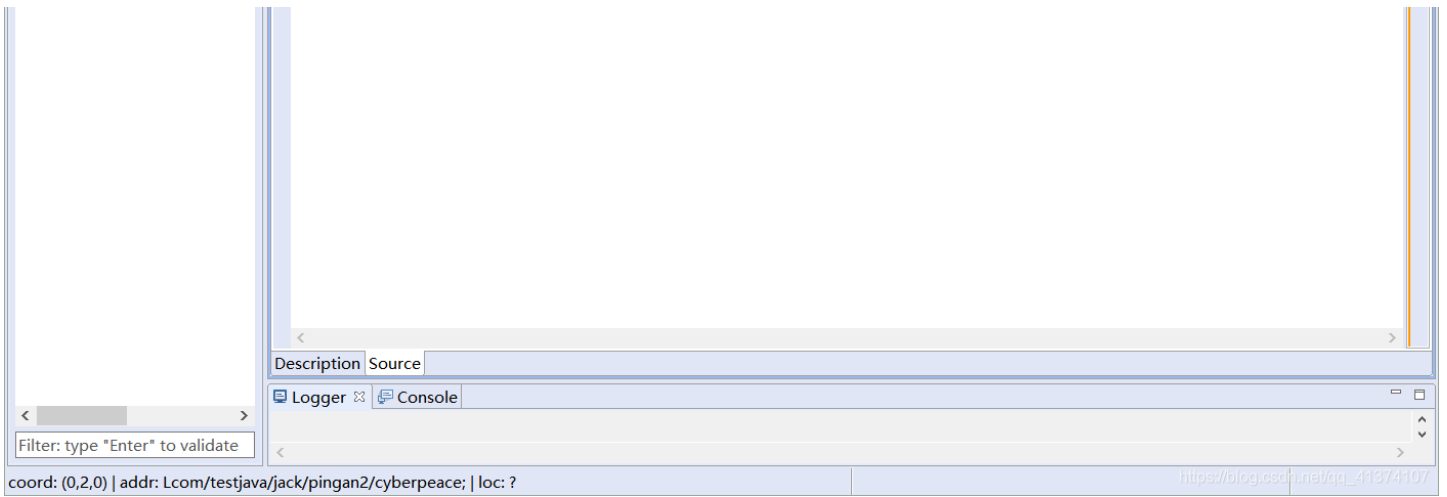
Shell 此apk未采用加固或为未知加固厂商!

Exit

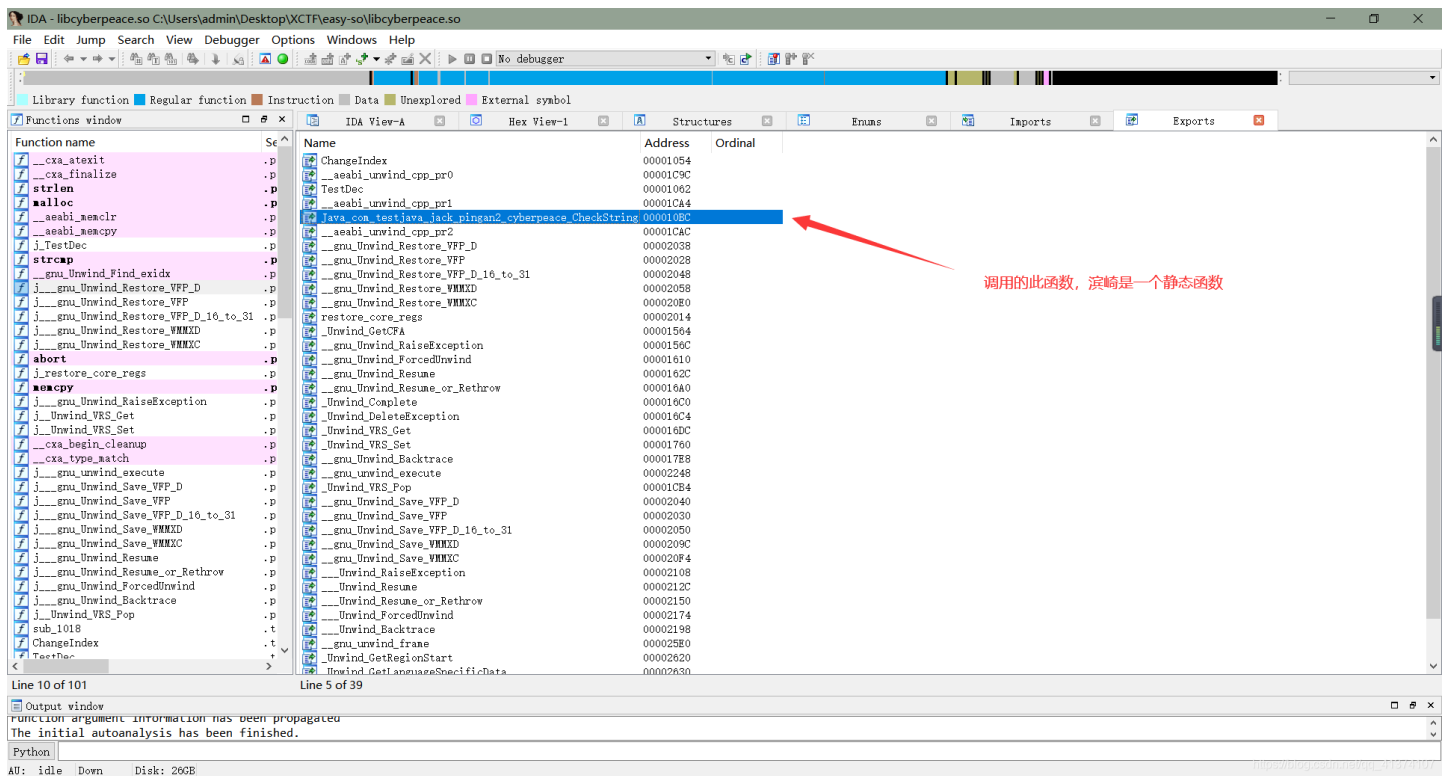
拖放文件至窗口即可!

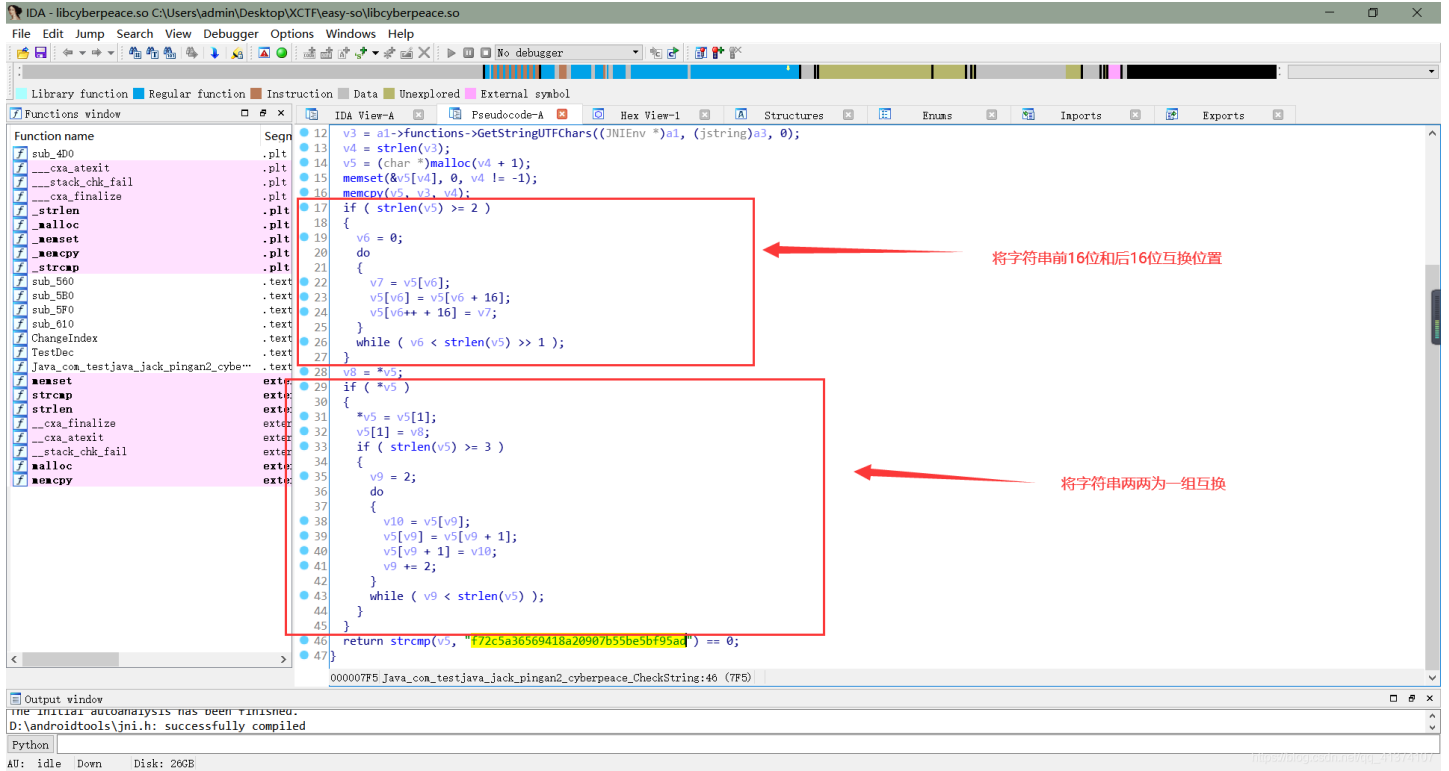
https://blog.csdn.net/ds_41374107





3、用IDA打开so文件（要提取x86文件夹下面那个so文件，两个arm文件夹下面的so文件用ida打开有问题），找到该静态函数，直接F5大法，静态分析该函数可知：首先将传入的字符串前16位与后16位互换，然后两两一组互换位置，最后将得到的字符串与字符串 `f72c5a36569418a20907b55be5bf95ad` 比较返回比较结果！！





4、写了一个python小脚本跑出flag，如下所示：

脚本代码：

```
string = 'f72c5a36569418a20907b55be5bf95ad'
strlist = list(string)
```

```
k = 0
```

```
for i in range(16):
    ch = strlist[1 + k]
    strlist[1 + k] = strlist[0 + k]
    strlist[0 + k] = ch
    k = k + 2
```

```
k = 0
```

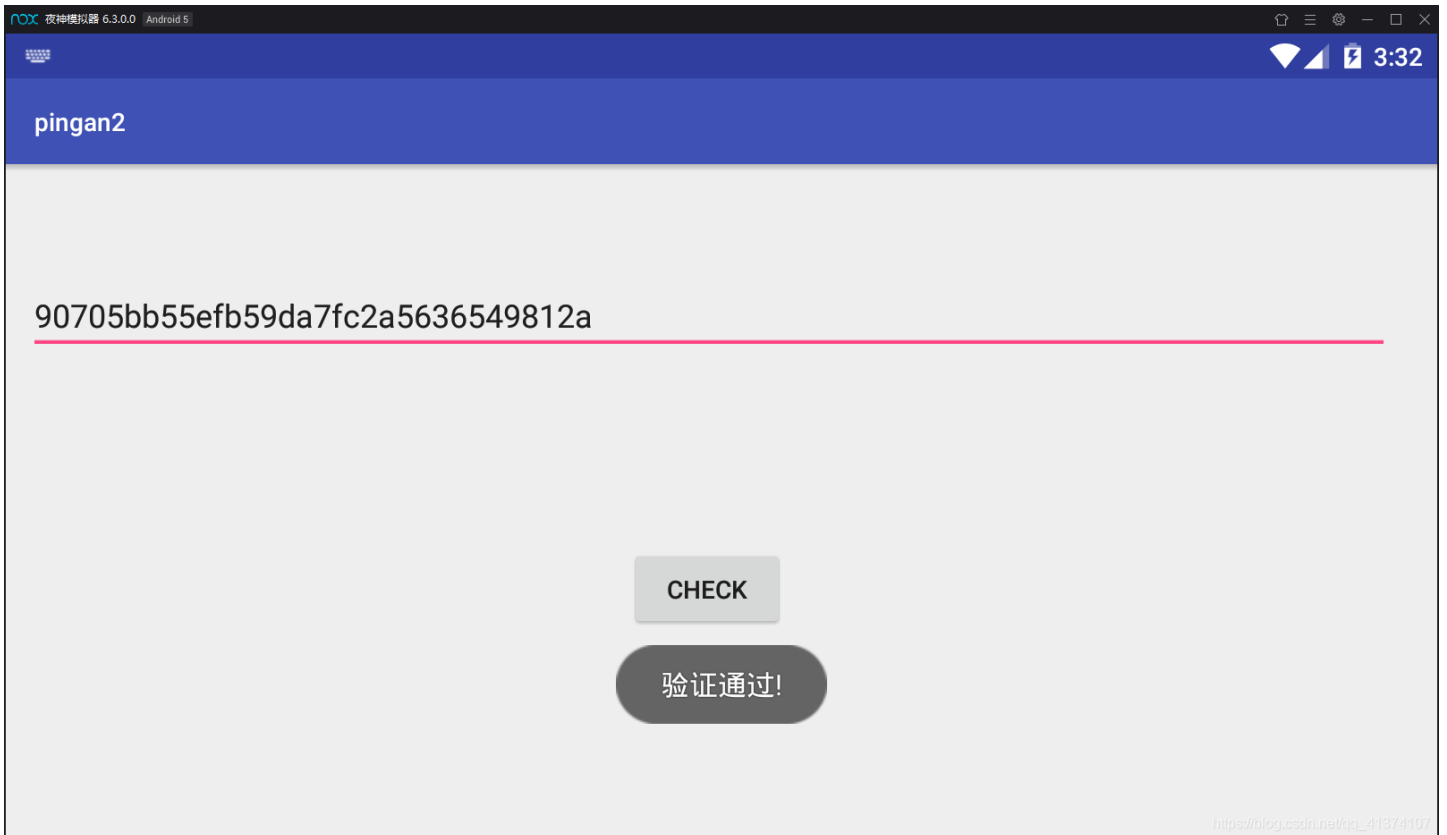
```
for i in range(16):
    ch = strlist[0 + k]
    strlist[0 + k] = strlist[16 + k]
    strlist[16 + k] = ch
    k = k + 1
```

```
print(''.join(strlist))
```

运行截图：

```
easy-so.py - Visual Studio Code
addCodeUi.py main.py SQLMethod.py testSql.py addCode.py JS testrule.js easy-so.py X codeStorageUi.py
E:\> py > easy-so.py > ...
1 string = 'f72c5a36569418a20907b55be5bf95ad'
2 strlist = list(string)
3
4 k = 0
5
6 for i in range(16):
7     ch = strlist[1 + k]
8     strlist[1 + k] = strlist[0 + k]
9     strlist[0 + k] = ch
10    k = k + 2
11
12 k = 0
13
14 for i in range(16):
15     ch = strlist[0 + k]
16     strlist[0 + k] = strlist[16 + k]
17     strlist[16 + k] = ch
18     k = k + 1
19
20 print(''.join(strlist))
21

输出 终端 调试控制台 问题 2: Python
PS C:\Users\admin> & D:/python/python.exe e:/py/easy-so.py
90705bb55efb59da7fc2a5636549812a
PS C:\Users\admin> flag
```

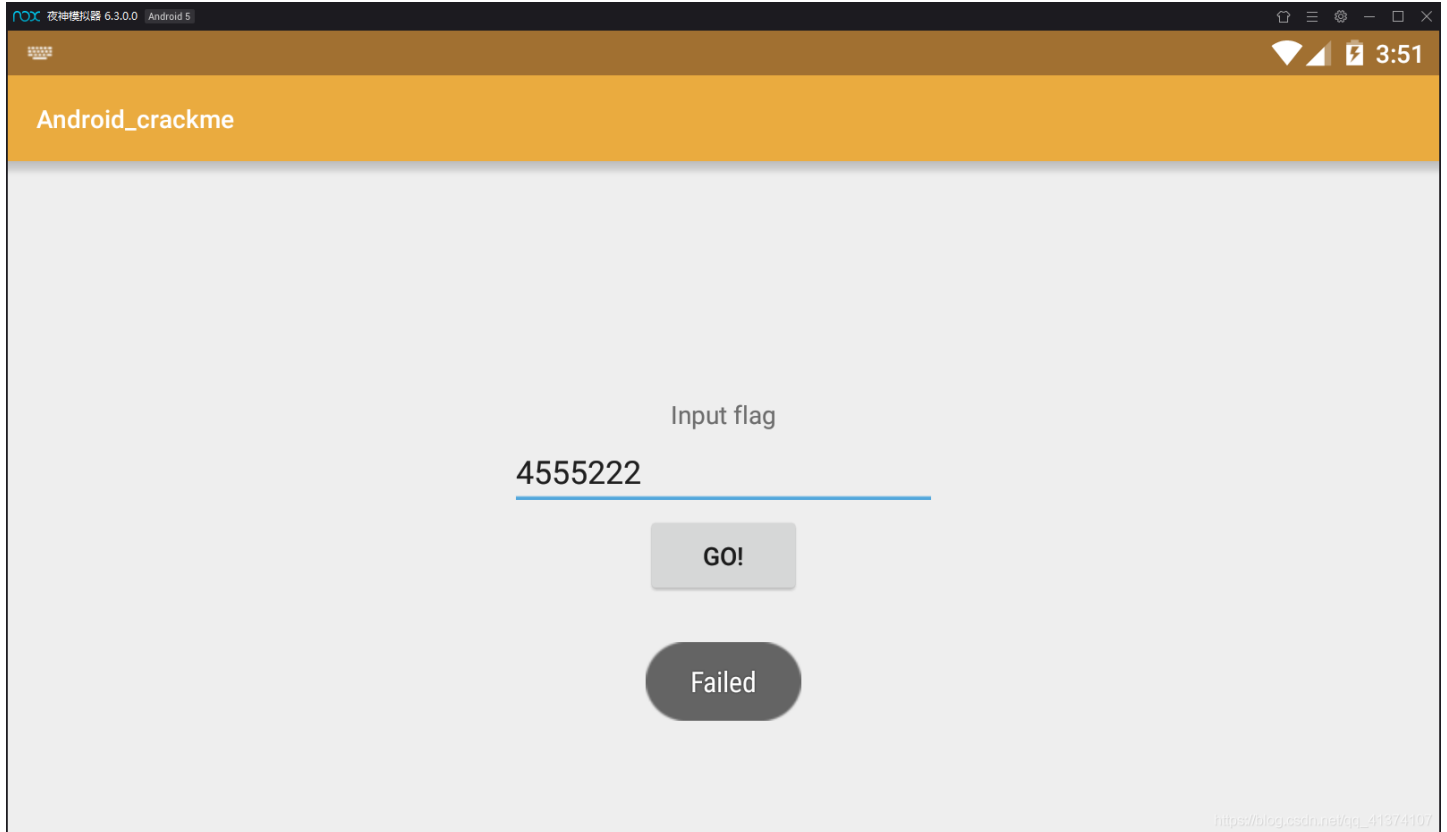


题目: app1

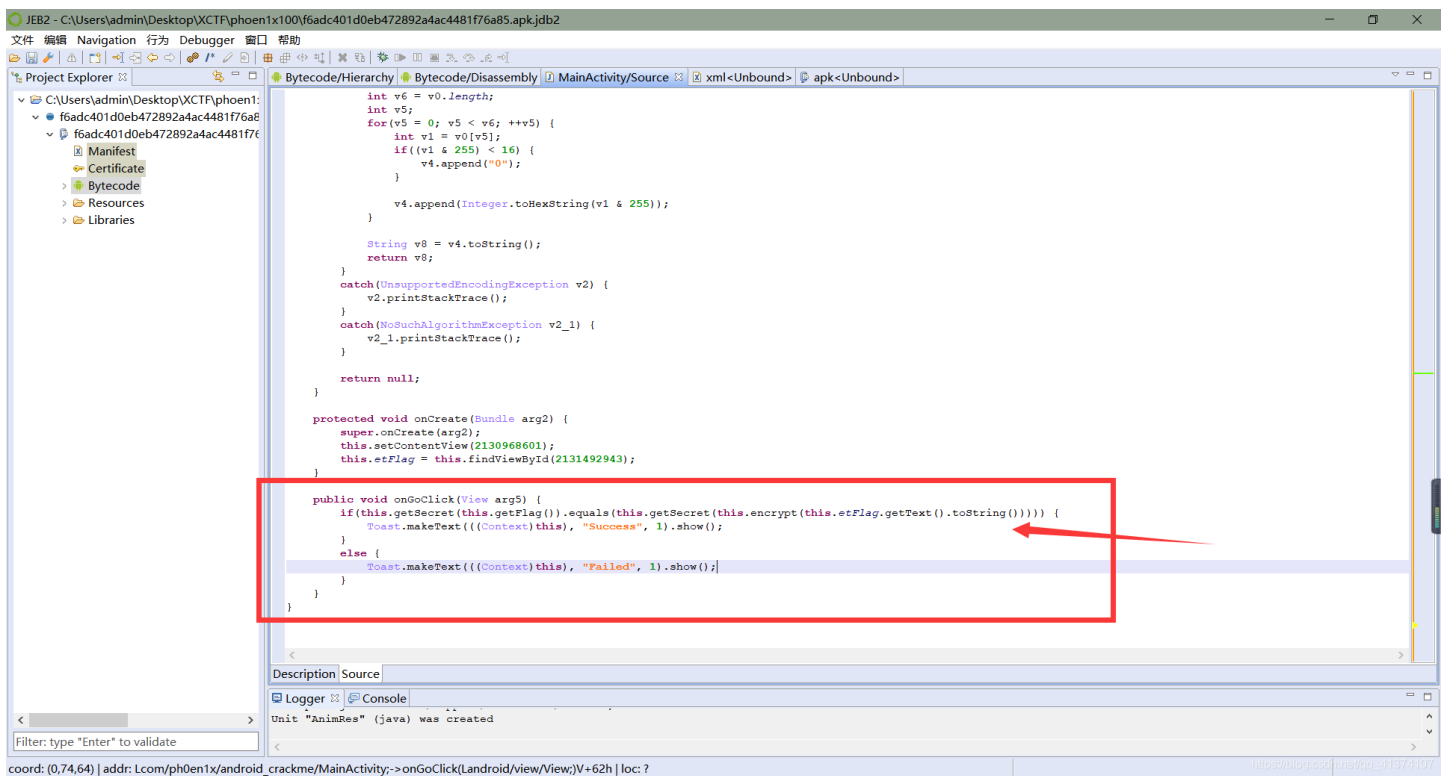
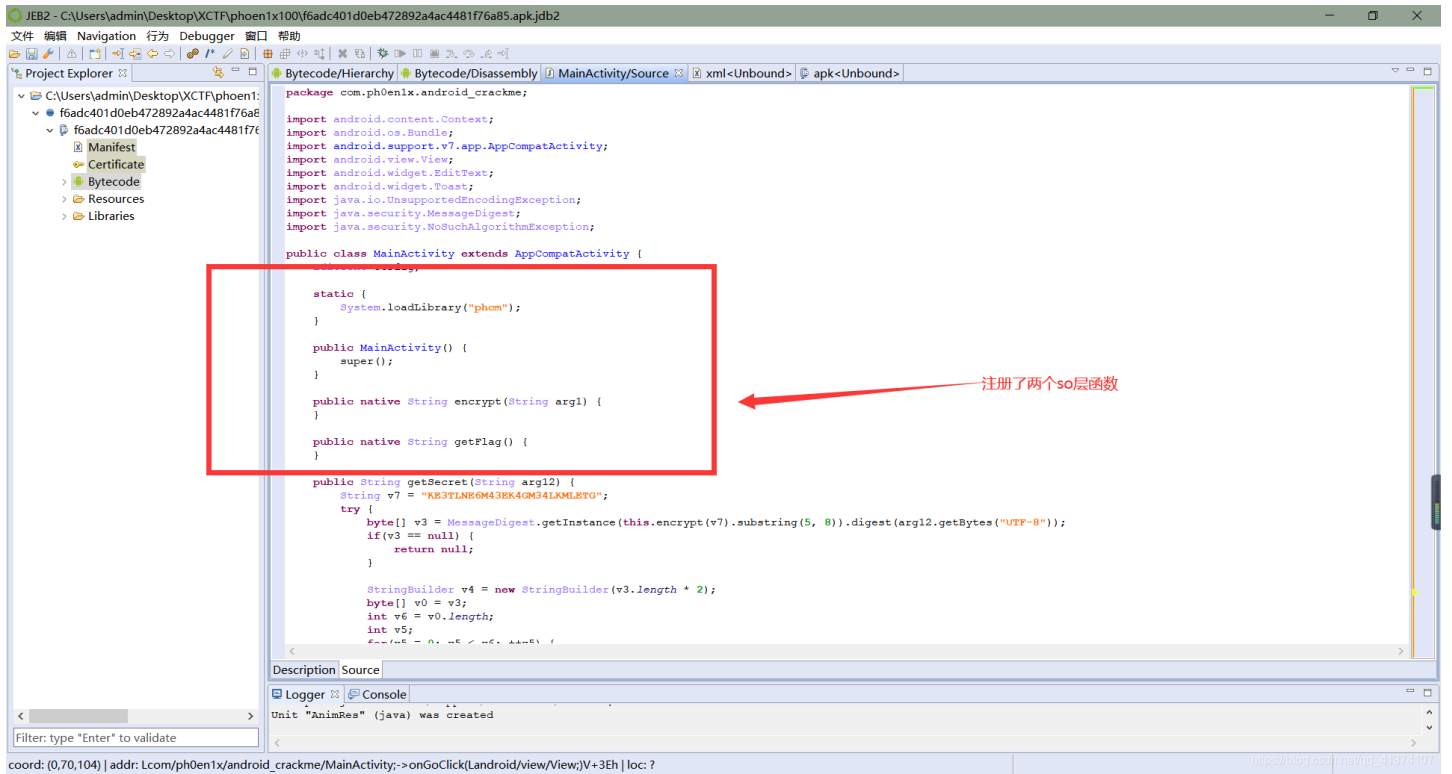
此题wp已经在个人博客写过，不在详细说明，详情请看链接：<https://www.cnblogs.com/aWxvdmVseXc0/p/11902184.html>

题目：Ph0en1x-100

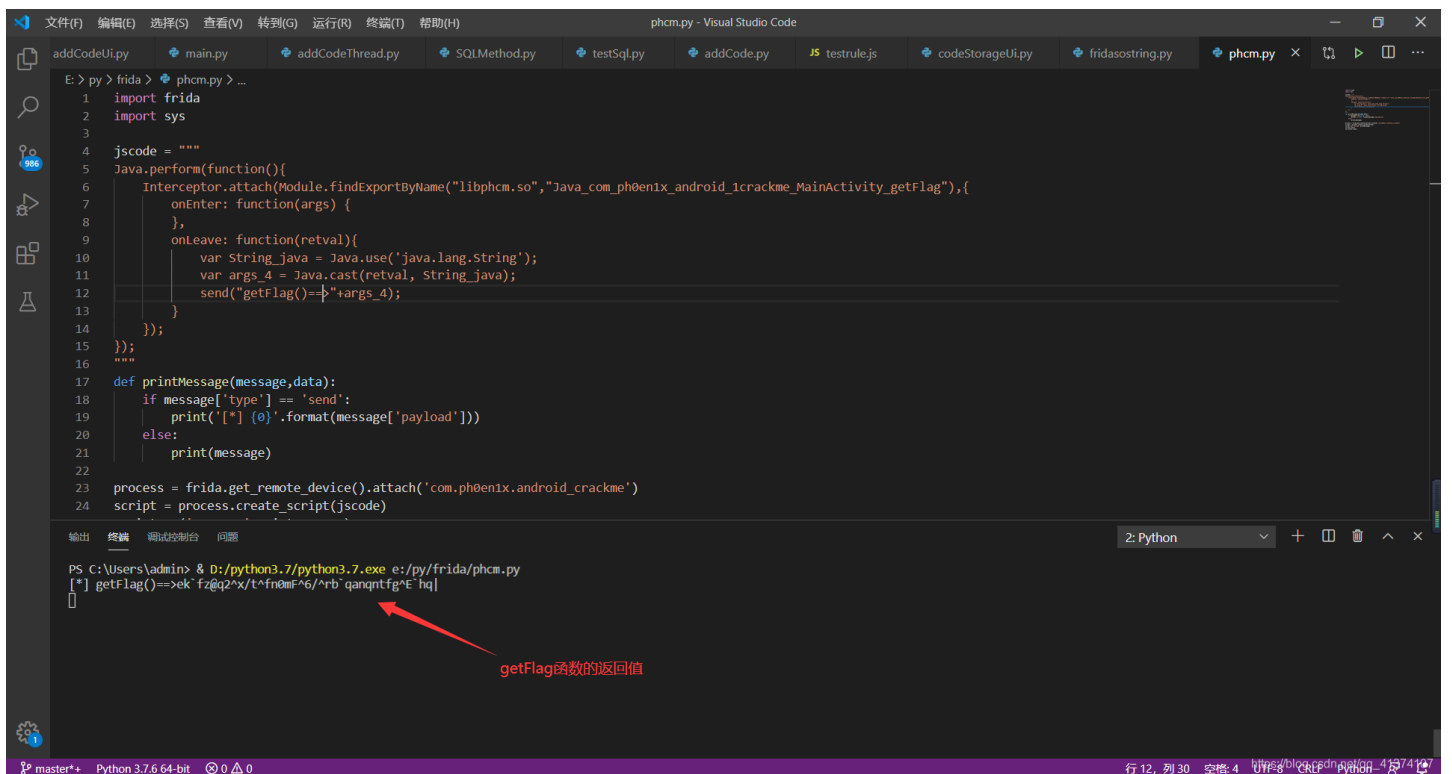
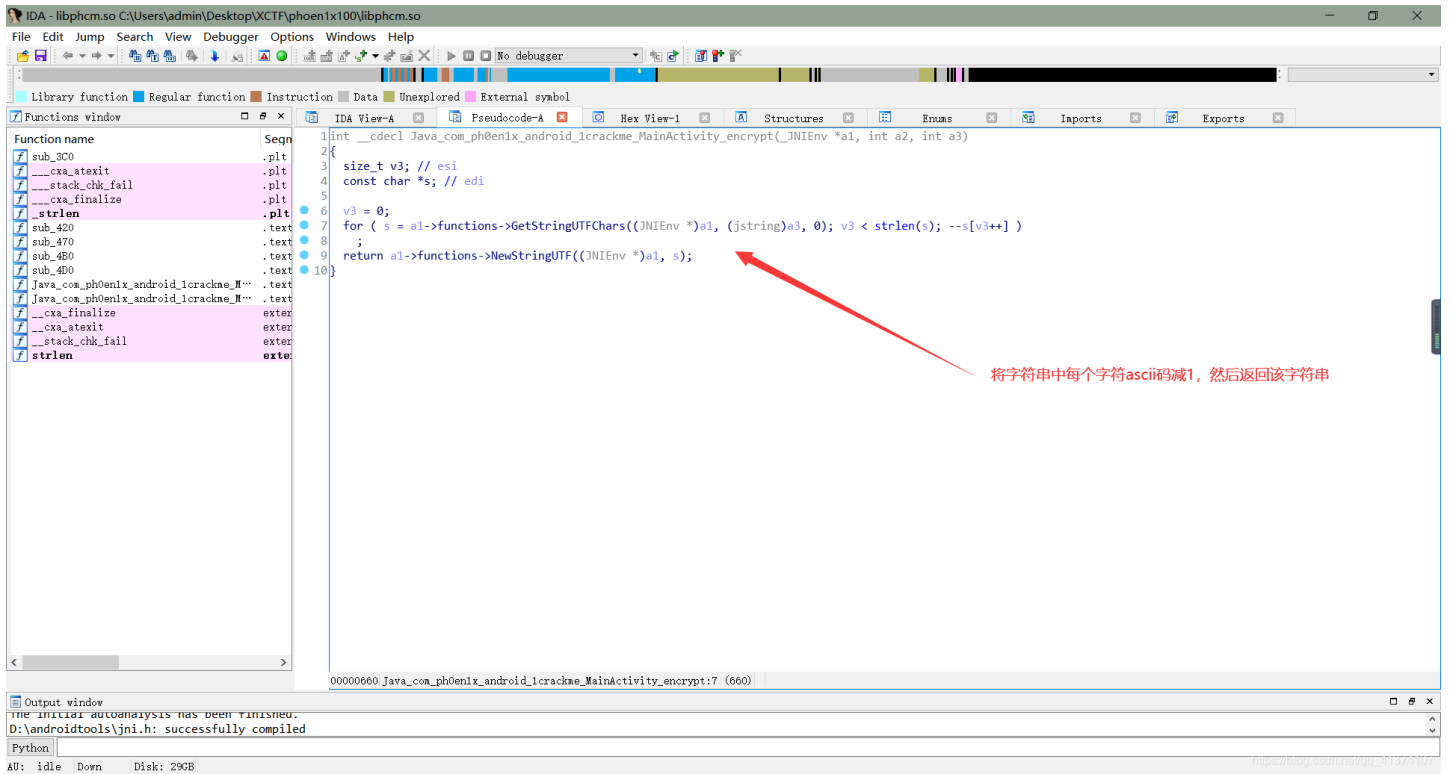
1、拖进夜神中安装运行，主界面只有一个输入框和一个按钮，随便输入信息，点击按钮后，弹出信息 **Failed!**，如下图所示：



2、查壳后无壳直接使用JEB反编译，查看MainActivity.java文件，发现要弹出信息 **Success** 逻辑如下：首先在so层注册了两个静态函数- `encrypt(String)`、`getFlag()` 函数，然后在java层有个函数 `getSecret(String)`，将so层函数 `getFlag` 返回值经过 `getSecret` 函数加密后与我们在输入框中输入的字符串经过 `encrypt` 函数后在经过 `getSecret` 函数加密比较，如果一致，则返回 **Success**，由于比较的两个字符串最外层都经过 `getSecret` 函数加密，所有我们不需要在管 `getSecret` 函数，直接让内部两个字符串一直一致即可得到flag!!!



3、使用IDA打开so文件，静态分析一下 encrypt 函数，发现逻辑很简单，就是将传进来的字符串的每个字符的ASCII码减一；对于 getFlag 函数，由于该函数没有输入只有输出，直接用frida Hook该函数得到返回值即可，如下图所示：



Frida代码:

```

import frida
import sys

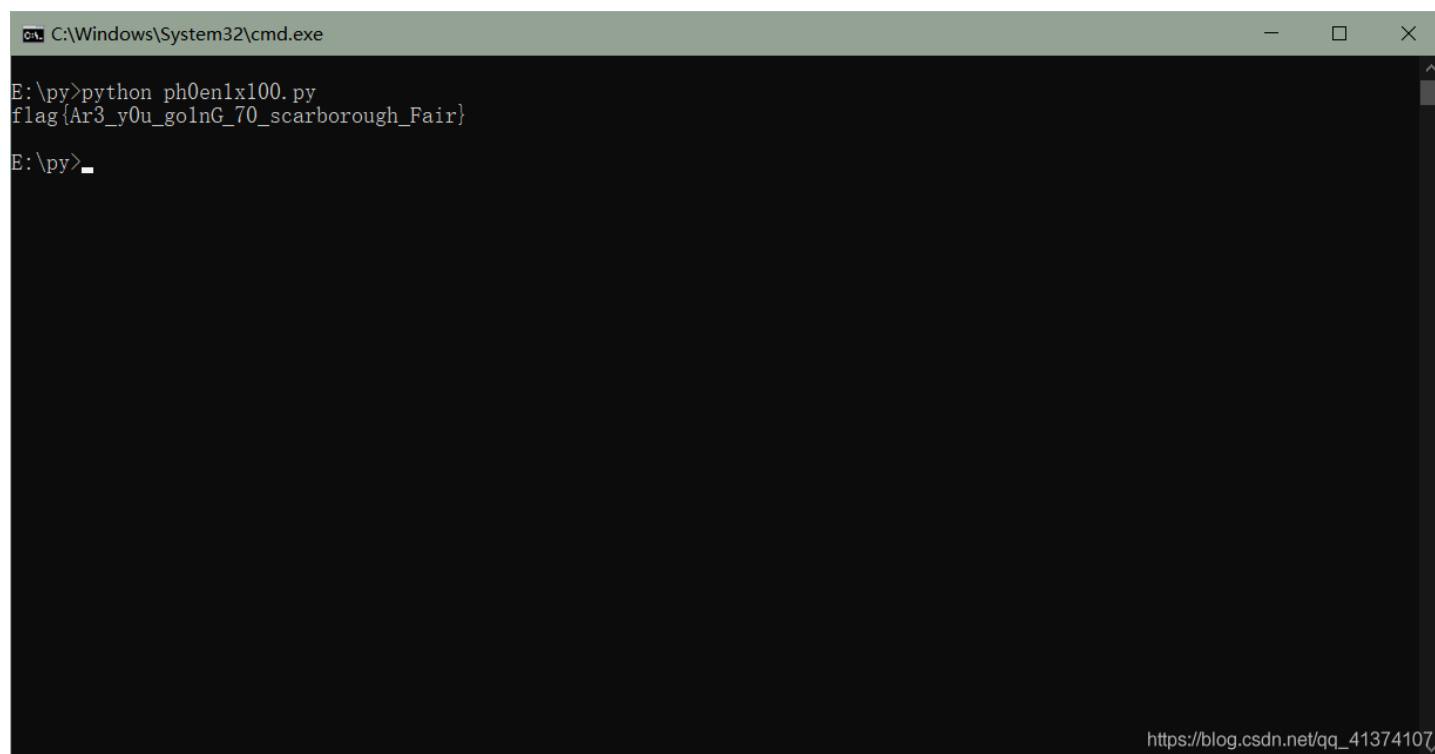
jrcode = """
Java.perform(function(){
    Interceptor.attach(Module.findExportByName("libphcm.so","Java_com_ph0en1x_android_1crackme_MainActivity_getFlag"),{
        onEnter: function(args) {
        },
        onLeave: function(retval){
            var String_java = Java.use('java.lang.String');
            var args_4 = Java.cast(retval, String_java);
            send("getFlag()==>" + args_4);
        }
    });
});
"""

def printMessage(message,data):
    if message['type'] == 'send':
        print('[*] {0}'.format(message['payload']))
    else:
        print(message)

process = frida.get_remote_device().attach('com.ph0en1x.android_crackme')
script = process.create_script(jrcode)
script.on('message',printMessage)
script.load()
sys.stdin.read()

```

4、得到以上信息后，使用python脚本跑出flag即可，如下所示：



```

C:\Windows\System32\cmd.exe
E:\py>python ph0en1x100.py
flag{Ar3_y0u_golnG_70_scarborough_Fair}
E:\py>

```

https://blog.csdn.net/qq_41374107



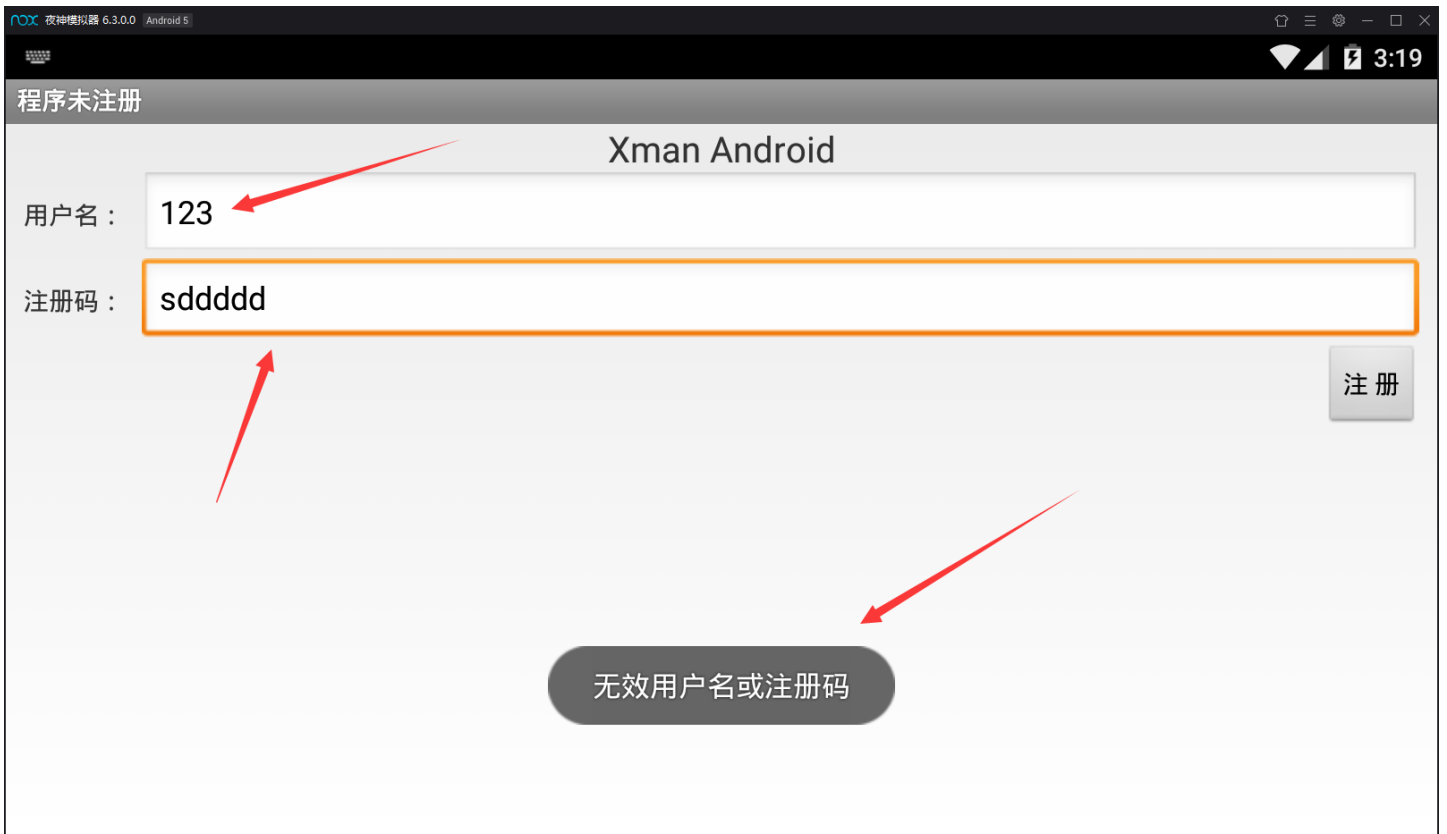
python脚本:

```
Flag = 'ek`fz@q2^x/t^fn0mF^6/^rb`qanqntfg^E`hq|'  
flaglist = list(Flag)  
stringlist = []  
for ch in flaglist:  
    stringlist.append(chr(ord(ch) + 1))  
print(''.join(stringlist))
```

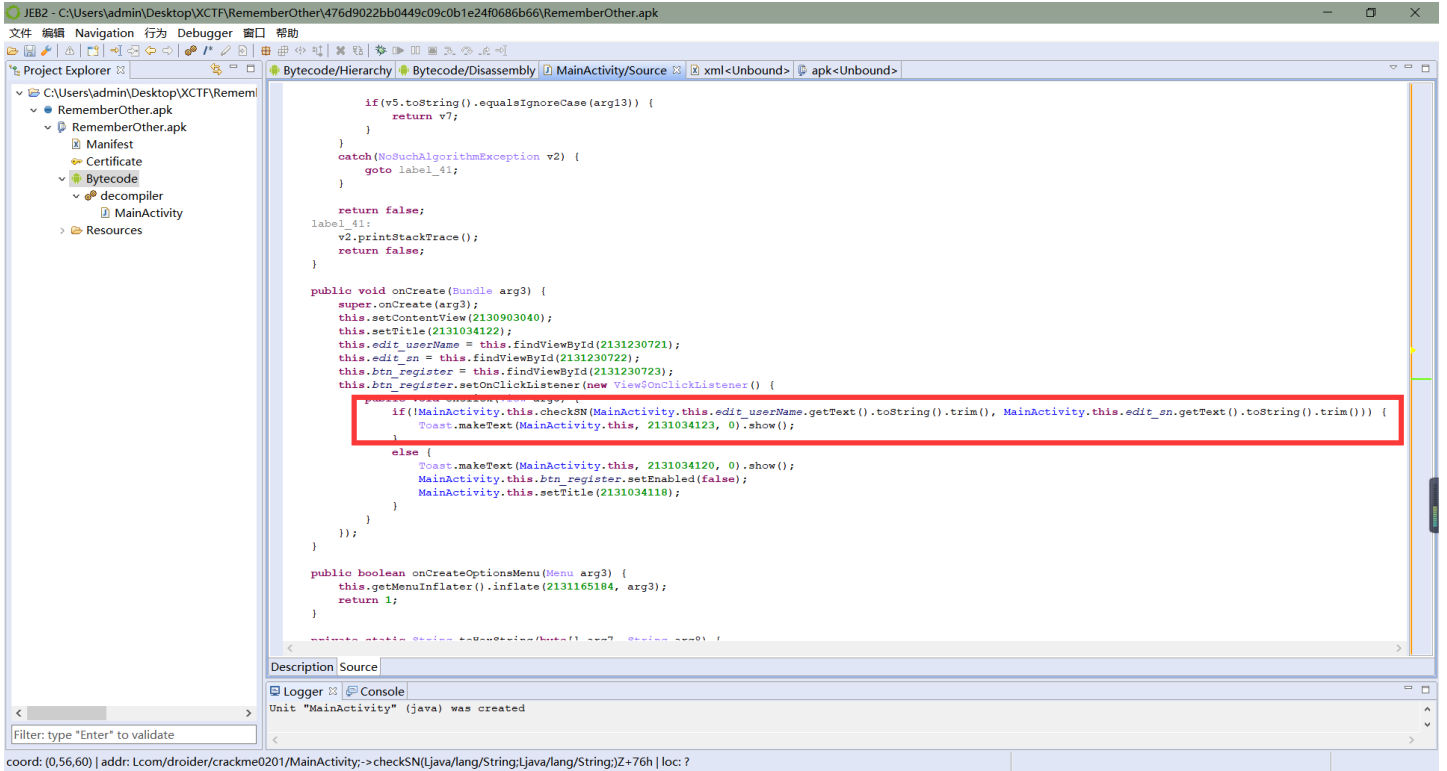
题目: RememberOther

PS: 此题为脑洞题, 披了一个安卓的皮而已!!!

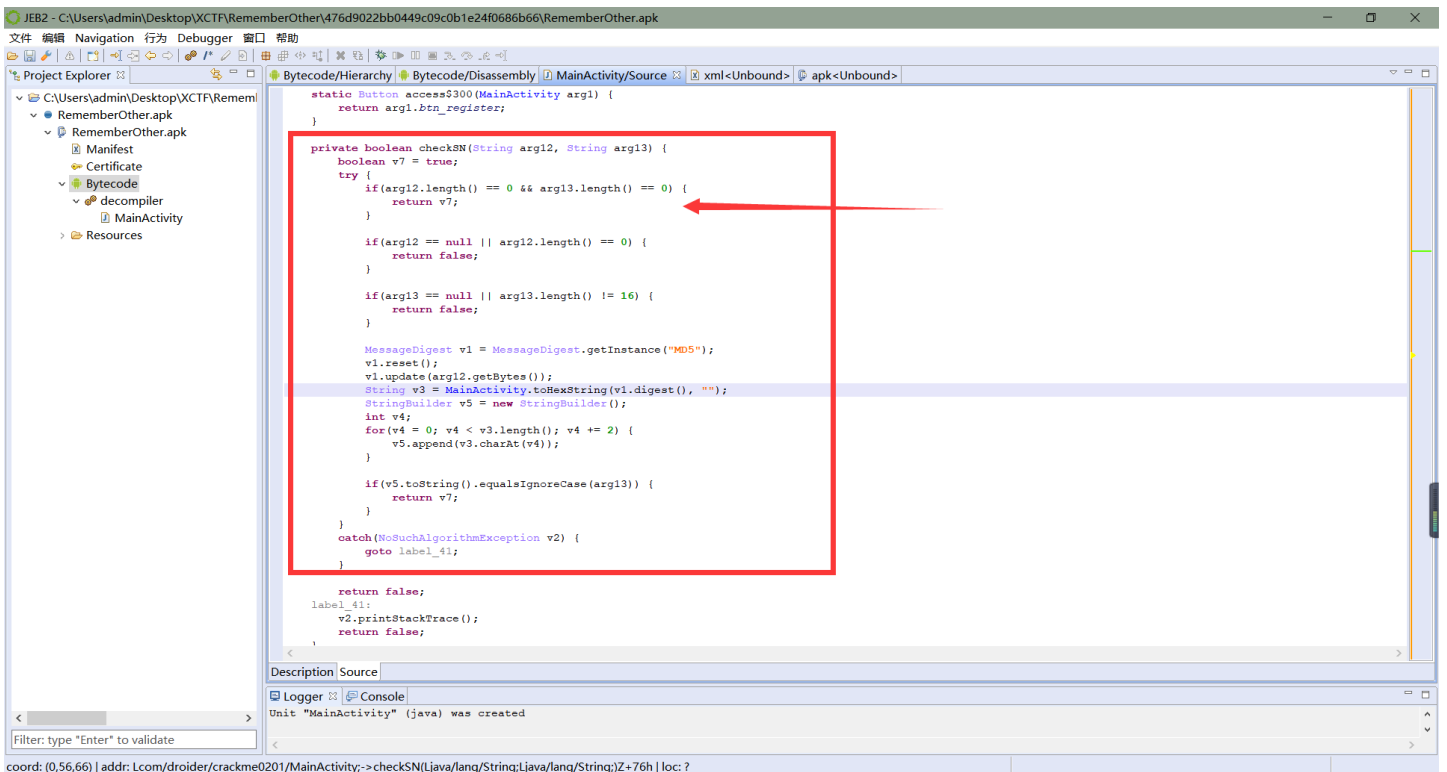
1、下载好题目后, 查壳发现无壳, 直接拖进夜神中, 要求输入用户名和注册码, 随便输入, 弹出信息 **无效用户名或注册码**, 如下图所示:



2、用JEB反编译后，查看 `onCreate` 方法，发现将我们输入的用户名和注册码作为参数调用 `checkSN` 函数，并且当该函数返回 `false` 时不弹出信息 `无效用户名或注册码`，接着去看 `checkSN` 函数，该函数当用户名和注册码为空时返回 `false`，返回 `false` 后弹出了一串 `md5` 值，再看 `checkSN` 函数其他逻辑，发现将输入的用户名经过 `md5` 加密后返回 `16` 进制字符串，然后取该字符串的奇数位拼接成一个新的字符串，再然后与我们输入的注册码进行比较，返回 `true`，最后也没有发现这个跟 `flag` 没什么关系，想起之前还有一串 `md5` 值，进行解密，解密得出 `YOU_KNOW_`，输入，提示 `flag` 错误。。。看了一下其他大佬的 `wp`，才发现在后面加上 `ANDROID` 就行了。。。。。。因为压缩包里面有个 `word` 文件，里面写了不懂安卓。。。。。。。



coord: (0,56,60) | addr: Lcom/droider/crackme0201/MainActivity;->checkSN(Ljava/lang/String;Ljava/lang/String;)Z+76h | loc: ?



coord: (0,56,66) | addr: Lcom/droider/crackme0201/MainActivity;->checkSN(Ljava/lang/String;Ljava/lang/String;)Z+76h | loc: ?

打开 隐藏面板 配置 关于
文件 视图 选项 帮助

开始 RememberOther

工程信息 工程管理器 工程搜索

搜索字符:
succeeded
搜索

替换字符:
全部替换

搜索选项:
搜索范围: 当前整个项目
文件类型: .smali|.xml|.txt|.htm|.html
字符编码: UTF8
 匹配大小写

搜索进度:

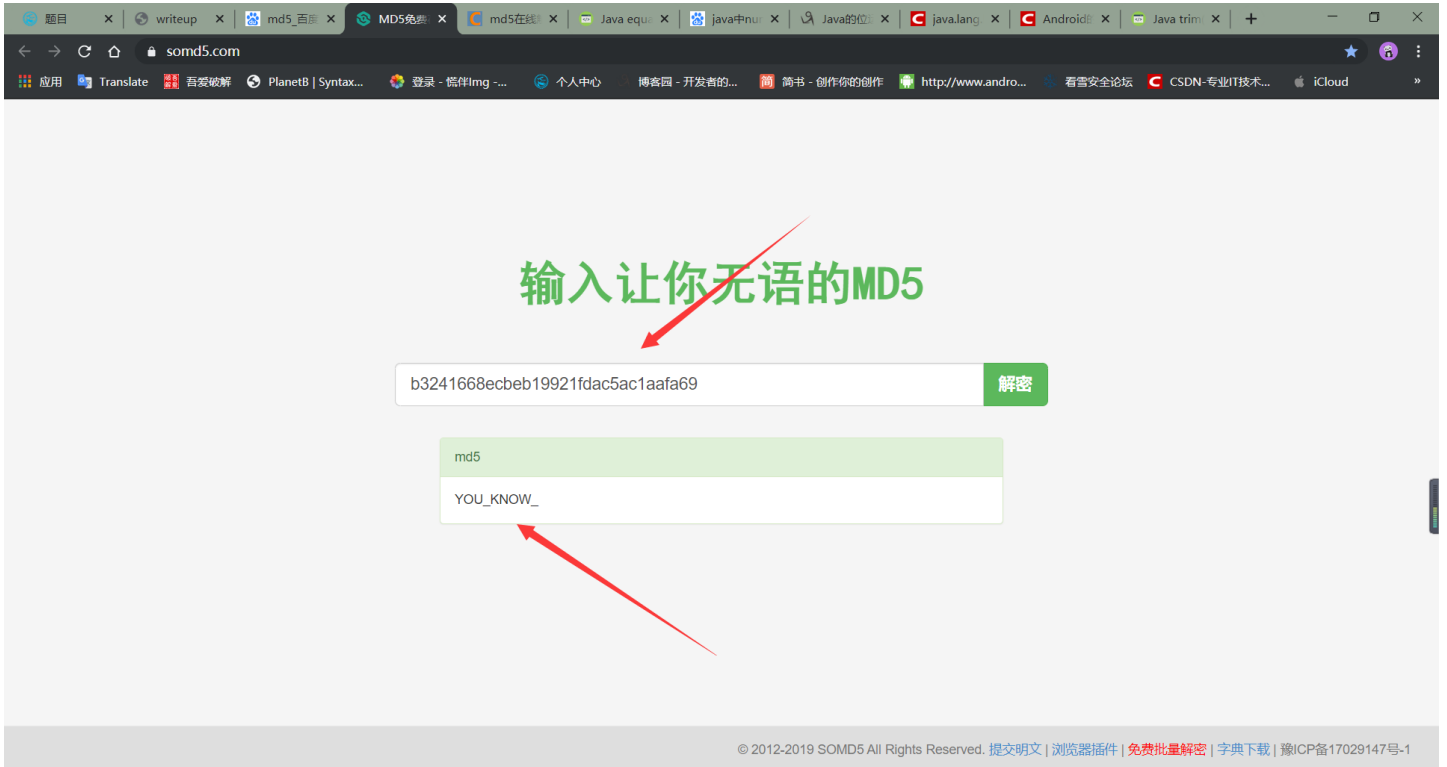
```
strings.xml public.xml  
4 <string name="hint_sn">请输入16位的注册码</string>  
5 <string name="hint_username">请输入用户名</string>  
6 <string name="info">Xman Android</string>  
7 <string name="menu_settings">Settings</string>  
8 <string name="register">注册</string>  
9 <string name="registered">程序已注册</string>  
10 <string name="sn">注册码: </string>  
11 <string name="succeeded">md5:b3241668ecbeb19921fdac5ac1aafa69</string>  
12 <string name="title_activity_main">Crackme</string>  
13 <string name="unregister">程序未注册</string>  
14 <string name="unsuccessful">无效用户名或注册码</string>  
15 <string name="username">用户名: </string>  
16 </resources>  
17
```

行: 11 列: 66 插入

b324 unsuccessful 0x7f05000b 7f050008 succeeded

- res\values\public.xml
- res\values\strings.xml
 - <string name="succeeded">md5:b3241668ecbeb19921fdac5ac1aafa69</string>
 - <string name="unsuccessful">无效用户名或注册码</string>
- smali\com\droider\crackme0201\R\$string.smali

日志输出 搜索结果 方法引用



帮助  告诉我你想要做什么

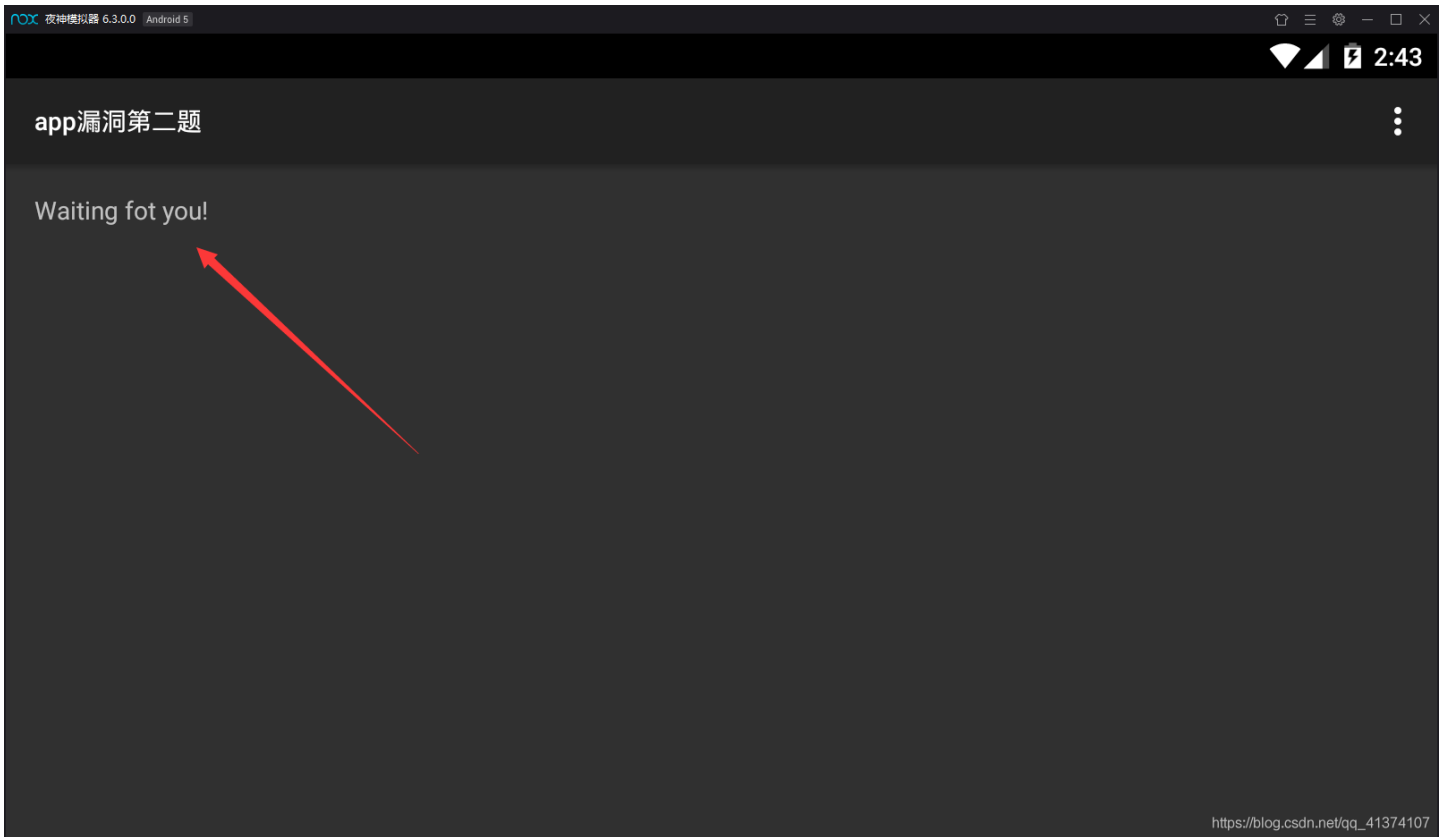
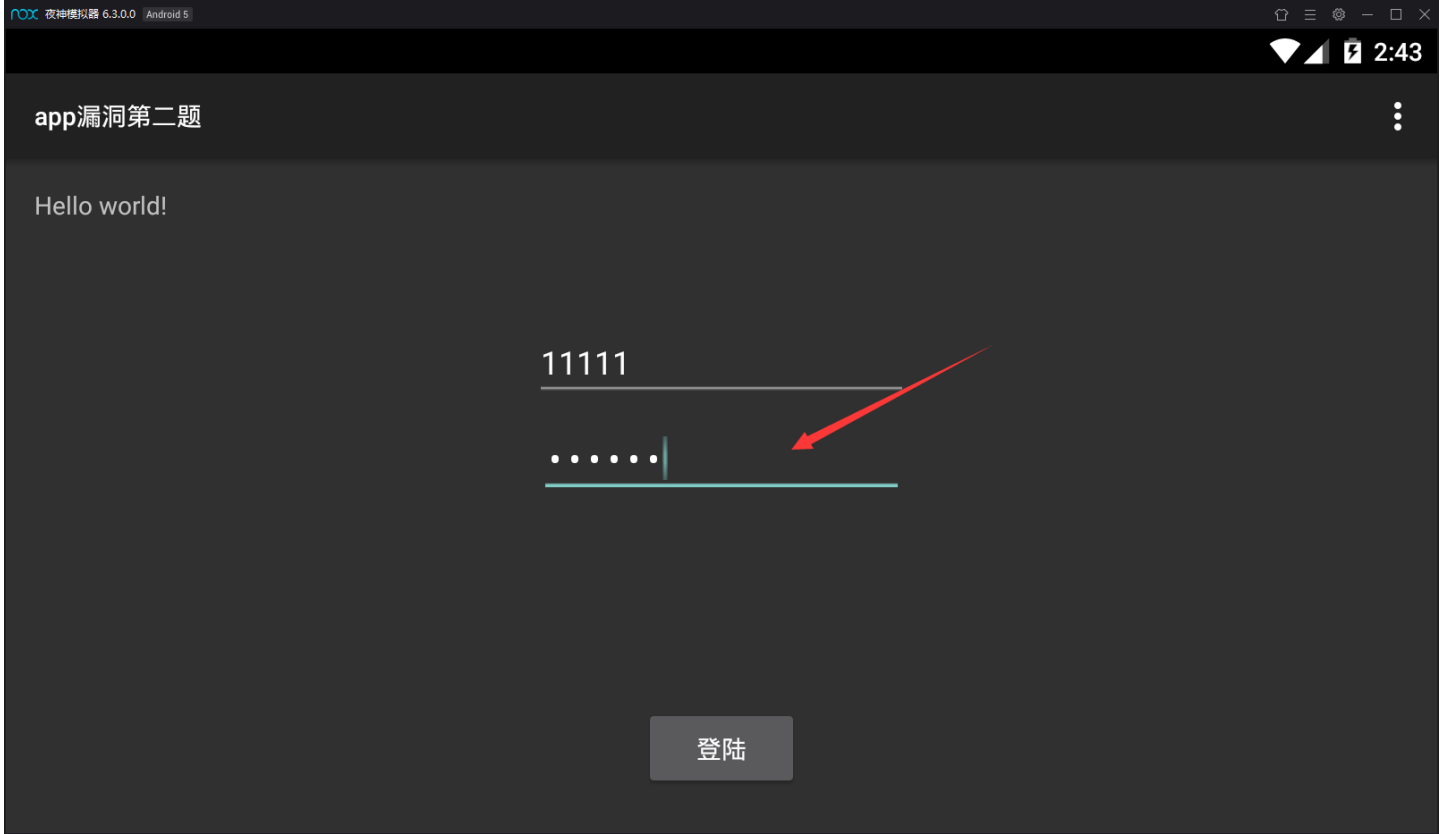
, 否则保持在受保护视图中比较安全。

启用编辑(E)

不懂安卓，所以就只是和安卓扯了扯边,,, Have fun~

题目：app2

- 1、将下载好的题目拖进夜神中，发现要求登陆，随便点击登陆后，如下所示：



2、将apk用jeb反编译后，首先看一下MainActivity这个入口文件，发现没什么，就是将我们输入的用户名和密码传入SecondActivity页面中，然后跳转到该页面，再来看一下SecondActivity，发现调用了so层函数doRawData，使其返回值和字符串VEIzd/V2UPYNdn/bxH3Xi==进行比较，如下所示：

JEB2 - C:\Users\admin\Desktop\XCTF\app2\2554cf208cfb4cdf9218a840fa9bf237.apk

文件 编辑 Navigation 行为 Debugger 窗口 帮助

Project Explorer

- C:\Users\admin\Desktop\XCTF\app2\2554cf208cfb4cdf9218a840fa9bf237.apk
 - 2554cf208cfb4cdf9218a840fa9bf237.apk
 - Manifest
 - Certificate
 - Bytecode
 - Resources
 - Libraries

Bytecode/Hierarchy | Bytecode/Disassembly | MainActivity/Source | xml<Unbound> | apk<Unbound>

```
private EditText c;
private EditText d;

public MainActivity() {
    super();
    this.b = null;
}

public void onClick(View arg6) {
    switch(arg6.getId()) {
        case 2131165187: {
            if(this.c.getText().length() != 0 && this.d.getText().length() != 0) {
                String v0 = this.c.getText().toString();
                String v1 = this.d.getText().toString();
                Log.e("test", v0 + " test2 = " + v1);
                Intent v2 = new Intent(((Context)this), SecondActivity.class);
                v2.putExtra("ili", v0);
                v2.putExtra("iil", v1);
                this.startActivity(v2);
                return;
            }

            Toast.makeText(((Context)this), "不能为空", 1).show();
            break;
        }
    }
}

protected void onCreate(Bundle arg5) {
    super.onCreate(arg5);
    this setContentView(2130903040);
    this.a = this.findViewById(2131165187);
    this.a.setOnClickListener(((View$OnClickListener)this));
}
```

Logger Console

Filter: type "Enter" to validate

coord: (0,39,43) | addr: Lcom.tencent/testvuln/MainActivity;->onClick(Landroid/view/View;)V+D2h | loc: ?

https://blog.csdn.net/qq_41374107

JEB2 - C:\Users\admin\Desktop\XCTF\app2\2554cf208cfb4cdf9218a840fa9bf237.apk

文件 编辑 Navigation 行为 Debugger 窗口 帮助

Project Explorer

- C:\Users\admin\Desktop\XCTF\app2\2554cf208cfb4cdf9218a840fa9bf237.apk
 - 2554cf208cfb4cdf9218a840fa9bf237.apk
 - Manifest
 - Certificate
 - Bytecode
 - Resources
 - Libraries

Bytecode/Hierarchy | Bytecode/Disassembly | SecondActivity/Source | xml<Unbound> | apk<Unbound>

```
super();

public void onReceive(Context arg3, Intent arg4) {
    Toast.makeText(arg3, "myReceiver receive", 0).show();
    arg3.getPackageName().equals(arg4.getAction());
}

private BroadcastReceiver c;

public SecondActivity() {
    super();
    this.c = new com.tencent.testvuln.SecondActivity$1(this);
}

protected void onCreate(Bundle arg6) {
    super.onCreate(arg6);
    this setContentView(2130903041);
    Intent v0 = this getIntent();
    String v1 = v0.getStringExtra("ili");
    String v2 = v0.getStringExtra("iil");
    if(Encrypto.doRawData(this, v1 + v2).equals("VEIzd/V2UPYNdn/bxH3Xig==")) {
        v0.setAction("android.test.action.MoniterInstallService");
        v0.setClass(((Context)this), MoniterInstallService.class);
        v0.putExtra("company", "tencent");
        v0.putExtra("name", "hacker");
        v0.putExtra("age", 18);
        this.startActivity(v0);
        this.startService(v0);
    }

    SharedPreferences$Editor v0_1 = this.getSharedPreferences("test", 0).edit();
}
```

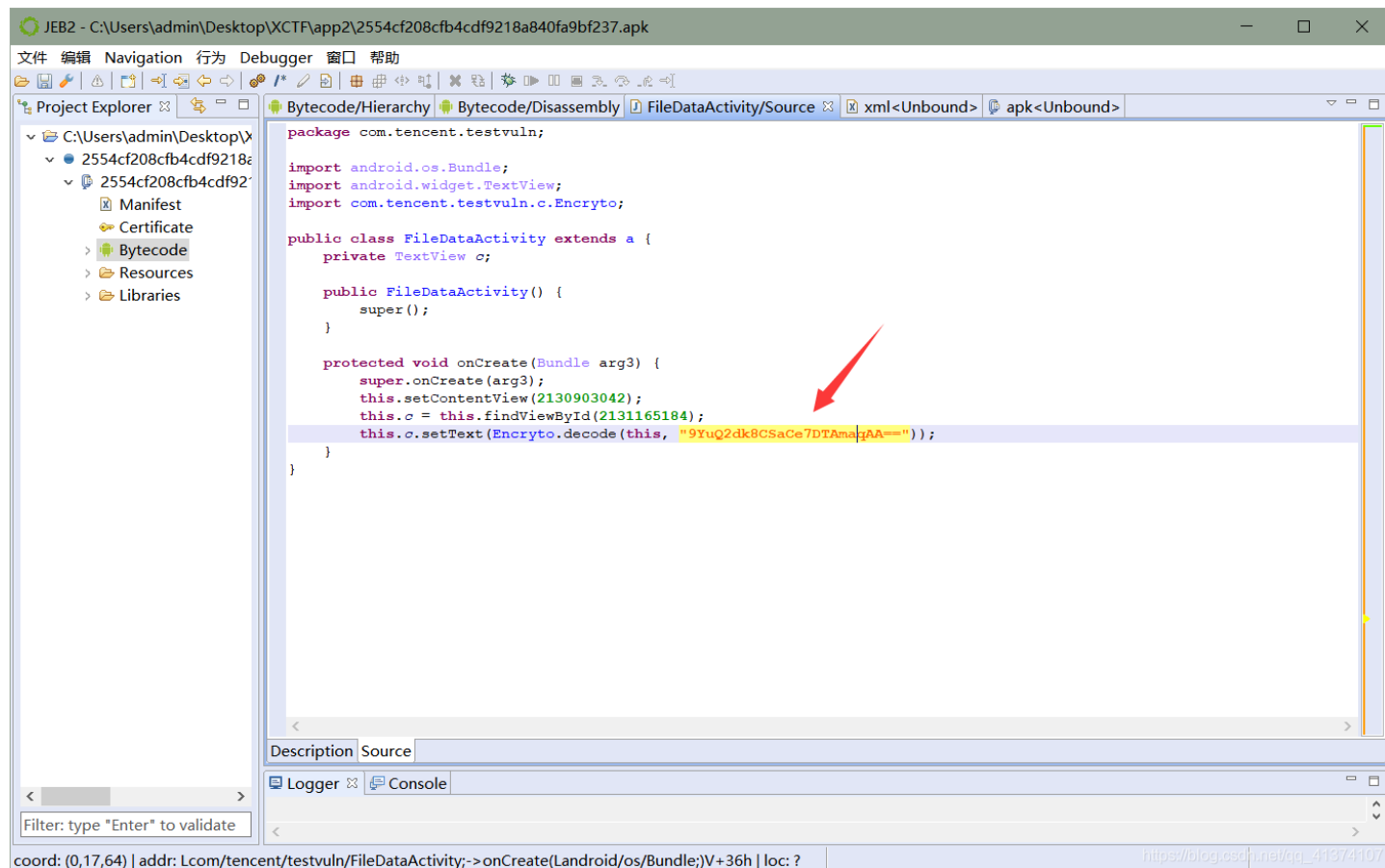
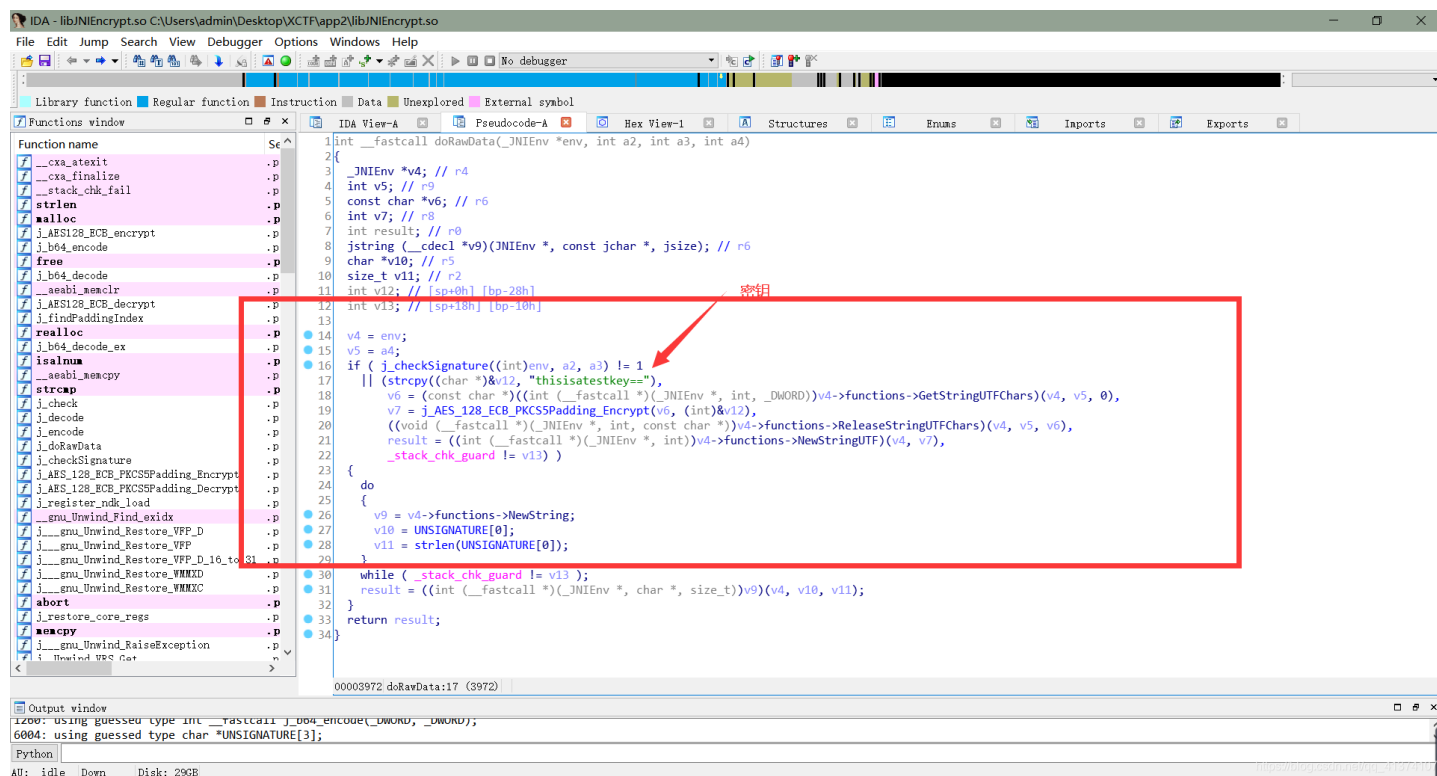
Logger Console

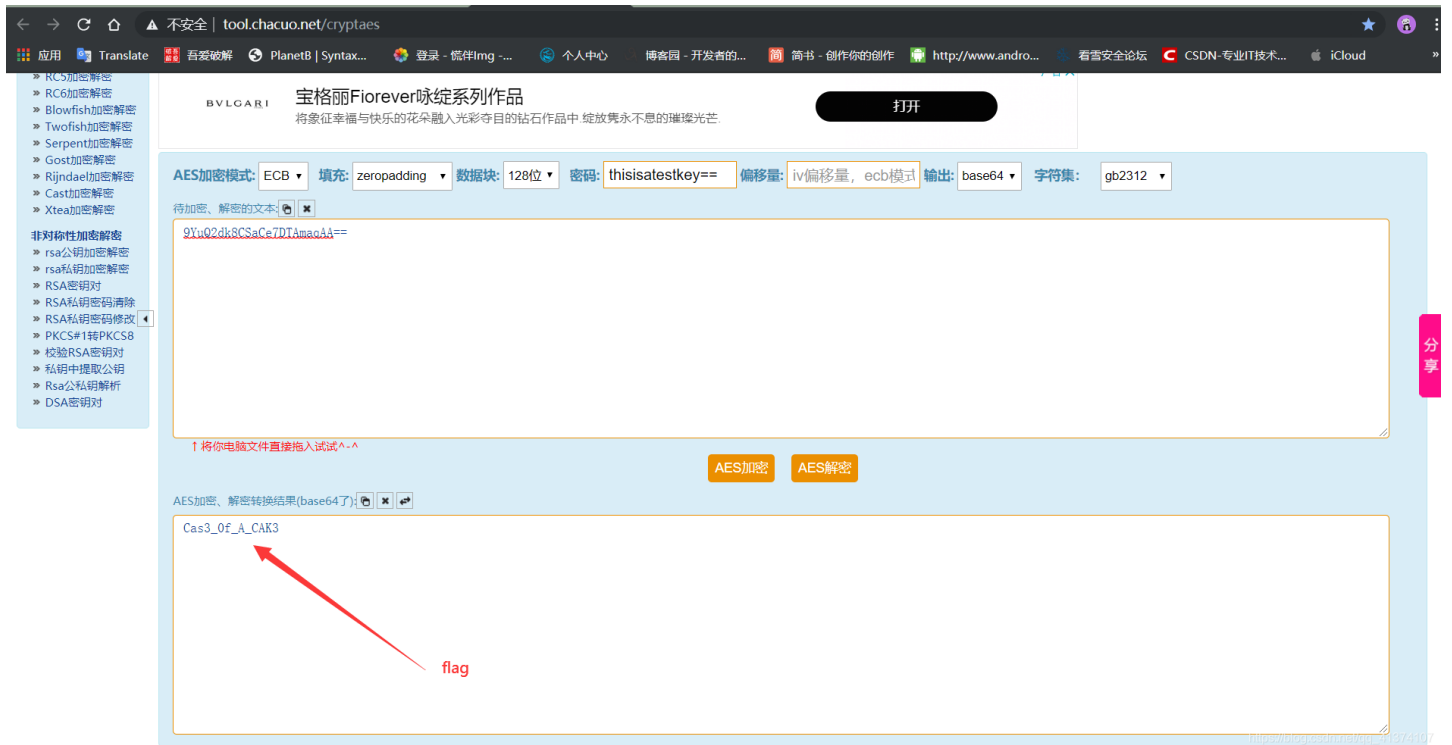
Filter: type "Enter" to validate

coord: (0,38,81) | addr: Lcom.tencent/testvuln/SecondActivity;->onCreate(Landroid/os/Bundle;)V+68h | loc: ?

https://blog.csdn.net/qq_41374107

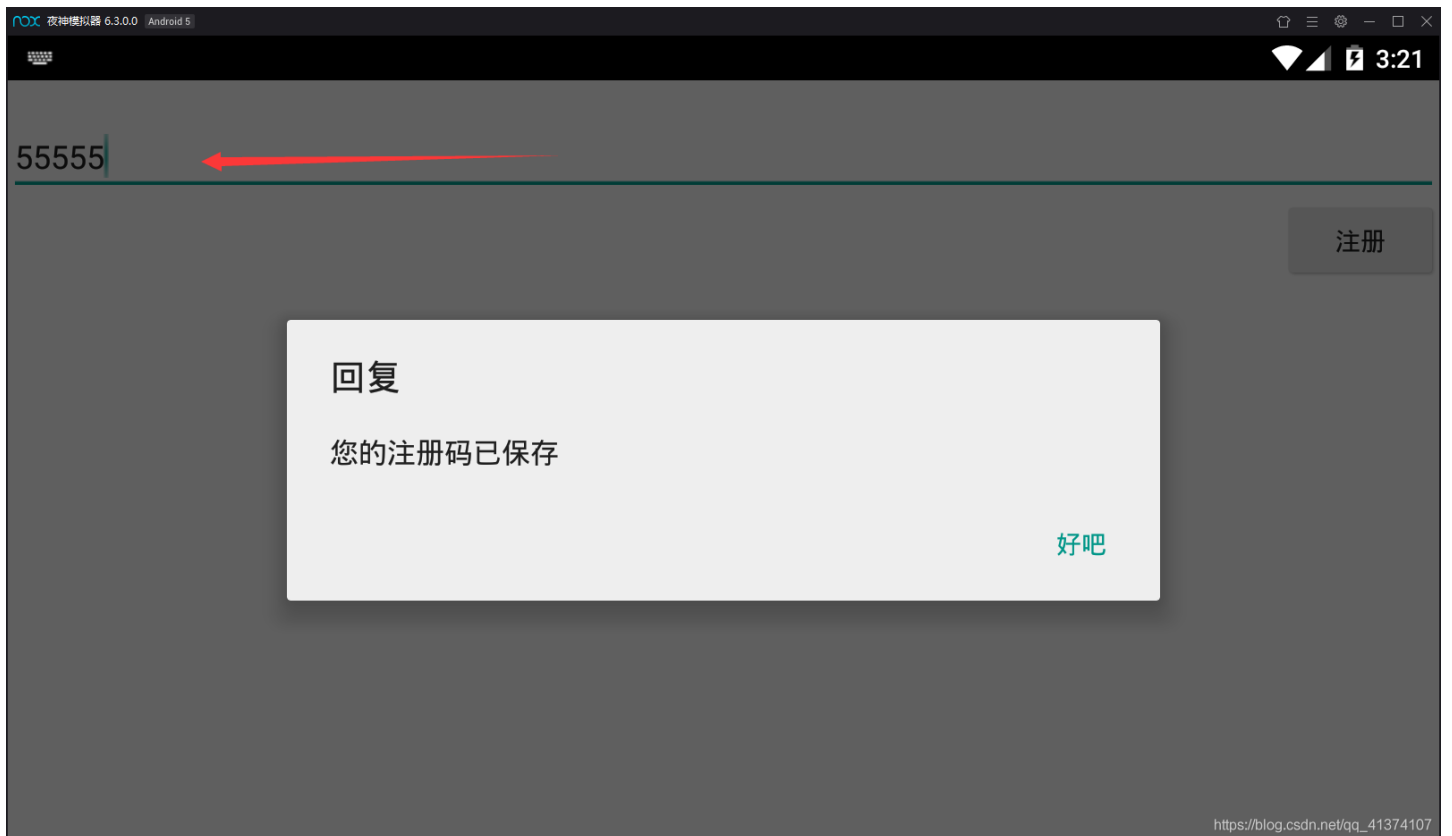
3、用IDA查看一下该函数，发现至少将传入的字符串进行了 AES-128-ECB 加密，并且发现密钥 `thisisatestkey==`，将开始我们发现的字符串 `VEIzd/V2UPYNdn/bxH3Xig==` 进行解密，得到的字符串输入后发现不是 flag，但是在 FileDataActivity 文件中发现另一串字符串 `9YuQ2dk8CSaCe7DTAmaqAA==`，对其解密，得到 flag: `Cas3_of_A_CAK3`



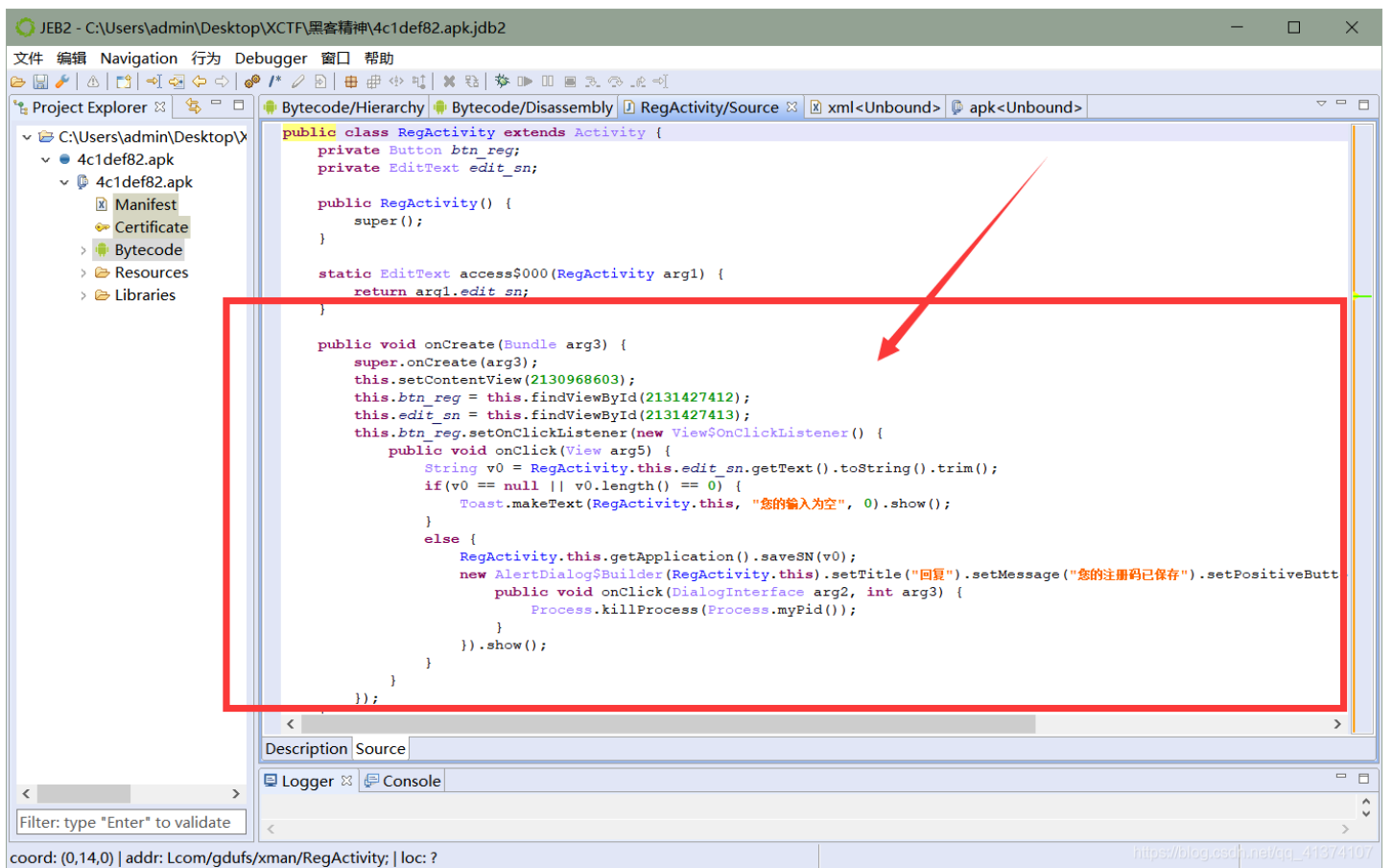
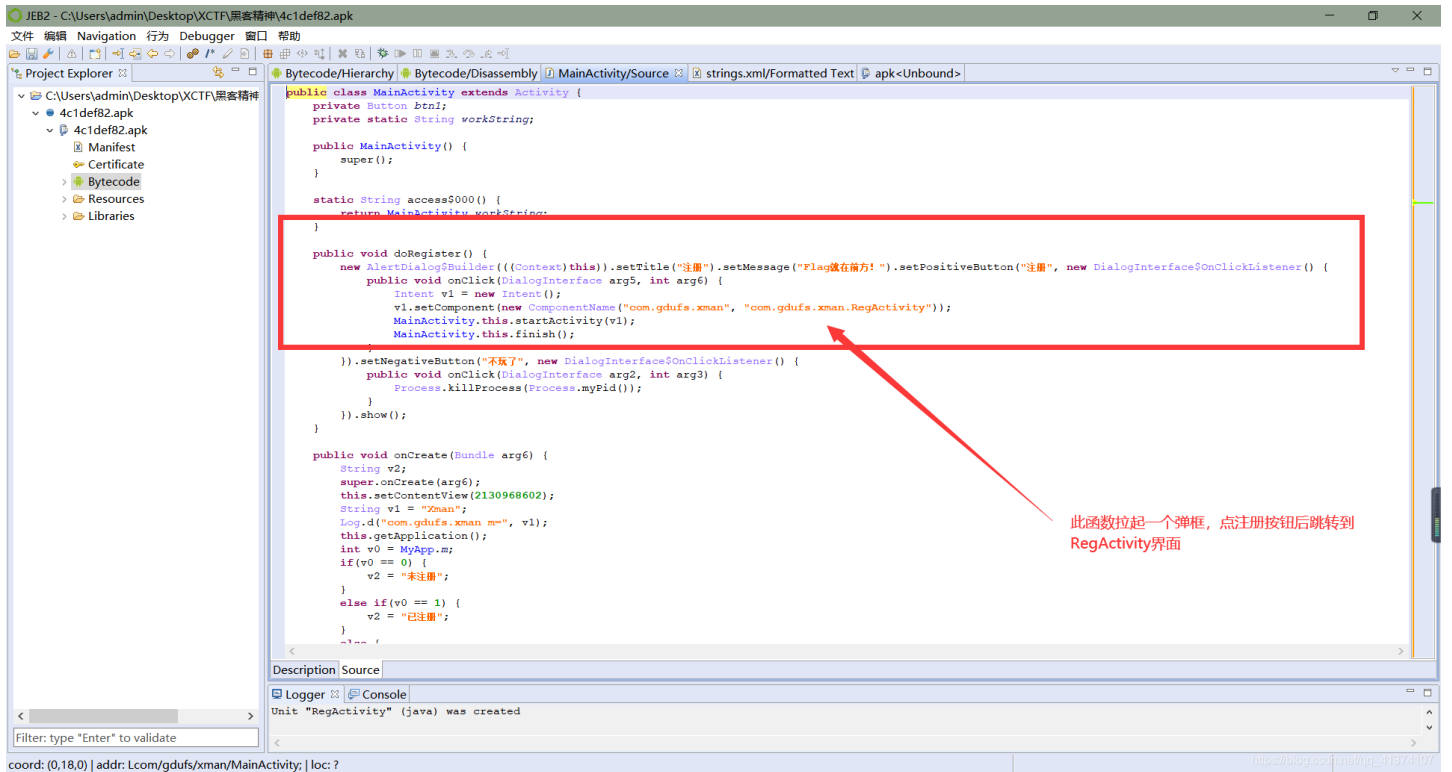


题目：黑客精神

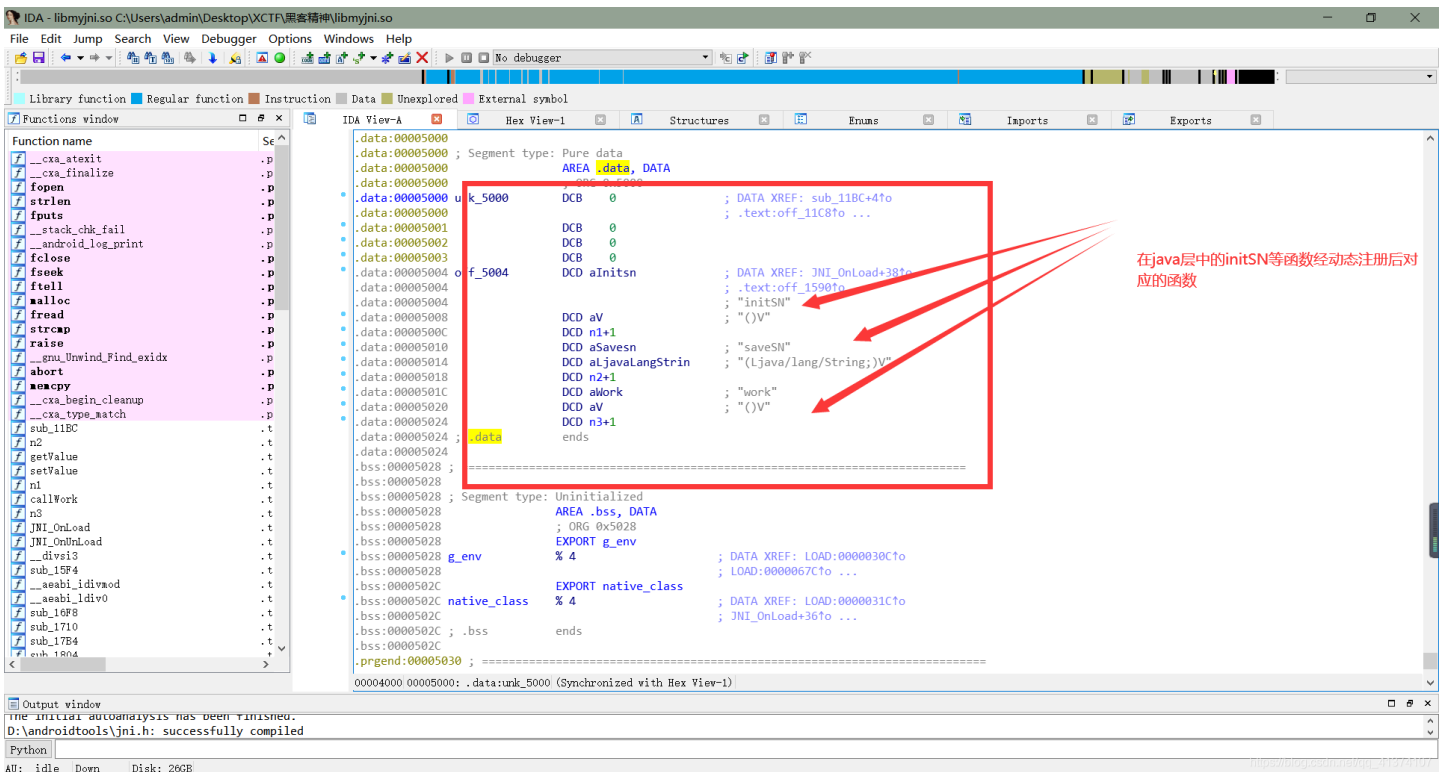
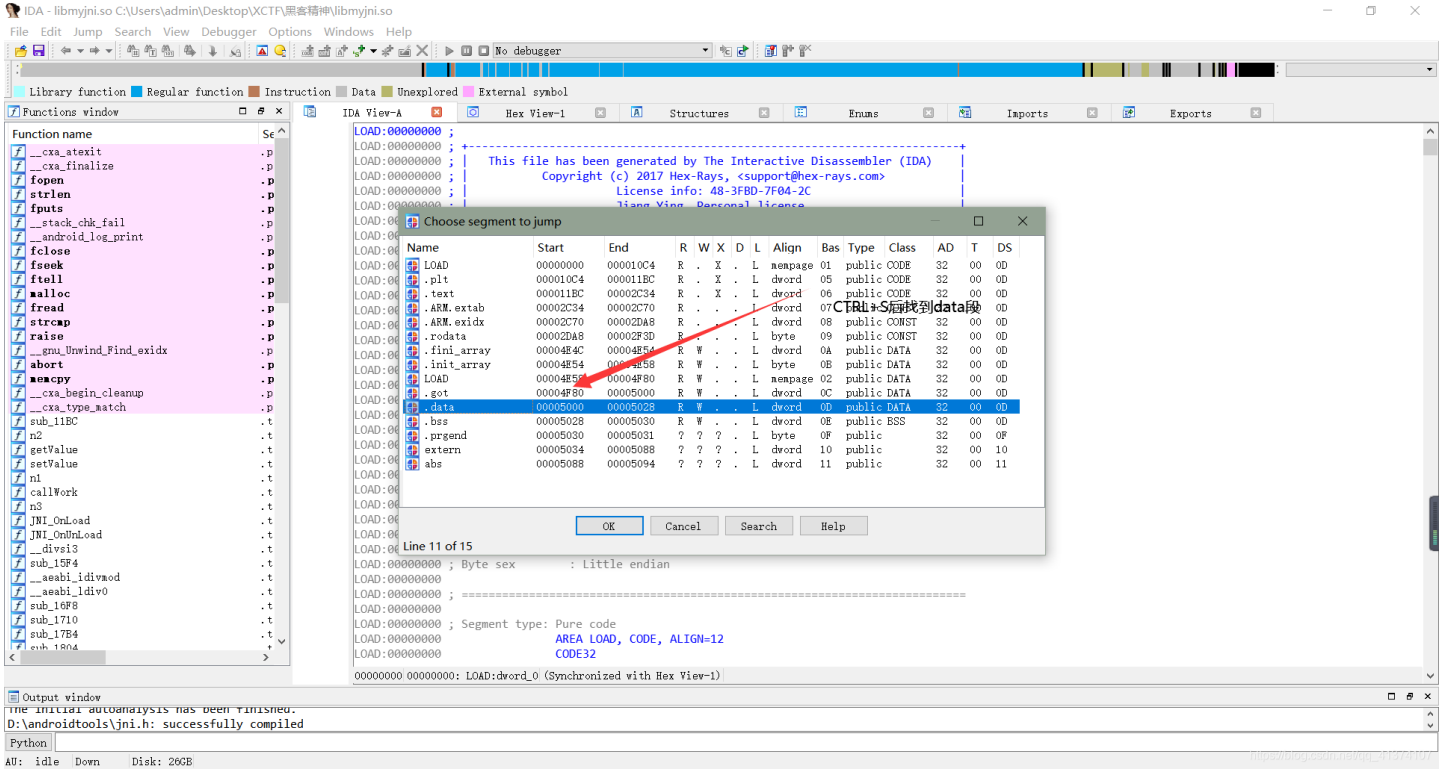
- 1、将下载好的题目拖进夜神模拟器中，发现要求注册，随便点击注册后，弹出一个弹框，提示 **已注册**。



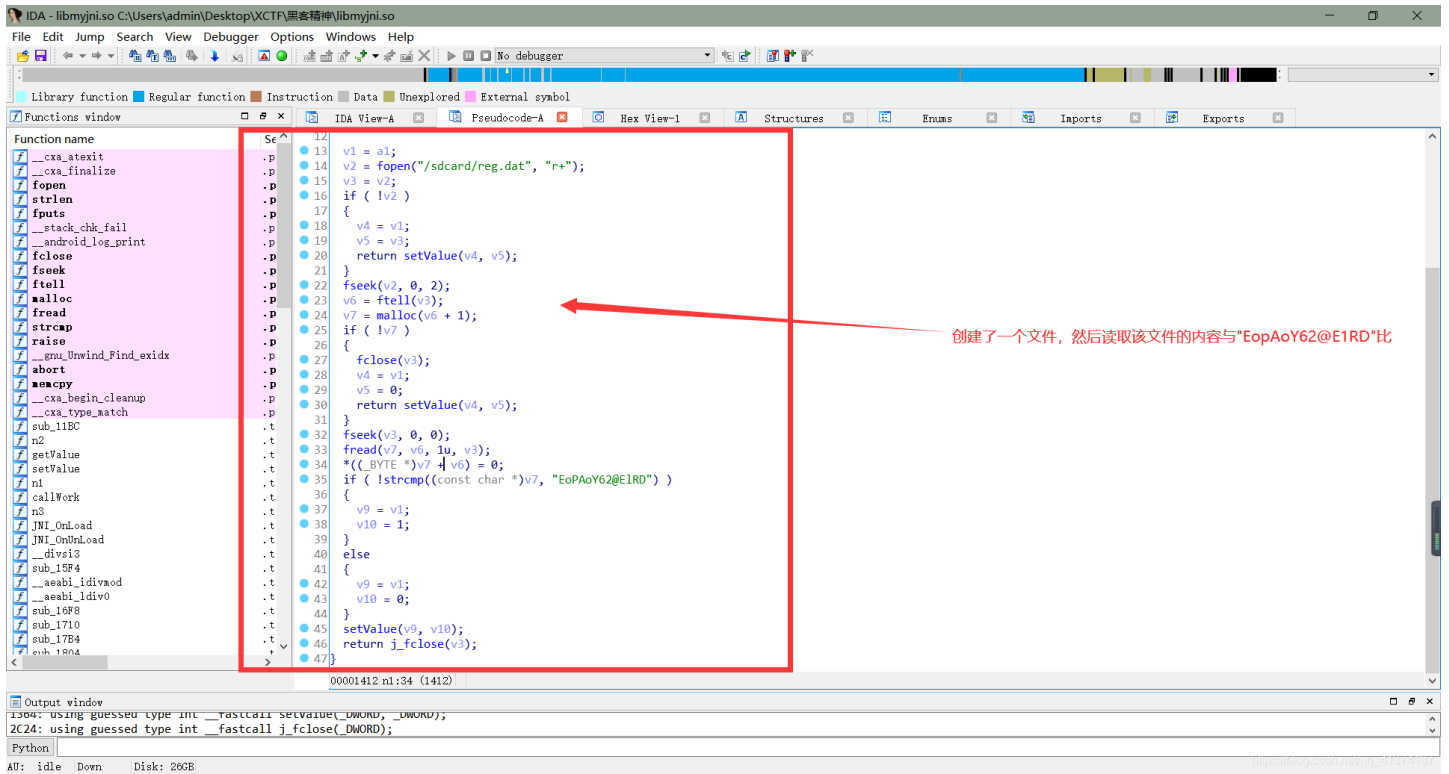
- 2、用JEB反编译后，跟进MainActivity文件，发现点击按钮后就一个弹出弹框，点击弹框确定后，跳转到RegActivity界面去，在该界面点击注册后，调用了so层函数 **saveSN**。



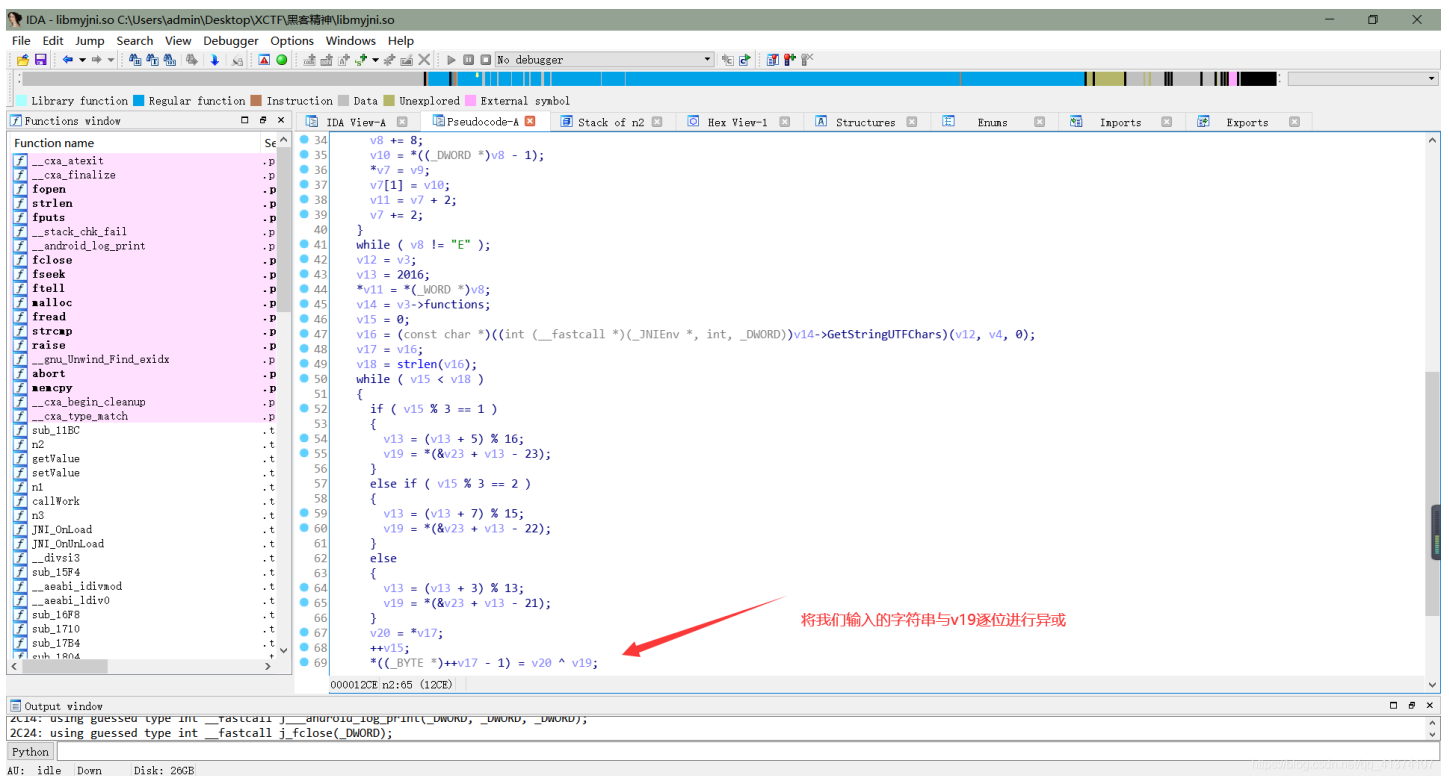
3、用IDA分析so文件发现在java层注册的native函数都是动态注册的，此时用 **Ctrl+S** 找到 **.data** 段进入，发现对应的函数 **n1**、**n2**、**n3**。



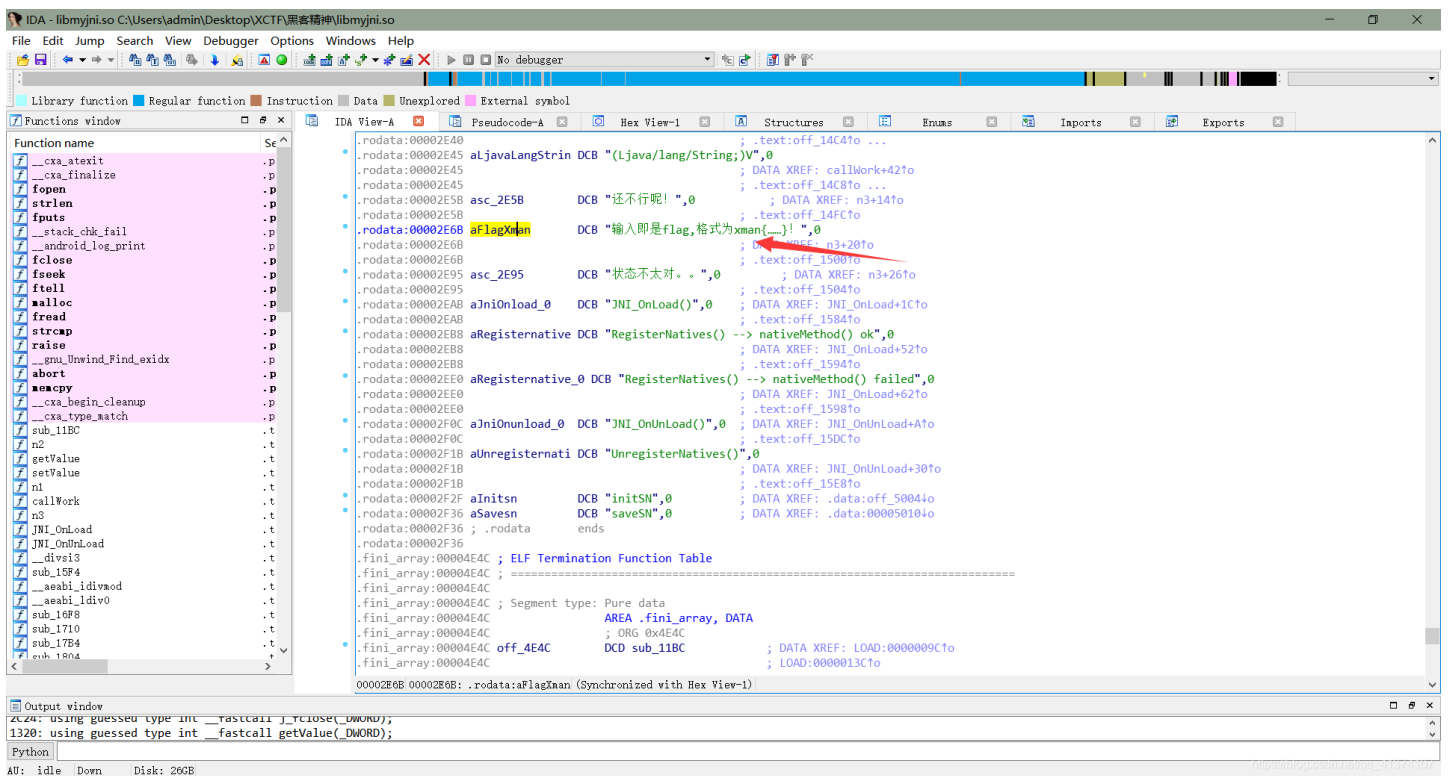
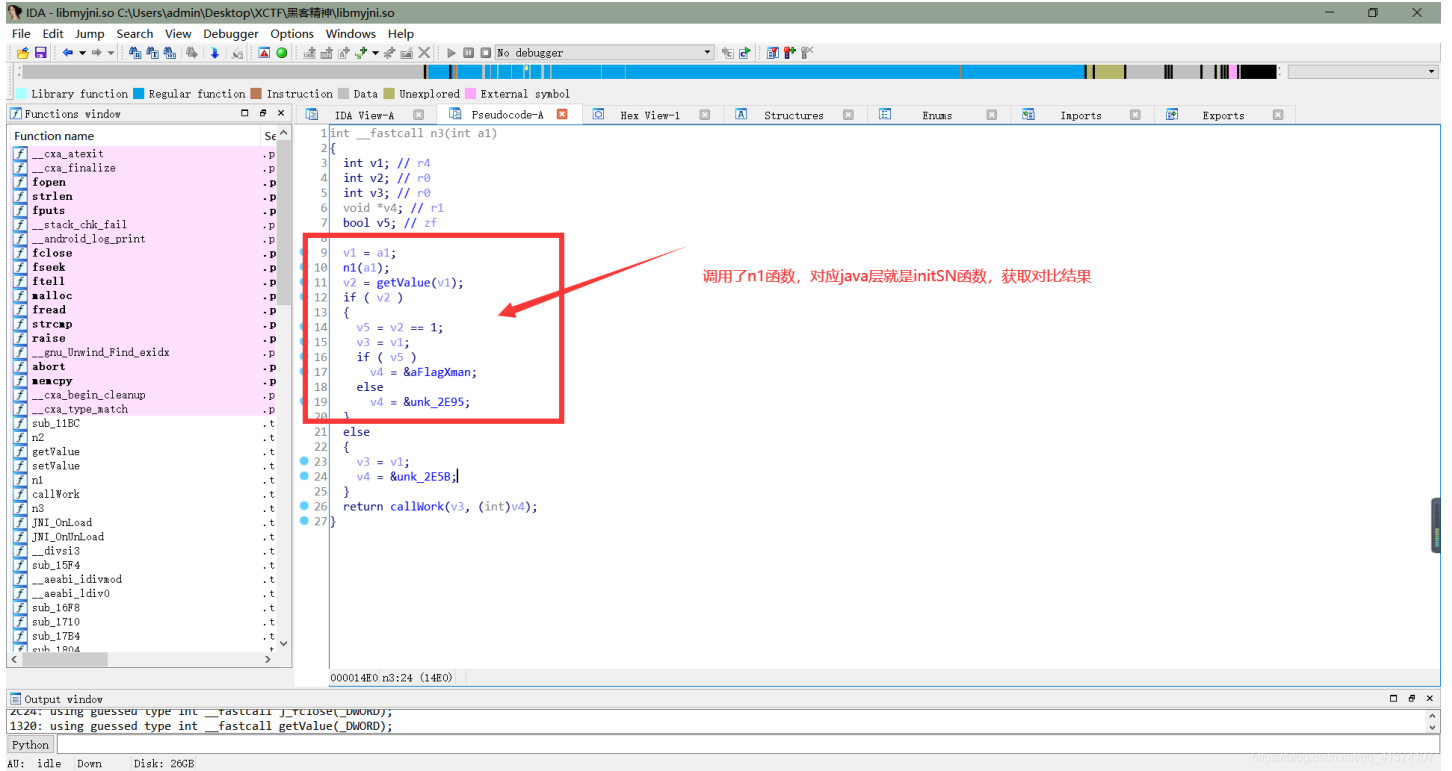
4、先分析一些 n1 函数，F5 大法后，可以很轻易看出该函数作用是创建了一个文件 /sdcard/reg.dat，然后读取该文件，与字符串 EoPAoY62@EIRD 进行比较；再来看一下 n2 函数，该函数对应这java层的 saveSN 函数，首先前面做了一大堆令人看不懂的操作，然后将我们输入的注册码的每个字符与另一个字符进行异或操作（该字符我们不知道，所以无法得到最终异或结果）；再来看一下 n3 函数，该函数首先调用了 n1 函数，若结果为真，则将一大串字符串赋值给 v4，双击去看一些该字符串，使用 A 键将其转为ascii后，发现提示 输入即是flag,格式为xman{.....}!。



创建了一个文件，然后读取该文件的内容与'EopAoY62@E1RD'比

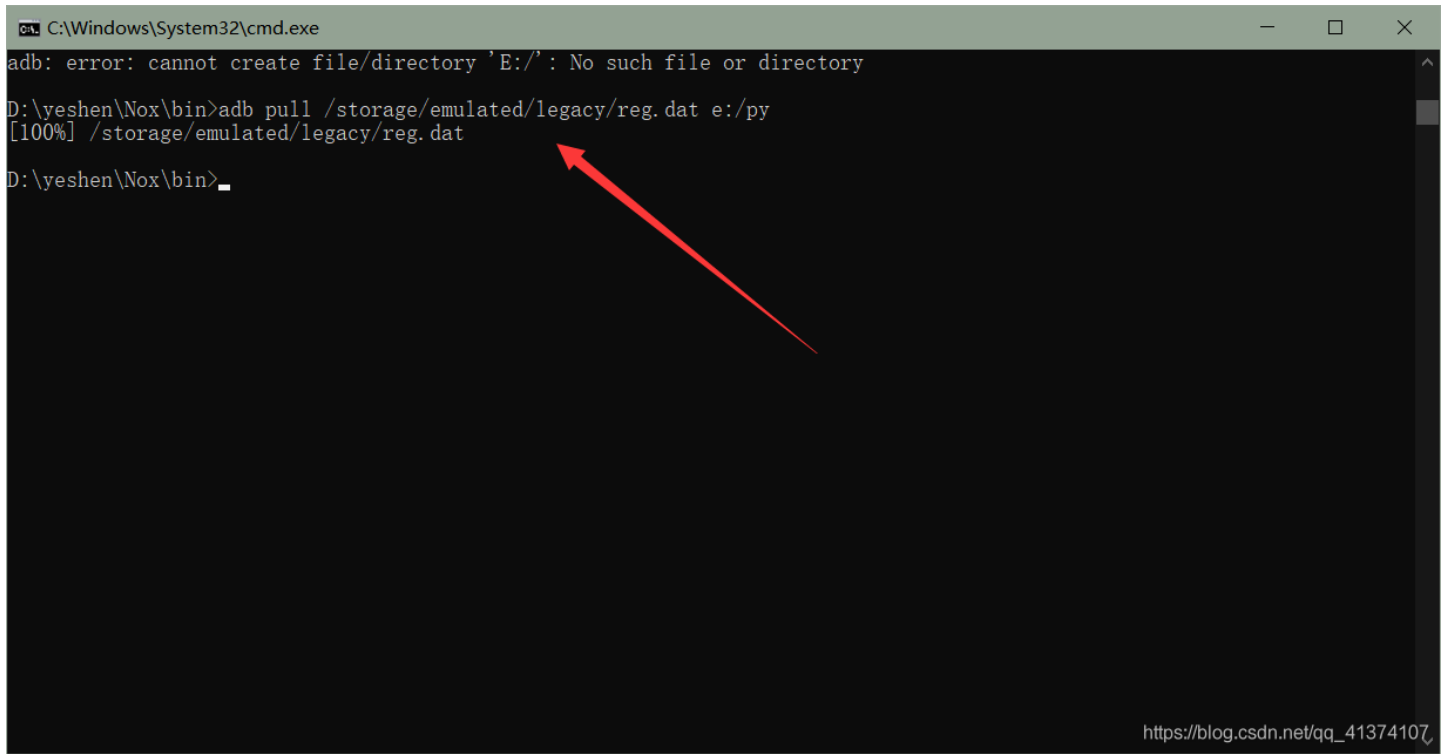


将我们输入的字符串与v19逐位进行异或

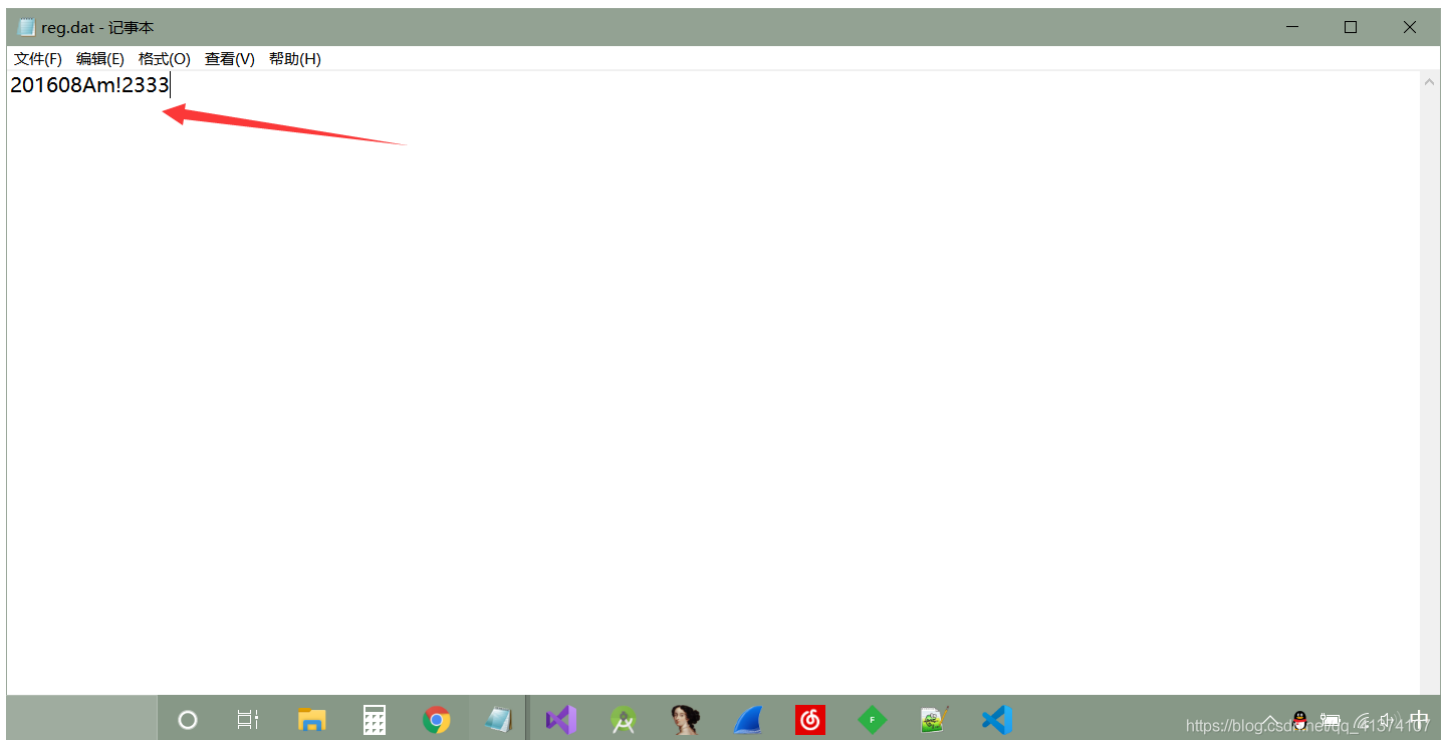


5、按照提示，我们将开始发现的字符串 **EoPAoY62@EIRD** (只发现这一个字符串，不输入这个输入啥)作为字符串进行输入，然后将文件 `/sdcard/reg.dat` 拷贝出来，打开文件一看，发现flag!!!


```
C:\Windows\System32\cmd.exe
adb: error: cannot create file/directory 'E:/' : No such file or directory
D:\yeshen\Nox\bin>adb pull /storage/emulated/legacy/reg.dat e:/py
[100%] /storage/emulated/legacy/reg.dat
D:\yeshen\Nox\bin>
```

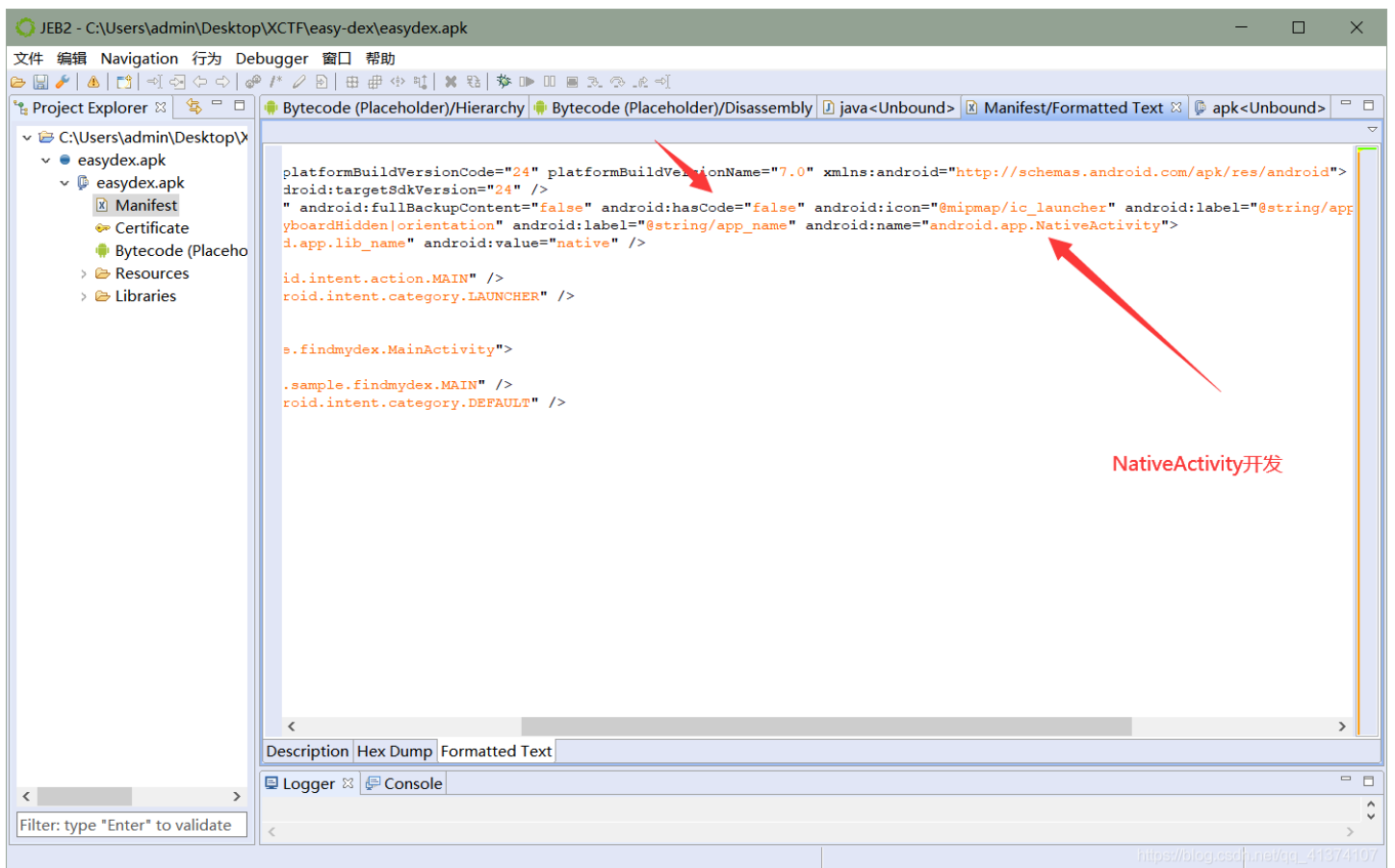
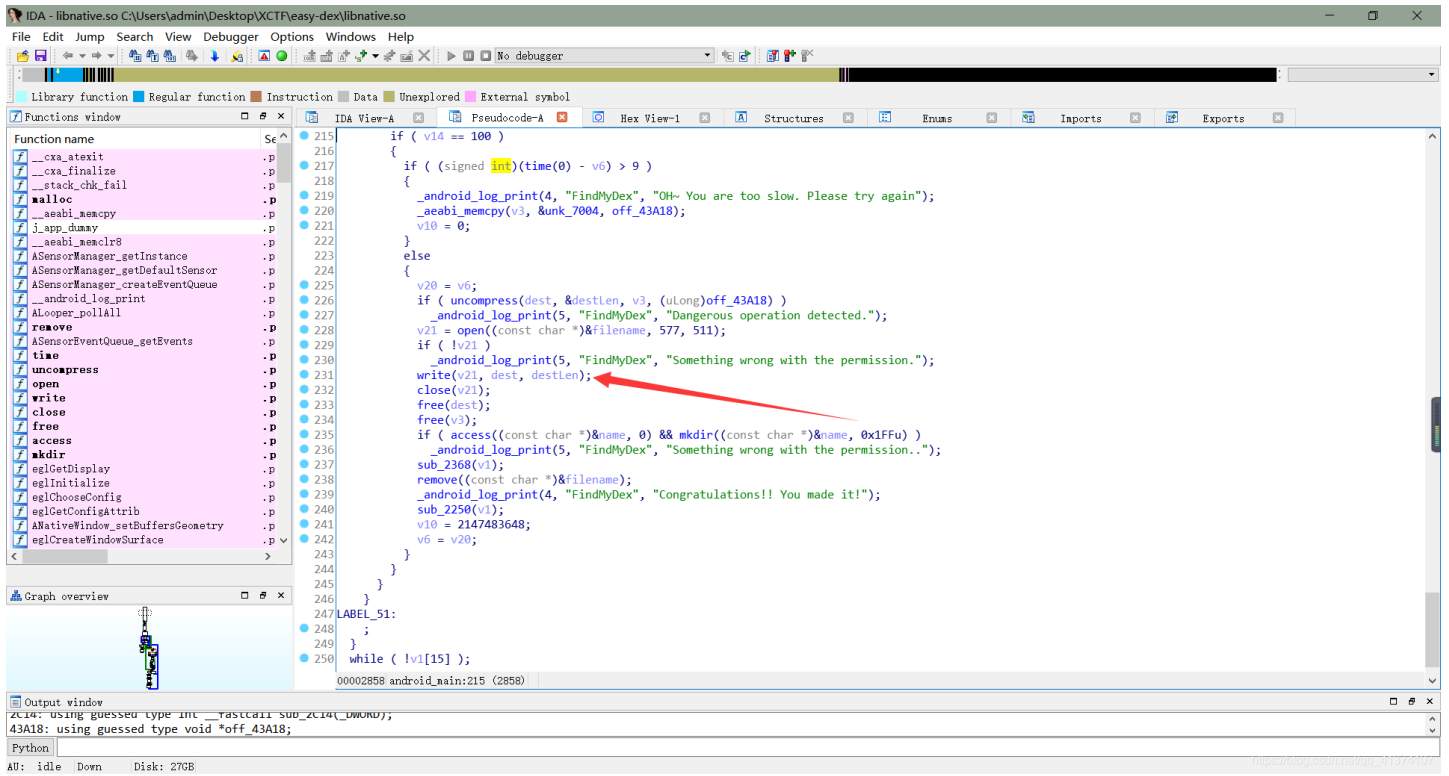


```
reg.dat - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
201608Am!2333
```

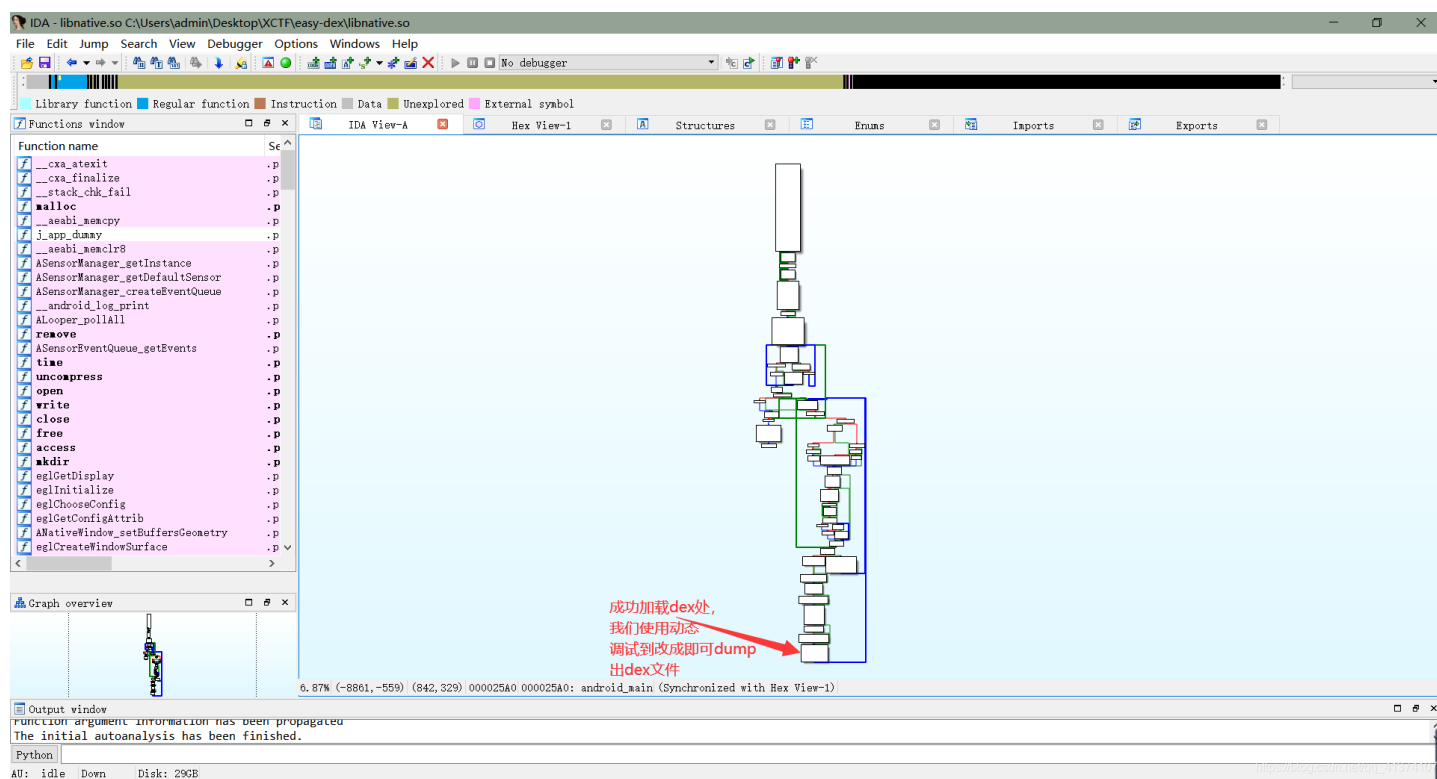


题目：easy-dex

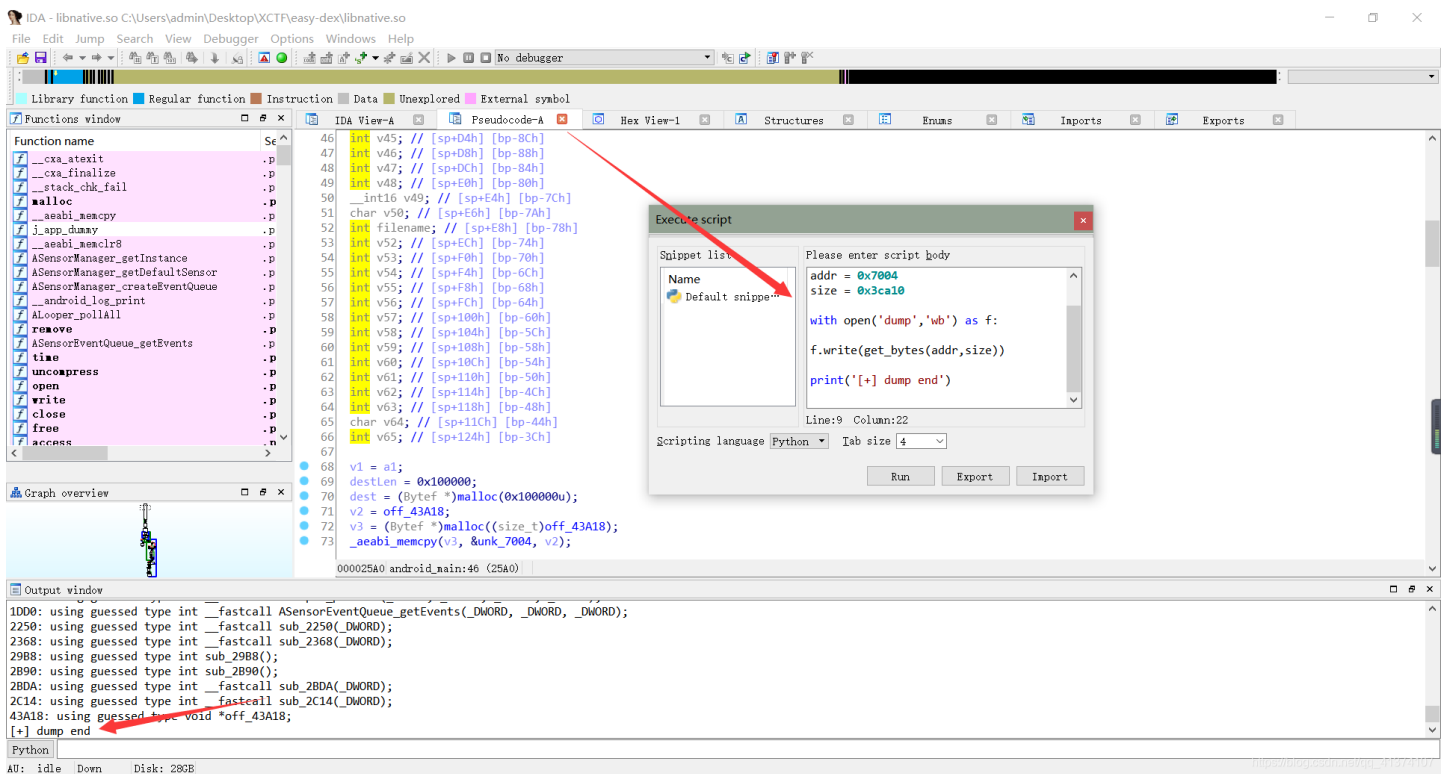
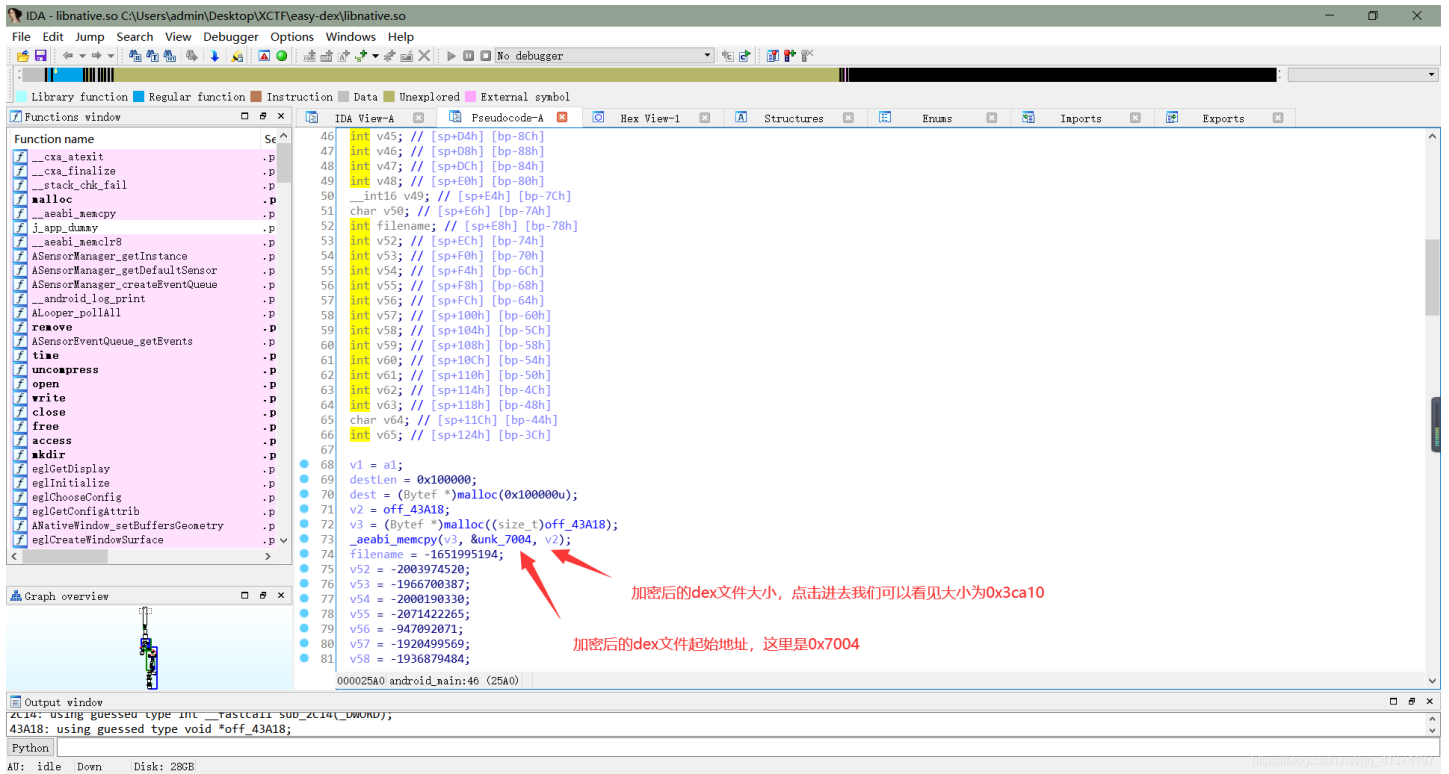
1、下载好题目后，拖进夜神中，发现直接是黑屏的，在JEB中反编译后，发现在 `AndroidManifest.xml` 文件 `application` 和 `activity` 标签中存在 `android:hasCode="false"` 和 `android:name="android.app.NativeActivity"`，说明这是个纯C++编写的，并且不含java代码，也就是 `Native Activity`。

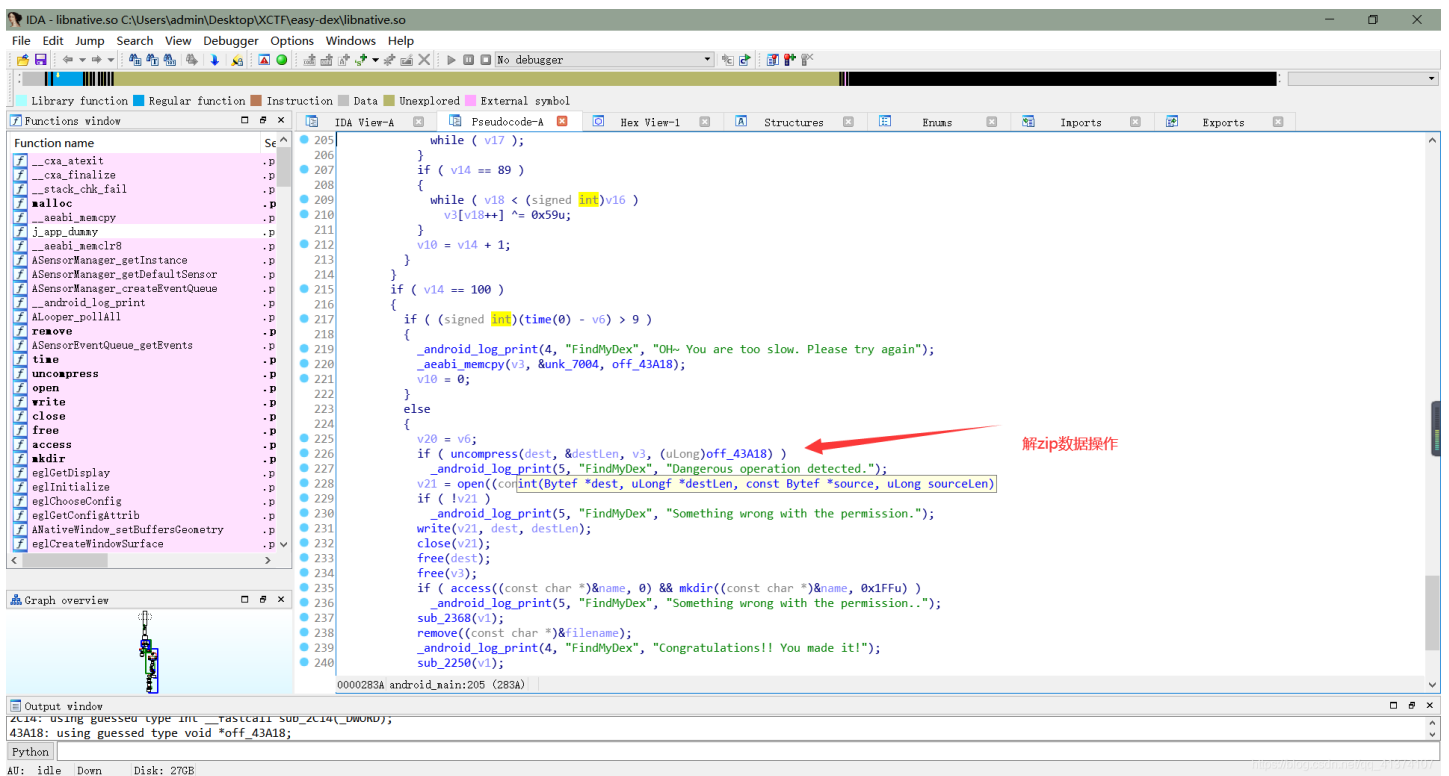
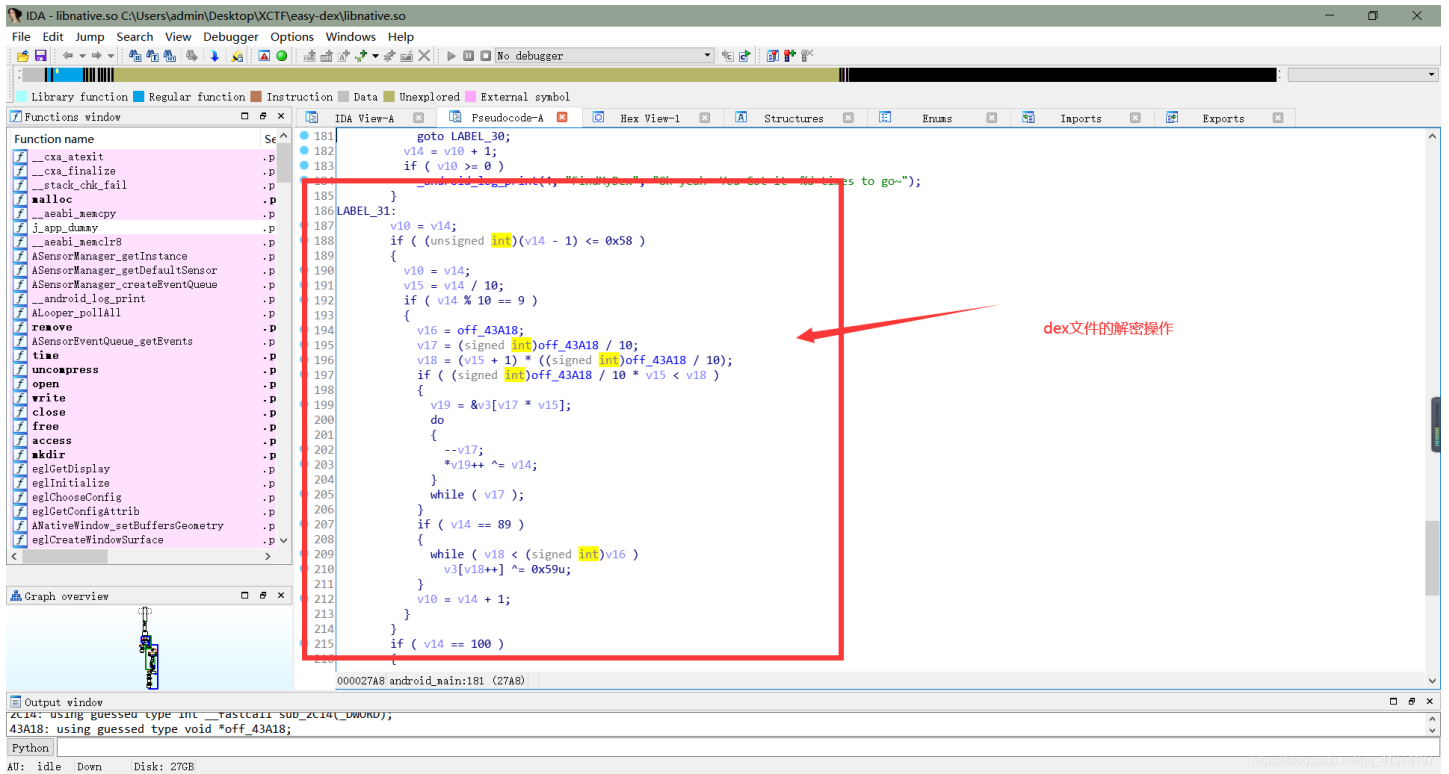


2、既然是 **Native Activity**，直接找到so文件，拖进IDA中，寻找 **android_main** 方法（这是 **Native Activity** 的入口方法，关于 **Native Activity** 的一些基础知识，我会在将一些我觉得写得比较好的博客链接附在文末），发现存在一个 **'write'** 方法，结合包名 **'findmydex'**，大胆推测一波此处就是将 **dex** 写入某个文件中，那么直接开启动态调试，在关键地方下好断点后，终于运行到了 **write** 函数处，结合传进去的参数，直接 **dump** 下来整个内存，用 **010** 打开一看，全都是 **'0000...'**，。。。。卒！！！！。。。。。



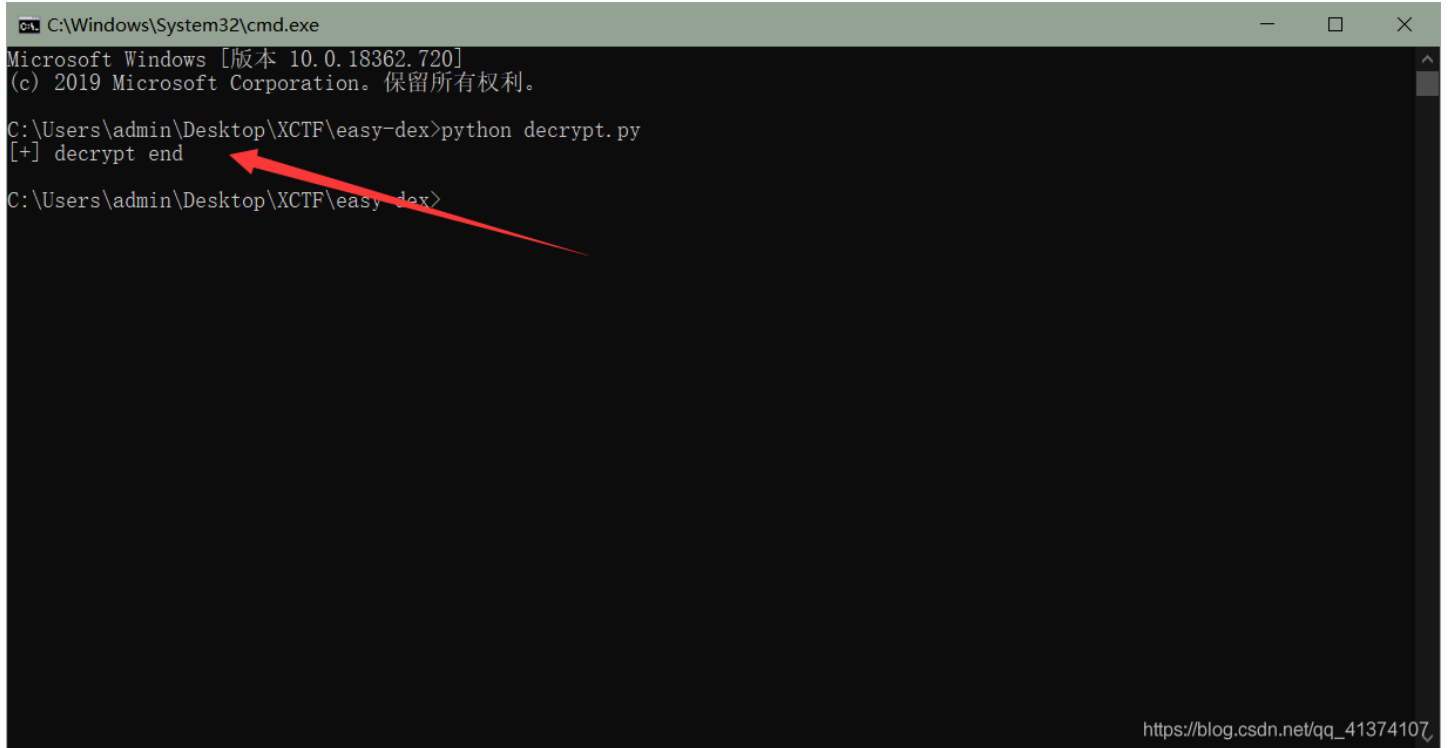
3、好吧，开玩笑的！！！动态调试 **dump** 下来的内存有问题，一看就不是 **dex** 文件，开始还以为调试的时候哪里出问题，导致 **dump** 出来的有问题，然后接着动态了一整个下午，发现好像 **dump** 下来的加密后的 **dex**，去看了一下大佬们的 **wp**，发现都是 **dump** 下加密后的 **dex**，然后直接解密，好吧，再来看 **android_main** 函数，发现一开始就通过 **_aeabi_memcpy** 函数将加密后的 **dex** 文件加载进来了，我们可以轻松看到加密后的 **dex** 文件首地址为 **0x7004** (ida使用 **F5** 后，要使用那一块内存空间地址直接是以 **&unk_** 地址命名的，所以首地址可以轻松看出来是 **0x7004**)，大小为 **0x3ca10**，那么直接在静态下执行 **dump** 脚本即可，至于解密，把想应的 **c** 语言转换为 **python** 即可，至于最后为啥要进行一次 **zip** 的解压操作，是因为在 **android_main** 函数中解密完成后调用了 **uncompress** 函数进行了解压缩（更偷懒的可以直接把 **F5** 后的 **c** 代码复制下来，替换一下就行了）。





```
C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.18362.720]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\admin\Desktop\XCTF\easy-dex>python decrypt.py
[+] decrypt end
C:\Users\admin\Desktop\XCTF\easy-dex>
```



IDA dump脚本:

```
import idaapi

addr = 0x7004
size = 0x3ca10

with open('dump','wb') as f:
    f.write(get_bytes(addr,size))

print('[+] dump end')
```

python解密dex脚本（我电脑上运行环境为3.6）:

```

import zlib

with open('dump','rb') as f:
    data1 = f.read()

    data = list(data1)

    count = 0

    while True:
        if count <= 0x59:
            count_tmp = (int)(count / 10)
            if count % 10 == 9:
                size = 0x3ca10
                size_tmp = (int)(size / 10)
                xor = (count_tmp + 1) * size_tmp
                if (size_tmp * count_tmp) < xor:
                    index = size_tmp * count_tmp
                    while size_tmp:
                        data[index] = data[index] ^ count
                        index = index + 1
                        size_tmp = size_tmp - 1
                if count == 89:
                    while xor < size:
                        data[xor] = data[xor] ^ 0x59
                        xor = xor + 1
            else:
                break
            count = count + 1

filebytes = bytes(data)
with open('easy-dex.dex','wb') as f1:
    f1.write(zlib.decompress(filebytes))
print('[+] decrypt end')

```

4、将解密后的dex文件拖进jeb中反编译，首先看一下 MainActivity.java 文件的 onCreate 函数，发现有一个按钮监听事件，触发后调用了 a.java 里面的 onClick 函数，那么去看一下 onclick 函数，发现调用了 MainActivity 里面的 a 函数，并且传入了两个字符串参数，第一个字符串是输入框中的值，第二个参数是 string.xml 资源文件中的字符串，然后与一个字节数组进行比较，结合题目反编译后的 public.xml 和 string.xml 文件，该字符串为 I have a male fish and a female fish.，并且资源ID为 two_fish，好吧，都已经是明示了，这里是 TwoFish 加密（关于TwoFish加密我会将一下我觉得写得还行的博客链接贴在最后面），并且该字符串就是密钥，与之比较字节数组就是加密后的结果。那么首先将字节数组转为字符串再说吧，一看字符串里面还有负数，那就直接与一下0xff咯（关于为啥要与0xff，是因为数字在java中是以补码形式表示的，与0xff相当于将一个有符号数转为了无符号数，看来我确实没有写博客的天赋，感觉说得不明不白的，老规矩，就把我觉得写得可以的博客链接贴在文末），然后与完后，直接转为ascii拼接成字符串，发现有些根本无法显示出具体的字符来。。。。又卡了，突然发现里面有个 / 符号，一下就想到了 base64，尝试一下base4解码，结果解出来又是啥都不是。。。。好气哦这个题。。。。那就在来一个base64加密吧，得到字符串 iE3y2hEF1izgbVUfGKNQrUCTgFQFop7iEkbmRwldwsZ1HdQGcPxRVAKwzV/eDC9N（好吧，我承认我有赌的成分。。。。），随便找了个在线解密的网站，解密即得到flag。

JEB2 - C:\Users\admin\Desktop\XCTF\easy-dex\easy-dex.dex

文件 编辑 Navigation 行为 Debugger 窗口 帮助

Project Explorer

- C:\Users\admin\Desktop\XCTF\easy-dex\easy-dex.dex

```
        v3.add(b.a(v1, 0, v4));
        v1 = new byte[16];
    }

    ByteBuffer v2_1 = ByteBuffer.allocate(v3.size() * 16);
    Object[] v3_1 = v3.toArray();
    int v4_1 = v3_1.length;
    int v1_1;
    for(v1_1 = 0; v1_1 < v4_1; ++v1_1) {
        v2_1.put(v3_1[v1_1]);
    }

    v0_1 = v2_1.array();
}
catch(Exception v0) {
    v0_1 = new byte[1];
}

return v0_1;
}

static byte[] i() {
    return MainActivity.m;
}

protected void onCreate(Bundle arg4) {
    super.onCreate(arg4);
    this.setContentView(2130968602);
    this.findViewById(2131427413).setOnClickListener(new a(this, this.findViewById(2131427412), ((Context)this)));
}
}
```

coord: (0,10,0) | addr: Lcom/a/sample/findmydex/MainActivity; | loc: ?

https://blog.csdn.net/qq_41374107

JEB2 - C:\Users\admin\Desktop\XCTF\easy-dex\easy-dex.dex

文件 编辑 Navigation 行为 Debugger 窗口 帮助

Project Explorer

- C:\Users\admin\Desktop\XCTF\easy-dex\easy-dex.dex

```
package com.a.sample.findmydex;

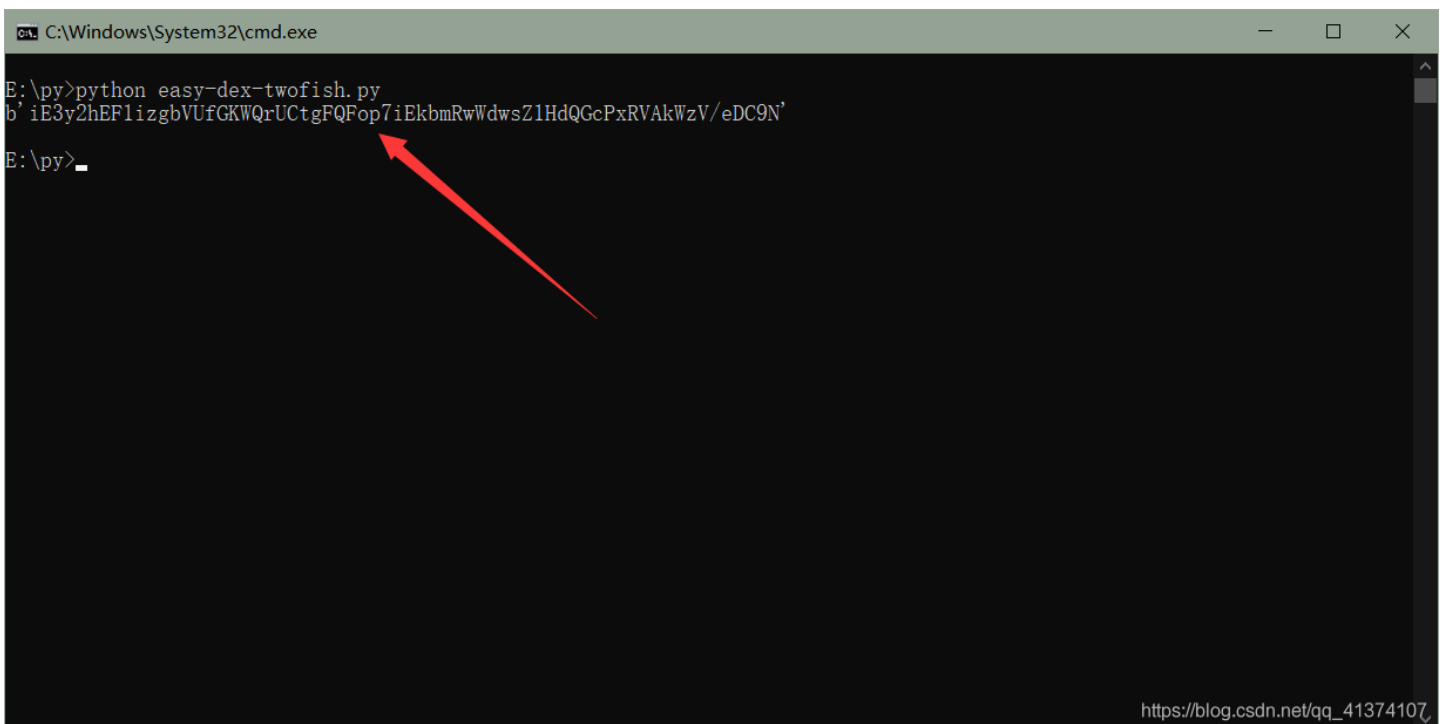
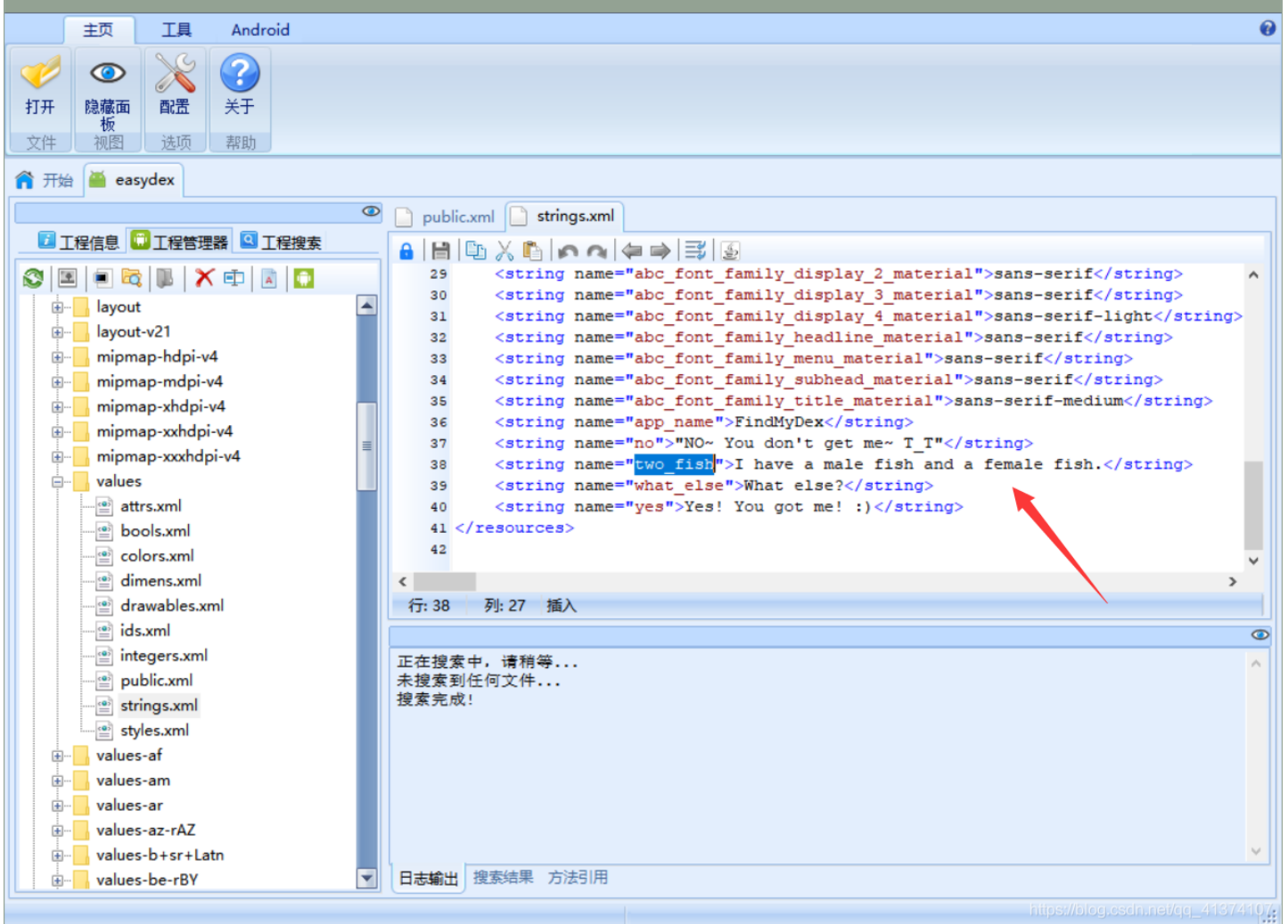
import android.content.Context;
import android.view.View.OnClickListener;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import java.util.Arrays;

class a implements View.OnClickListener {
    a(MainActivity arg1, EditText arg2, Context arg3) {
        this.o = arg1;
        this.a = arg2;
        this.b = arg3;
        super();
    }

    public void onClick(View arg5) {
        if(Arrays.equals(MainActivity.a(this.a.getText().toString(), this.o.getString(2131099683)), MainActivity.i())) {
            Toast.makeText(this.b, this.o.getString(2131099685), 1).show();
        }
        else {
            Toast.makeText(this.b, this.o.getString(2131099682), 1).show();
        }
    }
}
```

coord: (0,9,0) | addr: Lcom/a/sample/findmydex/a; | loc: ?

https://blog.csdn.net/qq_41374107



关于Native Activity的:

https://blog.csdn.net/qq_19683651/article/details/82623717

https://blog.csdn.net/qq_21071977/article/details/77878252

关于TwoFish加密的:

<https://blog.csdn.net/l540538550/article/details/5642435>

关于与0xff的:

https://blog.csdn.net/csdn_ds/article/details/79106006?depth_1-utm_source=distribute.pc_relevant.none-task&utm_source=distribute.pc_relevant.none-task