2021年11月逆向练习

原创

抒情诗、 于 2021-11-18 01:06:21 发布 1073 收藏 4

分类专栏： CTF 文章标签： 逆向工程 CTF Reverse

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/zhangxiansheng12/article/details/121389891

版权

CTF 专栏收录该内容

24 篇文章 0 订阅

订阅专栏

全文目录

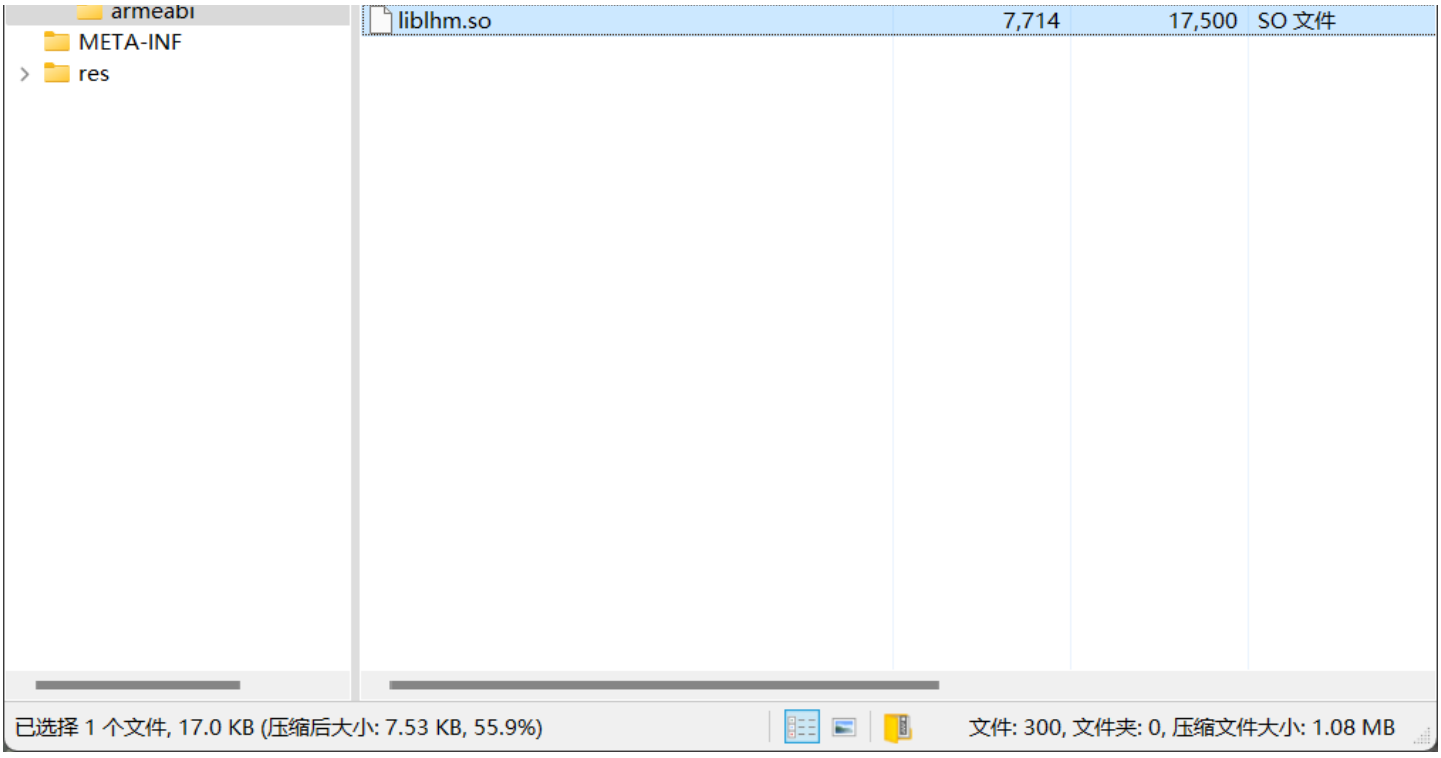
前言

最近每天打游戏，现在脑子里都快出现幻觉了，决定学习一段时间冷静一下脑子，下次打游戏务必冷静，每天打游戏不能超过60分钟！今天到月末，我一有空就会刷一会儿逆向题目来学习一下，每天至少要做一道逆向题目，也就是说到本月末要做至少13道题目，目标不大，现就这样吧。

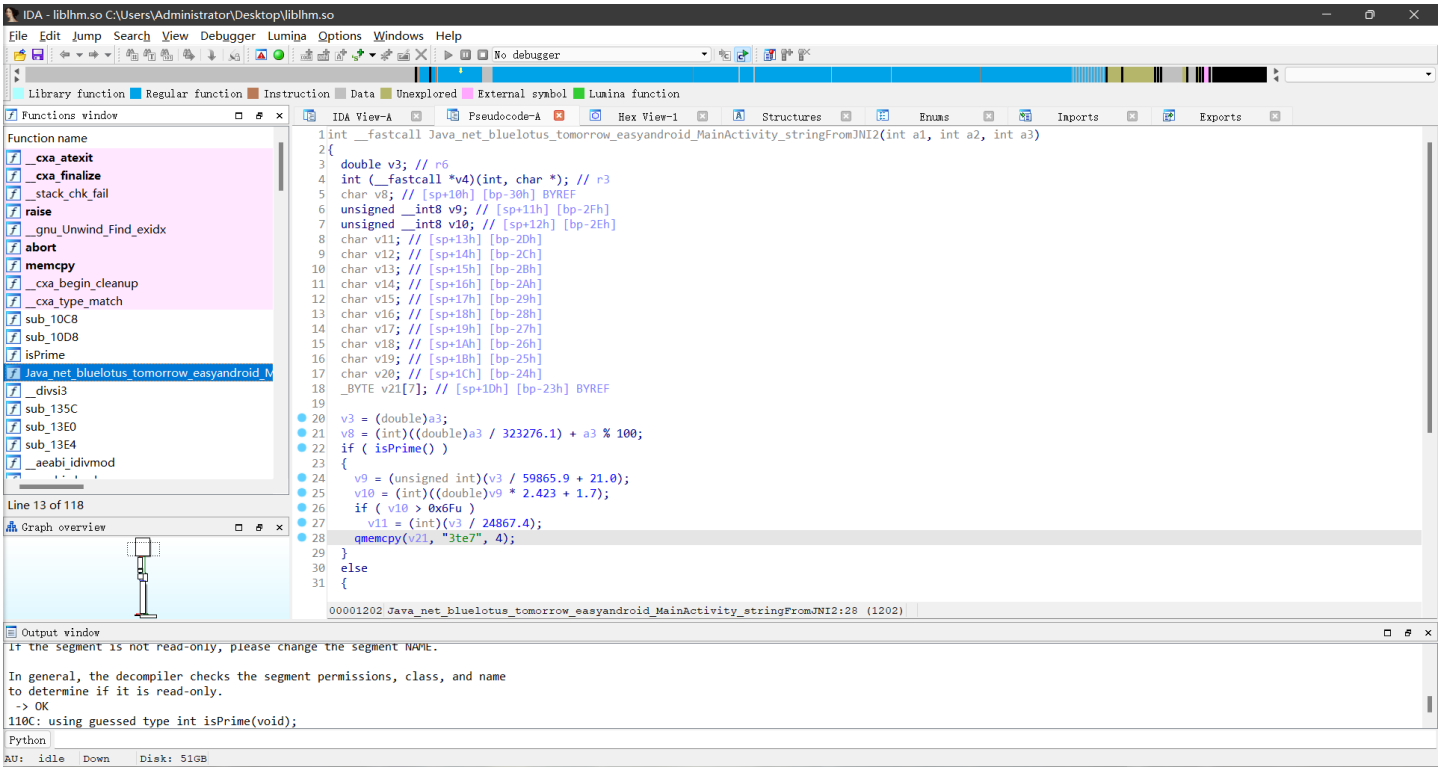
一、BugKu-Timer(阿里CTF)

1. 简单分析

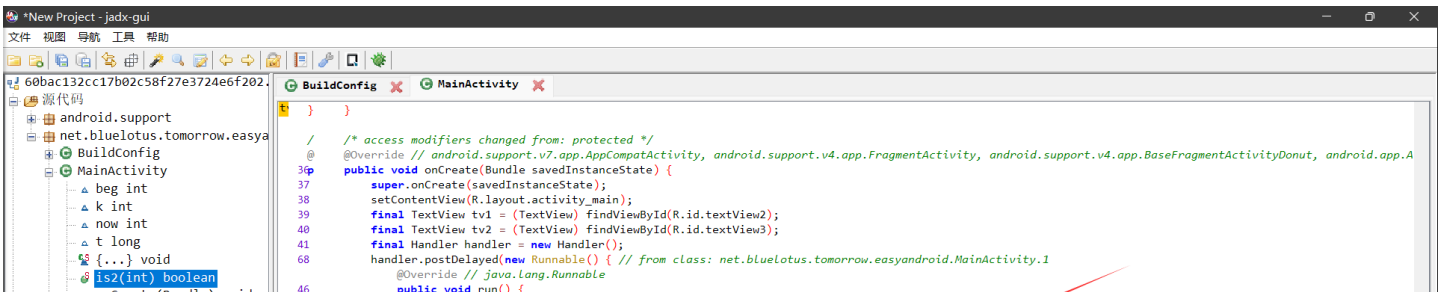
安装之后发现是一个倒计时，倒计时的初始数值非常大哈。那再拿到`jadx-gui`里面康康，发现内部加载了so文件，将apk文件以压缩包的形式打开之后拉出来一个so文件逆一下。

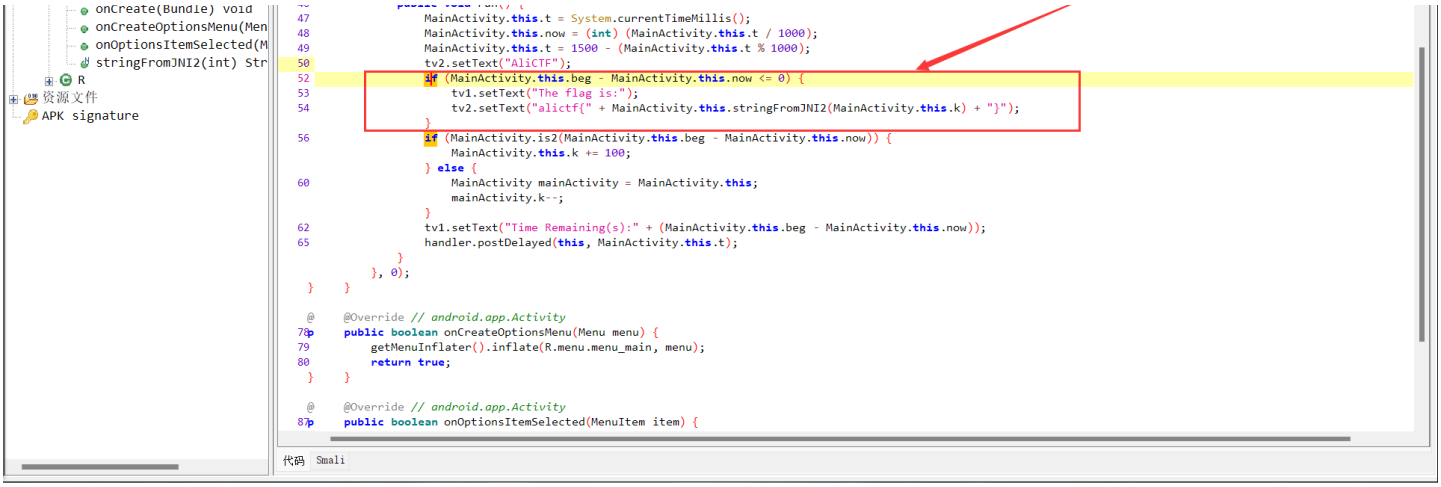


ida32位打开so文件，找到关键函数



看晕了，so文件一如既往的复杂，不想分析，直接先查查wp看看，他们好像都是修改安卓程序重新打包做的，那就再打开`jadx-gui`看一下关键的安卓代码。





先看看框里面的条件都是什么，首先是`beg`，就是倒计时的那个数值；然后now就是现在的时间(秒)。我们要做的就是修改k的值，就是模仿一下安卓程序的运行，看看满足上面框中的那个条件的时候，这个k的值为多少，然后因为stringFromJNI调用的是so文件里面的东西，而so文件不方便进行逆向，可以直接修改安卓程序的数值进而满足条件，输出flag。

下面写个python脚本简单算一下k的值应该为多少：

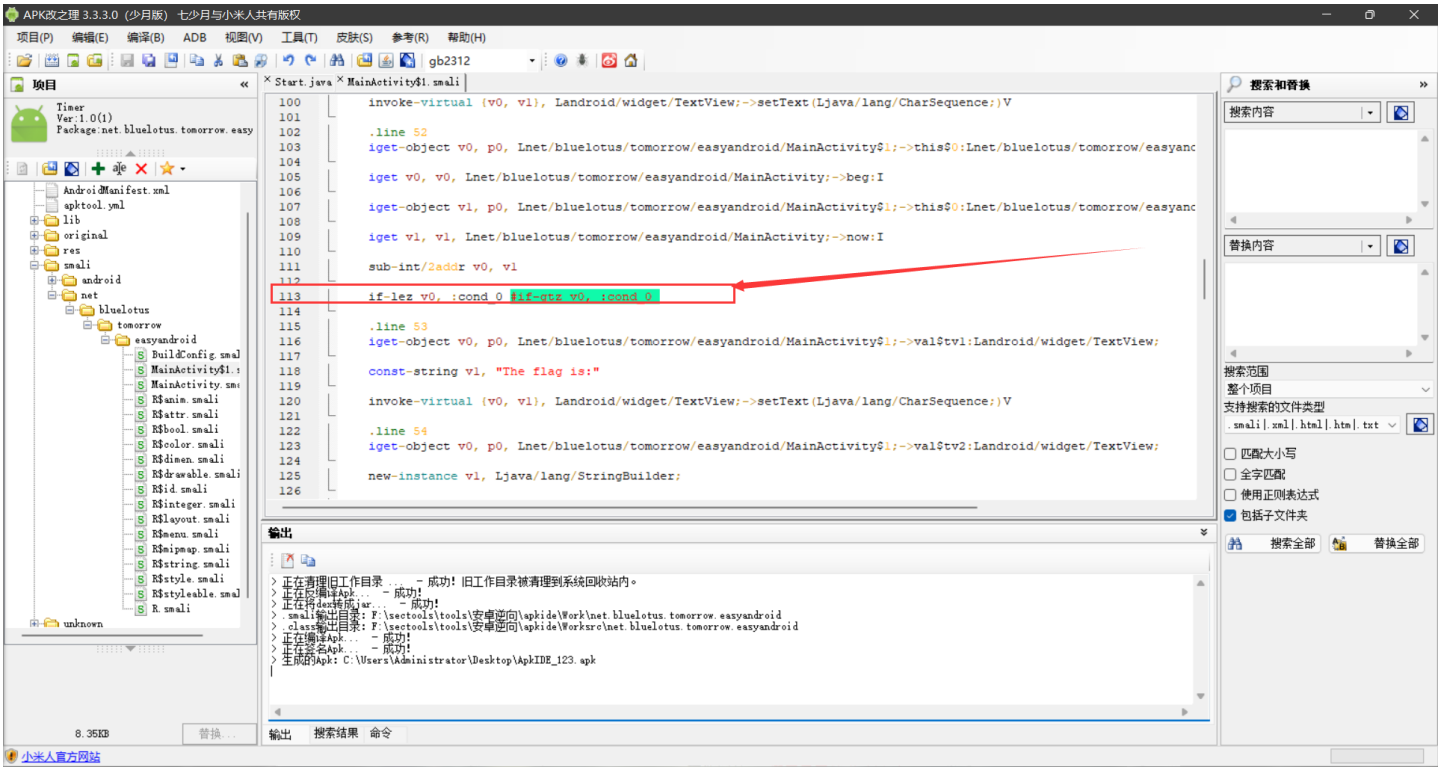
```python
def is2(n):
    if n <= 3:
        return n > 1
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while True:
        if i * i > n:
            break
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True


time = 200000
k = 0
while True:
    if time <= 0:
        break
    if is2(time):
        k += 100
    else:
        k -= 1
    time -= 1
print(k)
# 1616384
```

2. 修改安卓程序数值

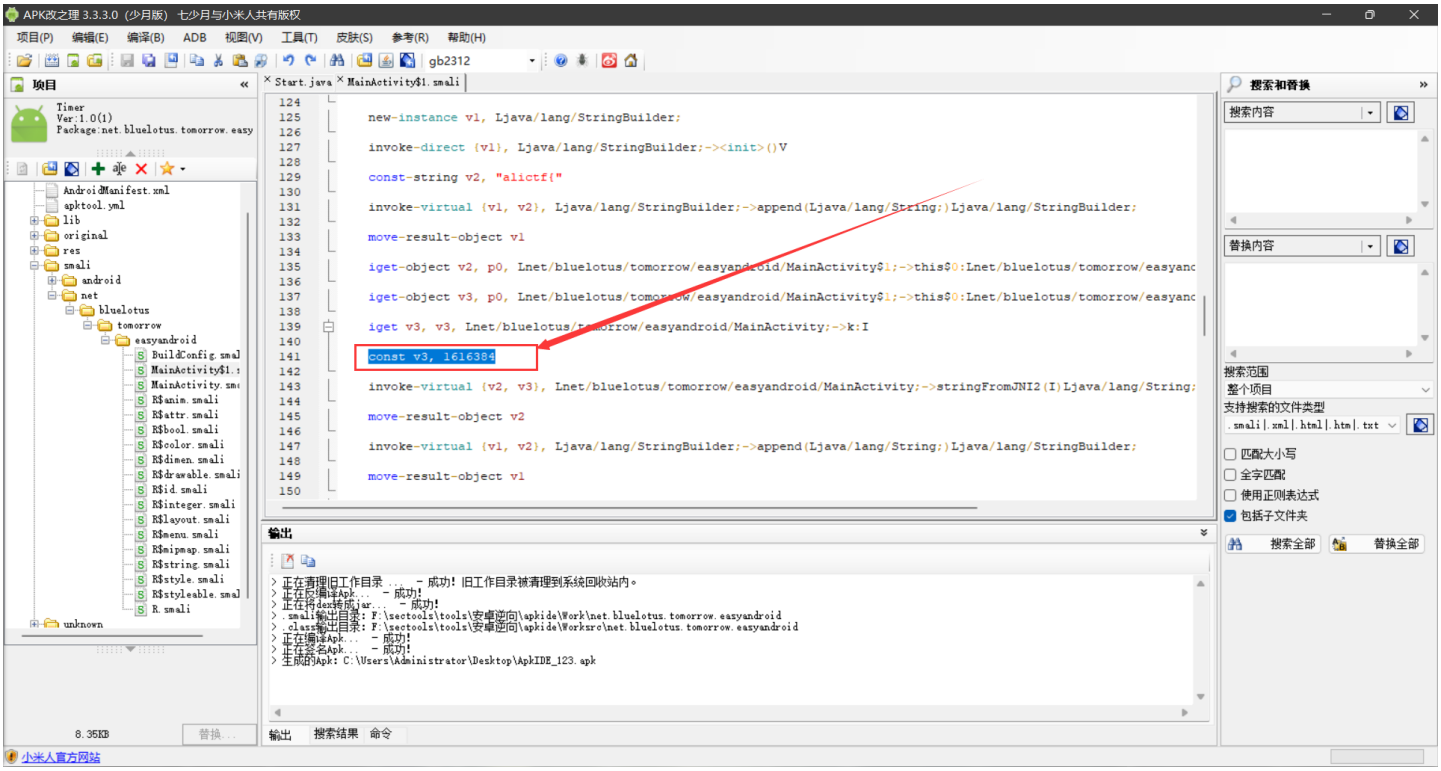
首先使用工具先把apk给反编译一下apktools之类的工具即可，然后打开`MainActivity.smali`文件，修改k的初始值，如下图，k的初始值为`0x0`，要修改为上面我们得到的数值`1616384`

做安卓的反编译时，我们应该只修改`MainActivity$1.smali`文件中的内容，需要改一下下面的框圈住的内容。



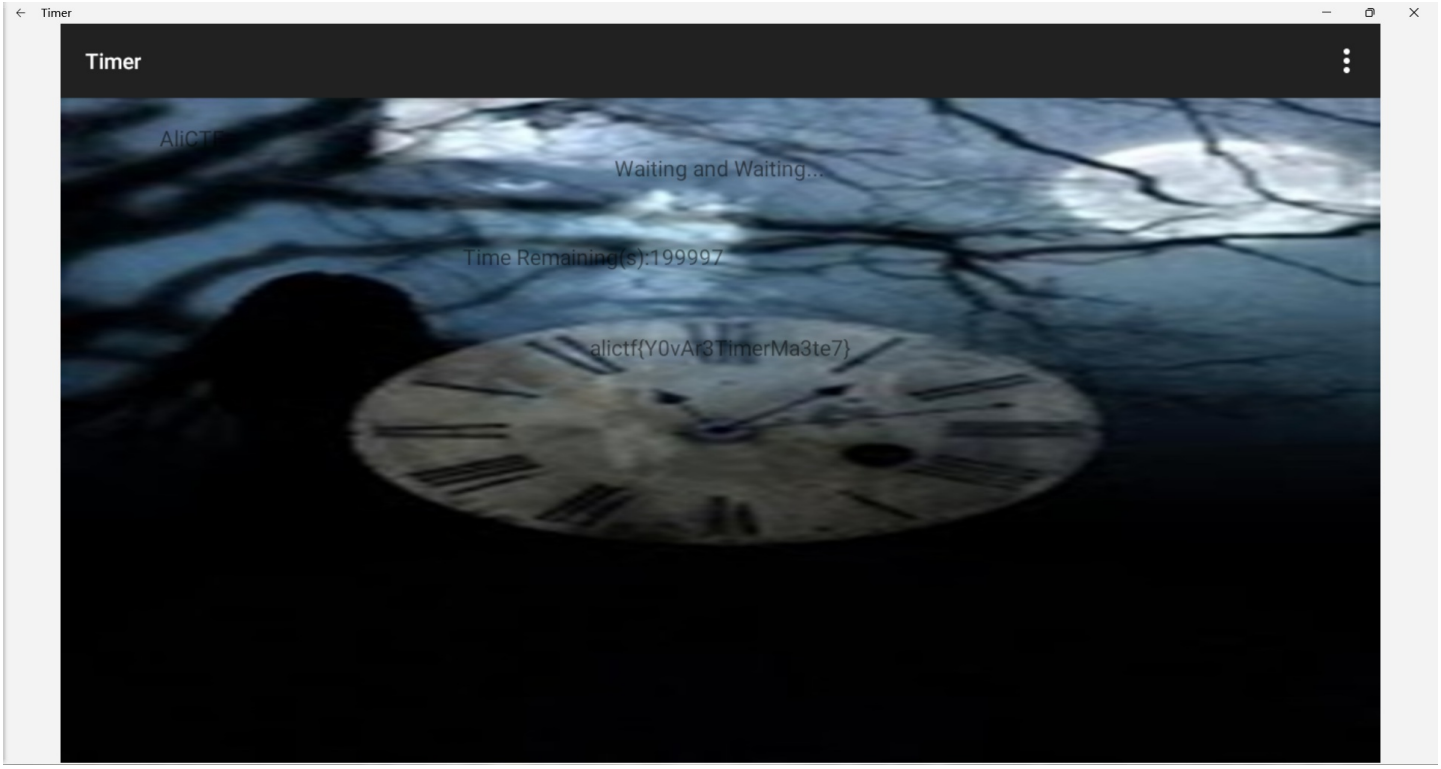
就是把那个倒计时的判断条件改了，原来大于现在改成小于了，这个条件程序一开始运行的时候就会满足。

然后就是修改k的值了，也是需要在这个`MainActivity$1.smali`文件中做修改，添加下面红圈中的内容即可`const v3, 1616384`：



此时再编译发现可以编译成功，直接安装执行即可获得flag。

```
flag{Y0vAr3TimerMa3te7}
```



3. 总结

程序没加壳破起来真舒服，这里用到的软件工具是ApkIDE(APK改之理)，老工具了，但是好用就完事了。

刚开始我不小心改到了`MainActivity.smali`这个文件，发现怎么整都不对，后来才发现在那个文件里面修改容易出现错误(或者是必然出现错误？)，总之只在`MainActivity$1.smali`文件中改就对了，只要smali语法没问题应该就是能编译通过的。

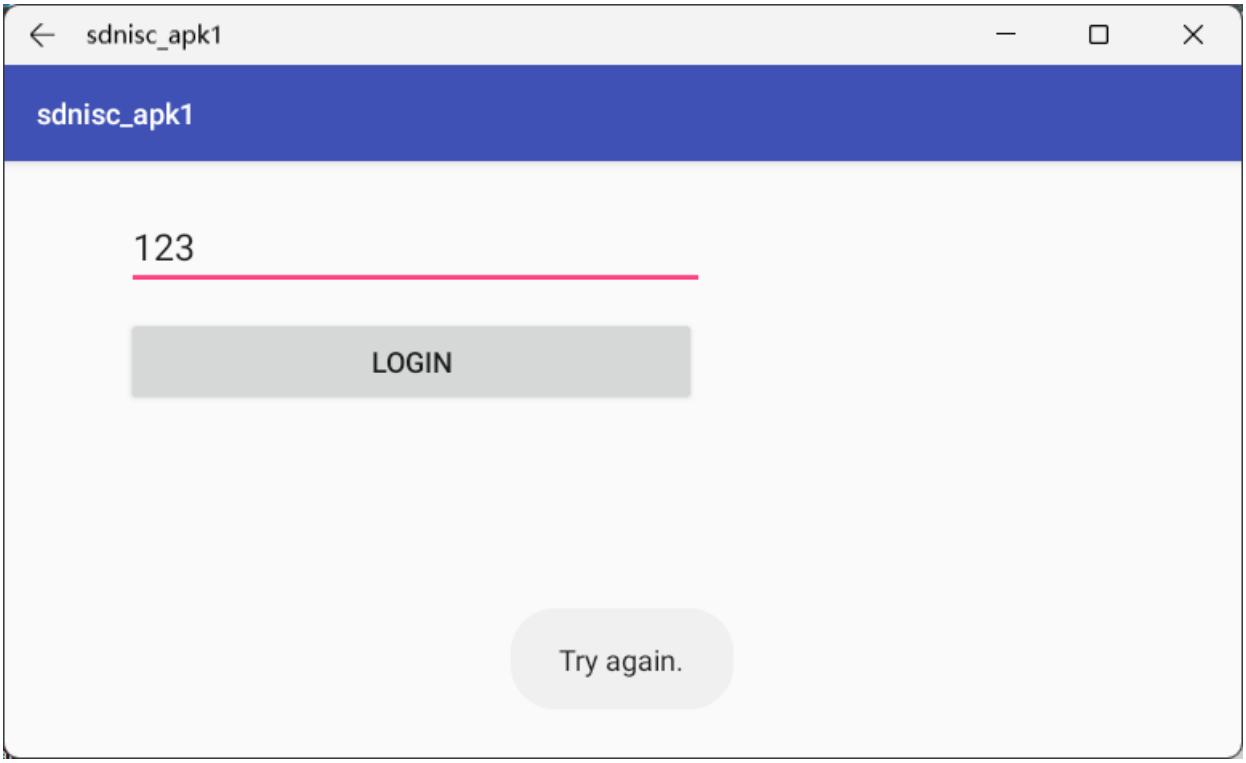
二、BugKu-signin

```
提　　示：君远至此，辛苦至甚。 窃谓欲状，亦合依例，并赐此题。
描　　述：来源：第七届山东省大学生网络安全技能大赛
```

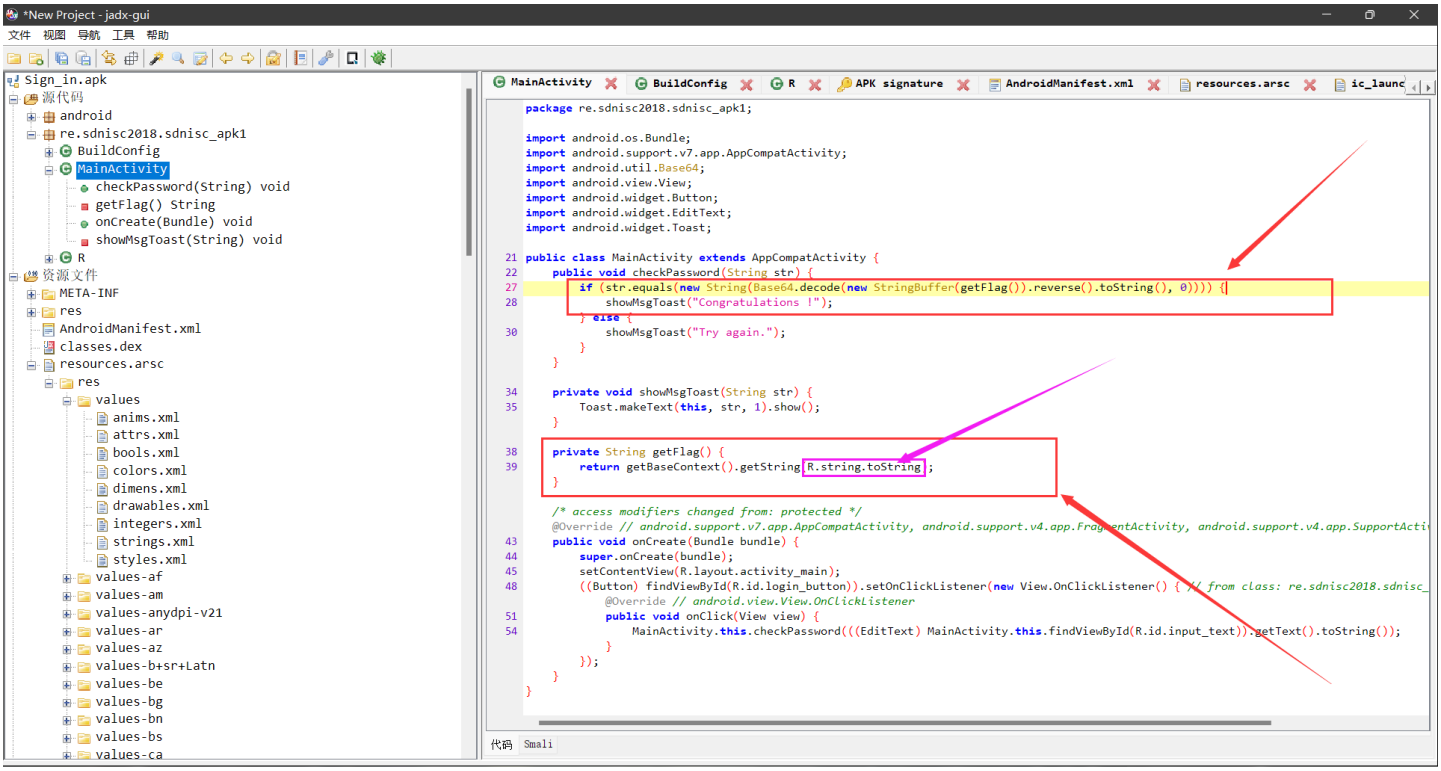
这题应该在上面那道题前面做的，但是昨天没看到这道题，今天特此补做一下。

1.功能分析

安装打开之后是下面的这个界面



看起来非常普通，那就用jadx先反编译一下。注意到下面的判断逻辑：



他这个是获得tostring这个加密后的flag之后进行了一个字符串的倒置，再进行了一个base64的解码，非常简单，现在我们只需要找到这个toString的字符串资源即可。

2.找到toString，获取flag

这个toString的字符串资源在下面这里可以看到

`resources.arsc > res > values` 文件夹下面找到 `string.xml` 就是本程序的字符串资源了

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    ......
    <string name="app_name">sdnisc_apk1</string>
    <string name="search_menu_title">Search</string>
    <string name="status_bar_notification_info_overflow">999+</string>
    <string name="toString">991YiZWOz81ZhFjZfJXdwk3X1k2XzIXZIt3ZhxmZ</string>
</resources>
```

根据解密原理，写出解密脚本如下：

```
import base64
tostring = "991YiZWOz81ZhFjZfJXdwk3X1k2XzIXZIt3ZhxmZ"[::-1]

print(base64.b64decode(tostring))
# b'flag{Her3_i5_y0ur_f1ag_39fbc_}'
```

3.总结

这题很基础，逻辑也很简单，就是如果你是安卓逆向的初学者的话，这个安卓的字符串资源你可能会不好找到在哪里，找到之后这就是个签到题，啥都不是。

三、BugKu-逆向入门

1. 功能分析

首先打开软件看看，竟然不能打开，应该是位数不匹配吧可能。那就先拿exeinfope查看一下文件的pe。发现并不是正常的exe文件，然后使用vscode看看发现是`data:image/png;base64`格式的图片文件，直接在线(使用浏览器打开)，发现一个二维码，如下所示：

2. 这是个misc?

关于为什么直接扫码之后就获得flag了，这是一个非常让人尴尬的问题，毫无re知识，不如放在misc里面，(::doge:□

flag

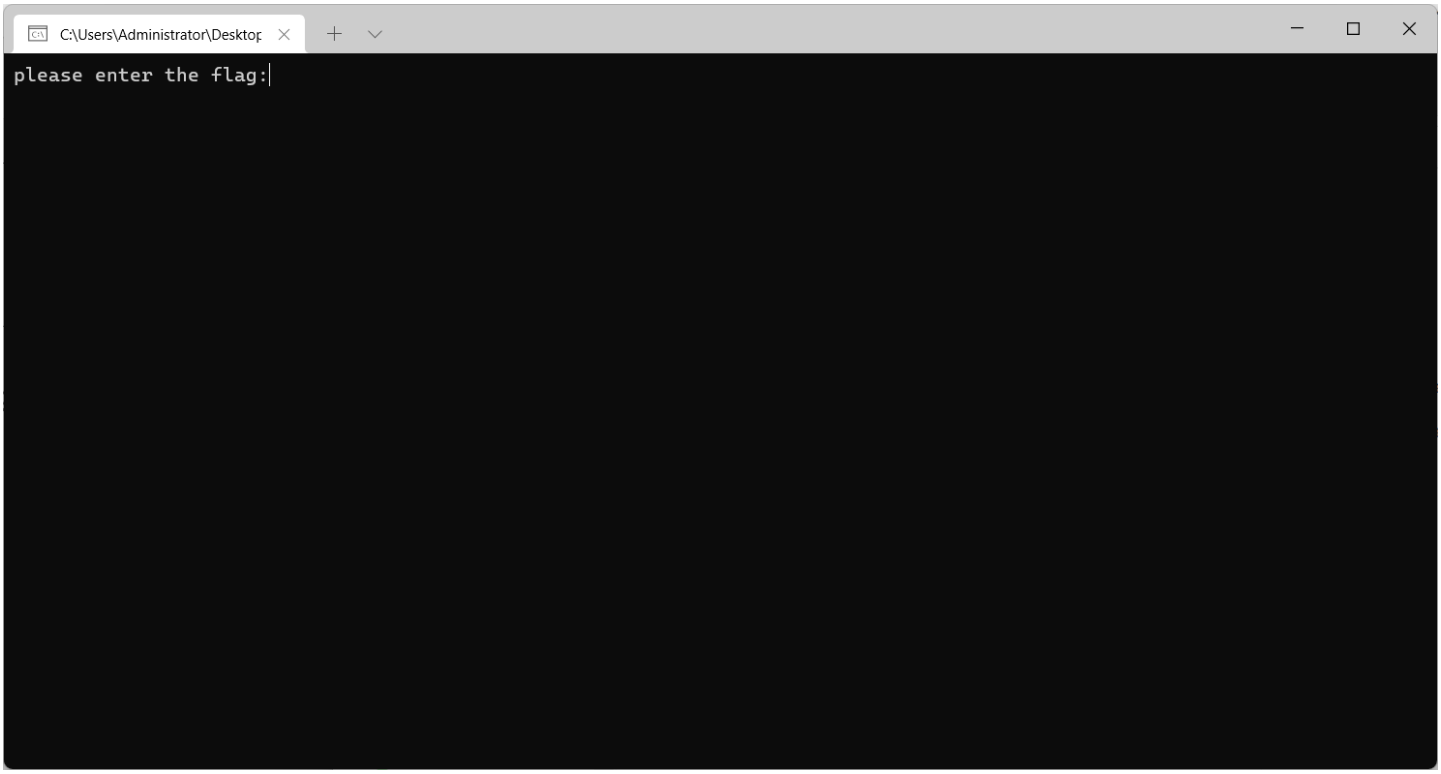
```
bugku{inde_9882ihsd8-0}
```

3. 总结

这题没什么好总结的，题目怎么说呢，总感觉我被当成低能儿了来着...

四、BugKu-love

运行之后是这个界面，一看就是老签到题了...



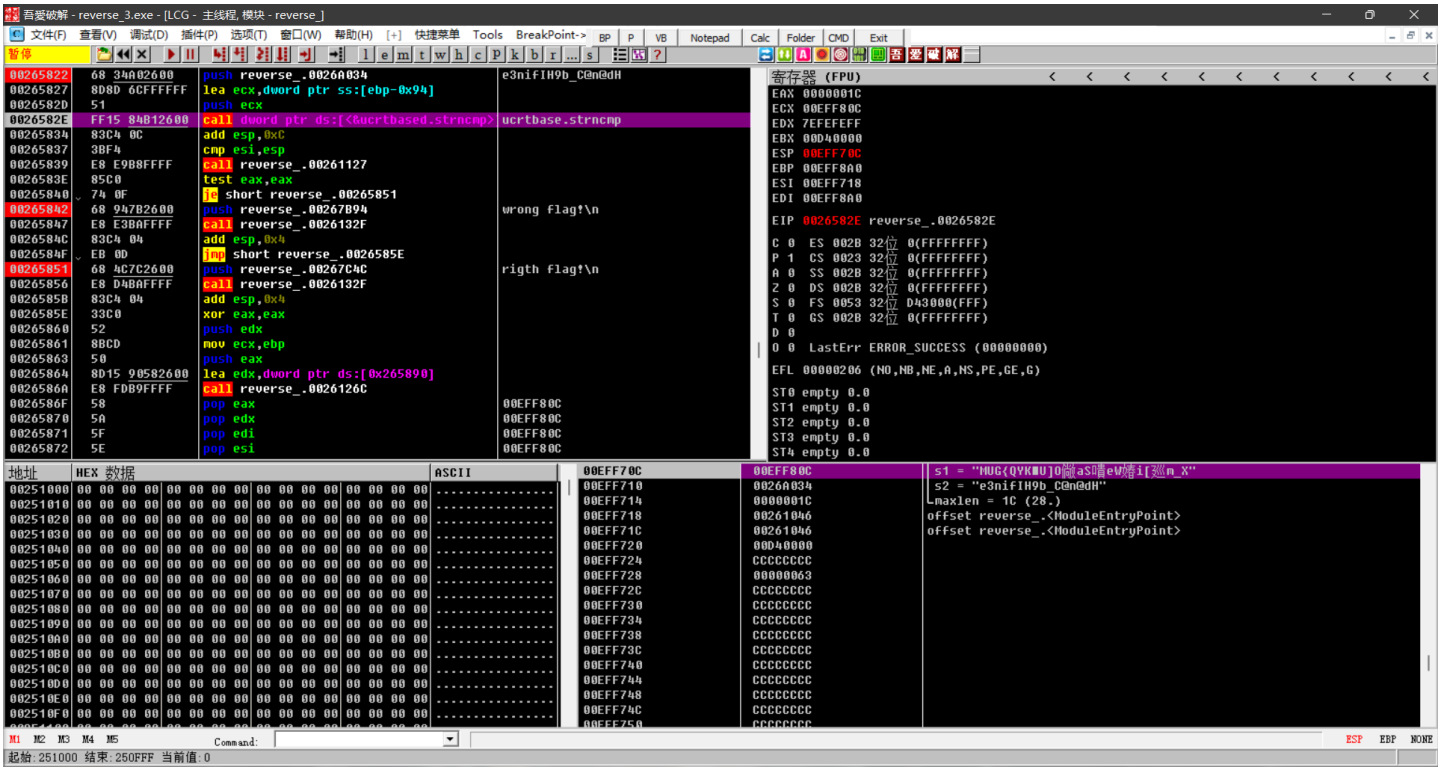
1. 简单分析

先拖入ida32位之中，找到`main函数`的代码位置：

```
int __cdecl main_0(int argc, const char **argv, const char **envp)
{
  size_t v3; // eax
  const char *v4; // eax
  size_t v5; // eax
  char v7; // [esp+0h] [ebp-188h]
  char v8; // [esp+0h] [ebp-188h]
  signed int j; // [esp+DCh] [ebp-ACh]
  int i; // [esp+E8h] [ebp-A0h]
  signed int v11; // [esp+E8h] [ebp-A0h]
  char Destination[108]; // [esp+F4h] [ebp-94h] BYREF
  char Str[28]; // [esp+160h] [ebp-28h] BYREF
  char v14[8]; // [esp+17Ch] [ebp-Ch] BYREF

  for ( i = 0; i < 100; ++i )
  {
    if ( (unsigned int)i >= 0x64 )
      j____report_rangecheckfailure();
    Destination[i] = 0;
  }
  sub_41132F("please enter the flag:", v7);
  sub_411375("%20s", (char)Str);
  v3 = j_strlen(Str);
  v4 = (const char *)sub_4110BE(Str, v3, v14);
  strncpy(Destination, v4, 0x28u);
  v11 = j_strlen(Destination);
  for ( j = 0; j < v11; ++j )
    Destination[j] += j;
  v5 = j_strlen(Destination);
  if ( !strncmp(Destination, Str2, v5) )
    sub_41132F("rigth flag!\n", v8);
  else
    sub_41132F("wrong flag!\n", v8);
  return 0;
}
```

逻辑并不复杂，顺便提一句，这种将输入的内容加密的这种逻辑使用od动调的方法应该是不行的，这里就应该使用静态分析了，这是因为动态调试并不能直接获得我们应该输入的flag，到最后我们也只能获得加密后的flag。



`s1`是我输入的内容加密后的结果，而`s2`应该就是flag加密后的结果了，这种很明显是不能直接获取flag的，必须写一些脚本来解密从而获取到flag。

2. 加密算法分析

然后我们可以对他进行的加密的伪代码分析一下，看看是怎么加密的，然后我们写个脚本解密一下就行了。

`sub_411375("%20s", (char)Str);`这个函数应该是用来接收输入的，这里可见我们的flag应该是不长于20个字符的，然后后面

```
v4 = (const char *)sub_4110BE(Str, v3, v14);
```

这个函数是正常的base64加密，后面的

```
for ( j = 0; j < v11; ++j )
    Destination[j] += j;
```

是个简单的移位密码，那我们解密的逻辑也非常清晰了，首先加密后的flag为`e3nifIH9b_C@n@dH`，我们先对这个加密后的flag进行反向移位获得base64编码后的内容，然后base64解一下码即可，下面写脚本。

3. 编写解密脚本获取flag

根据上面的加密算法的分析，这里写一个python脚本来解密是很简单的，下面是我写的解密脚本的内容：

```python
import base64

s2 = 'e3nifIH9b_C@n@dH'
s1 = ''

for i in range(len(s2)):
    s1 += chr(ord(s2[i]) - i)

print(base64.b64decode(s1))

# b'{i_l0ve_you}'
# flag{i_l0ve_you}
```

4. 总结

这题是简单的base64编码与简单的移位密码…

五、BugKu-mobile1(gctf)

1. 功能分析

简单安装一下看一看，随便输入一个内容弹出Toast，`错误!`

用jadx看看，关键部分代码如下：

```
@Override // android.app.Activity
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    setTitle(R.string.unregister);
    this.edit_userName = "Tenshine";
    this.edit_sn = (EditText) findViewById(R.id.edit_sn);
    this.btn_register = (Button) findViewById(R.id.button_register);
    this.btn_register.setOnClickListener(new View.OnClickListener() { // from class: com.example.crackme.MainAct
ivity.1
        @Override // android.view.View.OnClickListener
        public void onClick(View v) {
            if (!MainActivity.this.checkSN(MainActivity.this.edit_userName.trim(), MainActivity.this.edit_sn.get
Text().toString().trim())) {
                Toast.makeText(MainActivity.this, (int) R.string.unsuccessed, 0).show();
                return;
            }
            Toast.makeText(MainActivity.this, (int) R.string.successed, 0).show();
            MainActivity.this.btn_register.setEnabled(false);
            MainActivity.this.setTitle(R.string.registered);
        }
    });
}
......
private boolean checkSN(String userName, String sn) {
    if (userName == null) {
        return false;
    }
    try {
        if (userName.length() == 0 || sn == null || sn.length() != 22) {
            return false;
        }
        MessageDigest digest = MessageDigest.getInstance("MD5");
        digest.reset();
        digest.update(userName.getBytes());
        String hexstr = toHexString(digest.digest(), "");
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < hexstr.length(); i += 2) {
            sb.append(hexstr.charAt(i));
        }
        if (("flag{" + sb.toString() + "}").equalsIgnoreCase(sn)) {
            return true;
        }
        return false;
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
        return false;
    }
}
```

上面给出了用户名：`this.edit_userName = "Tenshine";`，需要校验码SN，校验码检查函数`checkSN`用来检查校验码的正确性，这个函数的主要逻辑为：

首先对username进行一个md5，然后对md5后获得的十六进制的32位字符串进行取偶数位字符，获得的内容加上`flag{}`就是正确的SN码了，也就是我们的flag

下面写出获取flag的代码

2. 写出解密代码

根据上面的分析写出解密代码获得flag

```
from hashlib import md5

username = b"Tenshine"

flag = ""
s1 = md5(username).hexdigest()
for i in range(0, 32, 2):
    flag += s1[i]
print("flag{" + flag + "}")

# flag{bc72f242a6af3857}
```

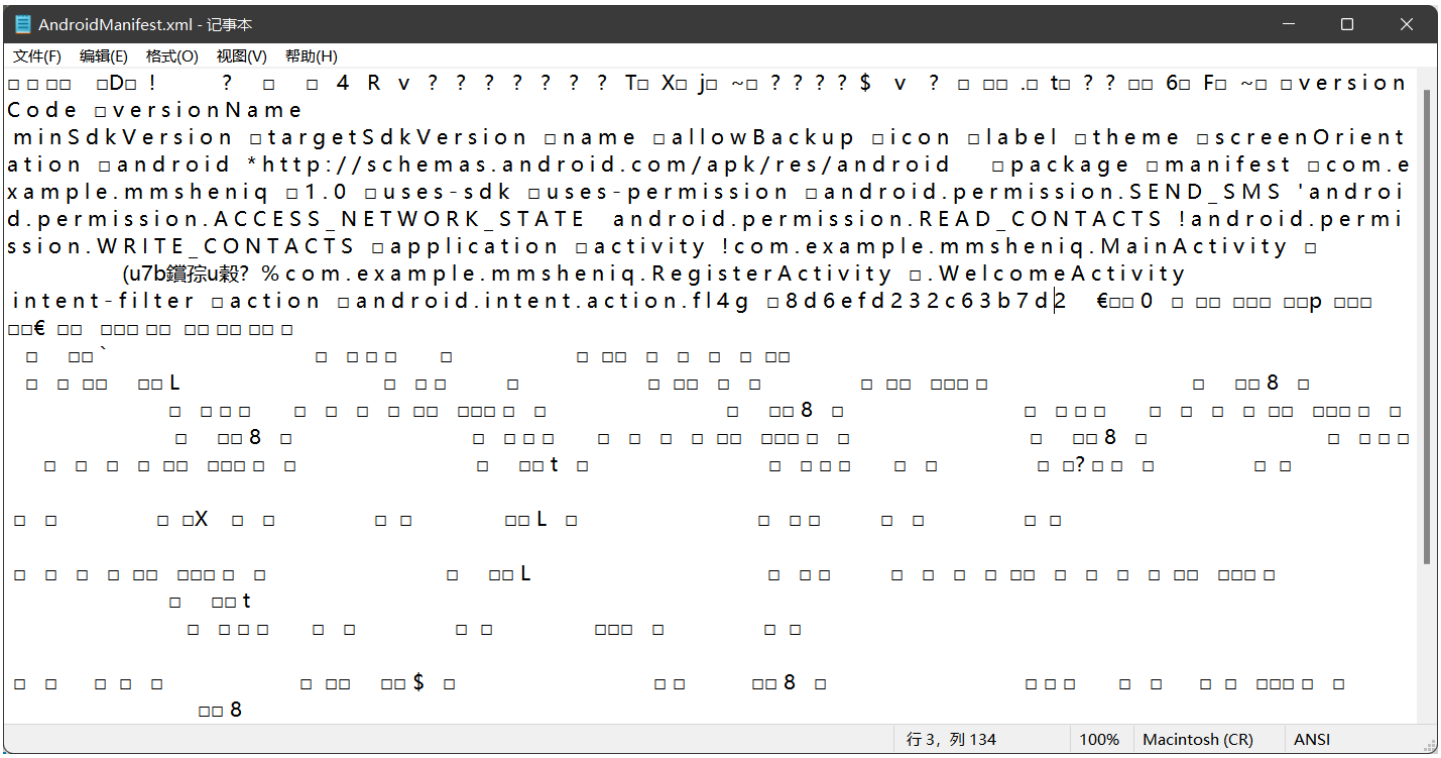
3. 总结

简单的md5校验，添了一丢丢的其他东西，但是还是签到题的难度…

六、BugKu-mobile2(gctf)

1. 傻弔题

我拖进jadx看了半天，啥都没发现，原来直接压缩包打开 `AndroidManifest.xml` 文件就有flag了…




```
fl4g{8d6efd232c63b7d2}
flag{8d6efd232c63b7d2}
```

七、BugKu-First_Mobile(xman)

1. 功能分析

拖进jadx，看主要代码：

```java
package com.example.xman.easymobile;

public class encode {
    private static byte[] b = {23, 22, 26, 26, 25, 25, 25, 26, 27, 28, 30, 30, 29, 30, 32, 32};

    public static boolean check(String str) {
        byte[] input = str.getBytes();
        byte[] temp = new byte[16];
        for (int i = 0; i < 16; i++) {
            temp[i] = (byte) ((input[i] + b[i]) % 61);
        }
        for (int i2 = 0; i2 < 16; i2++) {
            temp[i2] = (byte) ((temp[i2] * 2) - i2);
        }
        return new String(temp).equals(str);
    }
}
```

这个用java解比较简单，直接执行验证就行了，我没java的开发环境也，只好换成python。

2. 解码获得flag

```python
import string

dic = string.printable

b = [23, 22, 26, 26, 25, 25, 25, 26, 27, 28, 30, 30, 29, 30, 32, 32]


def decode():
    for i in range(len(b)):
        temp = ""
        for char in range(32, 128):
            temp = ((char + b[i]) % 61) * 2 - i
            if char == temp:
                print(chr(temp), end="")
                break


decode()
# XMAN{LOHILMNMLKHILKHI}
```

能用一个for循环来解决的问题非要用两个for循环.jpg，没什么难度，刚开始用flag{}包裹上交发现不对，看别人wp发现要用XMAN{}包裹。

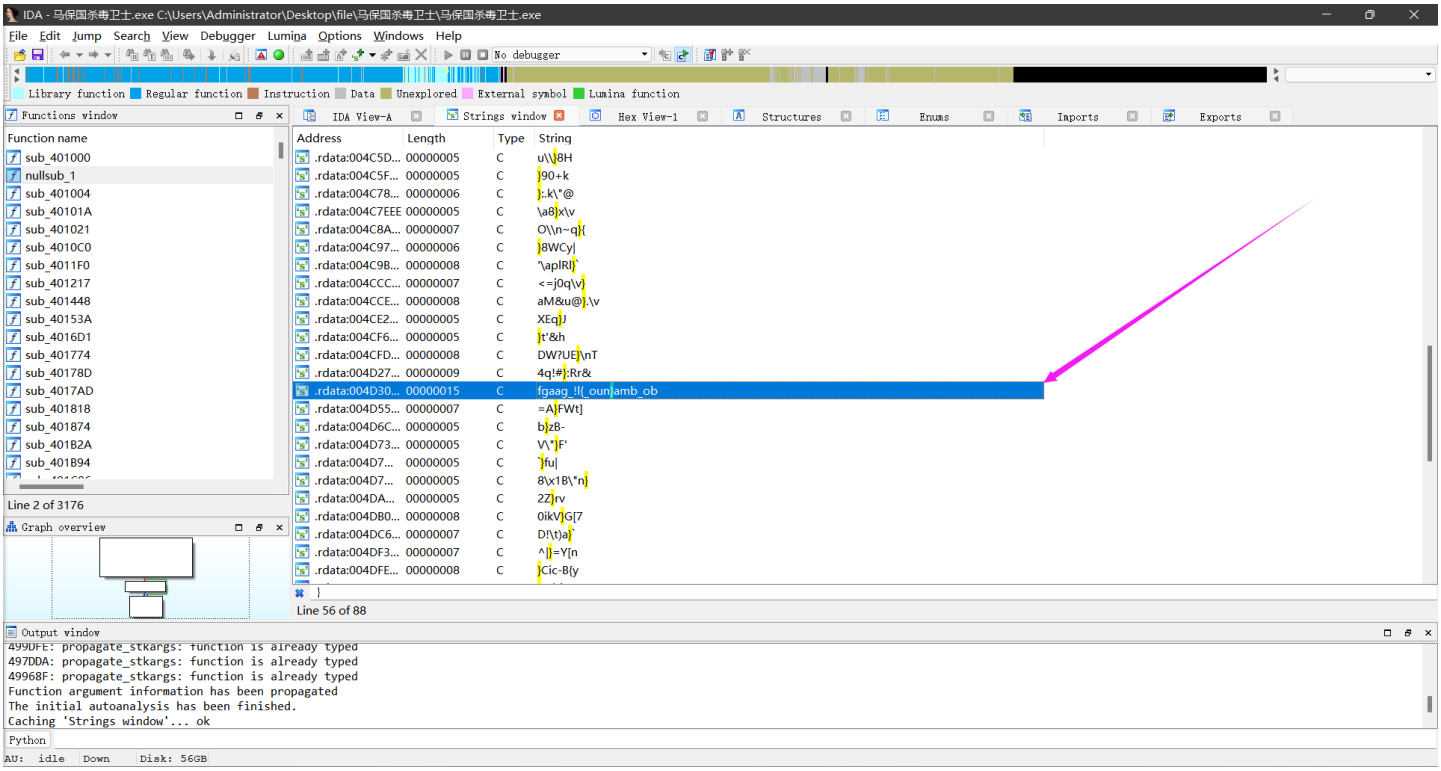
3. 总结

一般的爆破题，没什么难度…

八、BugKu-马老师杀毒卫士

1. 软件分析

首先看一下pe，发现是E语言，啊这，我连E语言的反编译工具都没有，一不小心有看了看wp，然后发现并不用用易语言的反编译工具，直接使用ida32打开看看，`shift+f12`看一下字符串，搜索字符`}`，找到一个跟flag非常相像的字符串



2. 获得flag

然后是栅栏密码，直接在线网站解码，是一个正常的三栏的栅栏密码

fgaag_!l{_oun}amb_ob

每组字数 3 加密 解密

flag{ma_bao_guo_nb!}

```
fgaag_!l{_oun}amb_ob

flag{ma_bao_guo_nb!}
```

3. 总结

有时候没什么思路的时候，`shift+f12`看看，也许会有意想不到的收获...

九、NoString

1. 分析

首先使用ida32打开文件，跳进去就看到了字符串，直接`f5反编译`来到主函数

```
int wmain()
{
  signed int v0; // ecx
  signed int i; // eax
  signed int v2; // ecx
  signed int j; // eax
  int k; // eax
  int v5; // eax
  signed int v6; // ecx
  signed int l; // eax
  signed int v8; // ecx
  signed int m; // eax
  char v11; // [esp+0h] [ebp-18h] BYREF
  __int128 v12; // [esp+1h] [ebp-17h]
  __int16 v13; // [esp+11h] [ebp-7h]

  v0 = strlen(Format);
  for ( i = 0; i < v0; ++i )
    Format[i] ^= 9u;
  printf("yelhzl)`gy|})|)oehnl3");
  v11 = 0;
  v13 = 0;
  v12 = 0i64;
  v2 = strlen(a80z);
  for ( j = 0; j < v2; ++j )
    a80z[j] ^= 9u;
  scanf(a80z, &v11);
  for ( k = 0; k < 19; ++k )
    *(&v11 + k) ^= 9u;
  v5 = strcmp(&v11, aOehnl3rHfCcgpt);
  if ( v5 )
    v5 = v5 < 0 ? -1 : 1;
  if ( v5 )
  {
    v6 = strlen(aLF);
    for ( l = 0; l < v6; ++l )
      aLF[l] ^= 9u;
    printf("l{{f{");
  }
  else
  {
    v8 = strlen(aNa);
    for ( m = 0; m < v8; ++m )
      aNa[m] ^= 9u;
    printf("{`na}");
  }
  printf("\r\n");
  system("pause");
  return 0;
}
```

可以看到它的输出都是一些乱码，但是真正函数输出的时候都是正确的，那就尝试一下看看它的输出跟他这里的字符串有什么联系：

```
s1 = "yelhzl)`gy|})||)oehnl3"
s2 = "please input u flage:"
for i in range(len(s1)):
    print(ord(s1[i]) ^ ord(s2[i]))
```

输出的是一堆9，那就是跟9异或了，找到一个变量`aOehnl3rHfCcgpt`，它的值为`oehnl3r=<?=hF@CCGPt`应该就是flag跟9异或之后的字符串，直接再与9一个一个地异或回去就得到了flag，但是多了一个`e:`，需要去掉

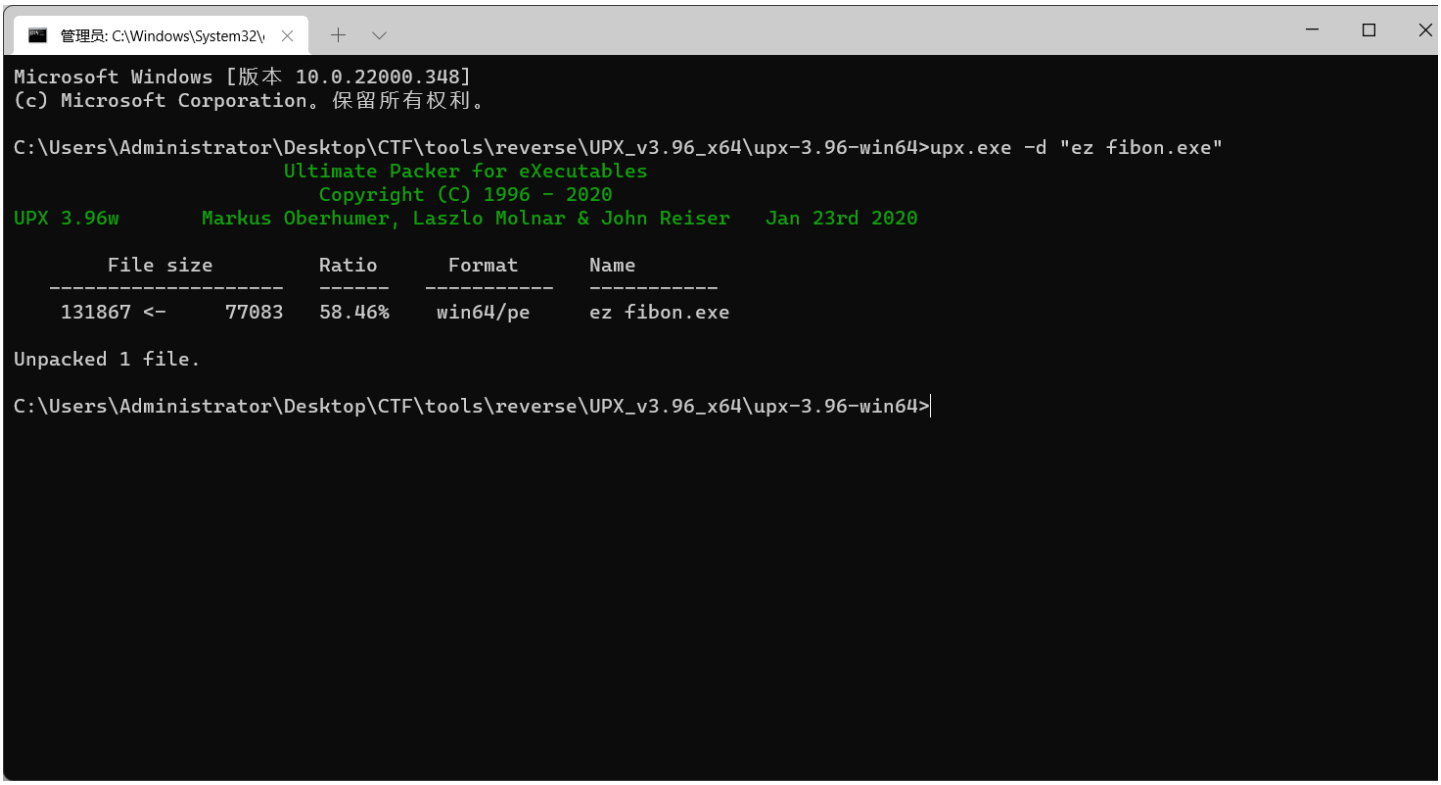
2. 获得flag

```
flag = ""
aOehnl3rHfCcgpt = "oehnl3r=<?=hF@CCGPt"
for i in range(len(aOehnl3rHfCcgpt)):
    flag += chr(ord(aOehnl3rHfCcgpt[i]) ^ 9)
print(flag)
# flage:{4564aOIJJNY}
# flag{4564aOIJJNY}
```

十、ez fibon

1. 脱壳

这是一个有壳的64位的程序，先使用官方的脱壳工具脱一下壳。



2. 分析&破解

然后对这个程序进行反编译，进main函数代码如下：

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // edx
  int v5[24]; // [rsp+20h] [rbp-60h]
  char Str[524]; // [rsp+80h] [rbp+0h] BYREF
  int j; // [rsp+28Ch] [rbp+20Ch]
  int v8; // [rsp+290h] [rbp+210h]
```

```
int v9; // [rsp+294h] [rbp+214h]
int i; // [rsp+298h] [rbp+218h]
int v11; // [rsp+29Ch] [rbp+21Ch]

_main();
v11 = 1;
puts("please input your flag:");
gets(Str);
for ( i = 0; i <= 21; ++i )
  *(_DWORD *)&Str[4 * i + 112] = Str[i];
if ( strlen(Str) == 22 )
{
  v9 = 1;
  v8 = 1;
  for ( j = 0; j <= 21; ++j )
  {
    if ( (j & 1) != 0 )
    {
      v8 += v9;
      v3 = (v8 + j + *(_DWORD *)&Str[4 * j + 112]) % 64 + 64;
    }
    else
    {
      v9 += v8;
      v3 = (v9 + j + *(_DWORD *)&Str[4 * j + 112]) % 64 + 64;
    }
    *(_DWORD *)&Str[4 * j + 112] = v3;
  }
  v5[0] = 100;
  v5[1] = 121;
  v5[2] = 110;
  v5[3] = 118;
  v5[4] = 70;
  v5[5] = 85;
  v5[6] = 123;
  v5[7] = 109;
  v5[8] = 64;
  v5[9] = 94;
  v5[10] = 109;
  v5[11] = 99;
  v5[12] = 116;
  v5[13] = 81;
  v5[14] = 109;
  v5[15] = 86;
  v5[16] = 83;
  v5[17] = 126;
  v5[18] = 119;
  v5[19] = 101;
  v5[20] = 110;
  v5[21] = 114;
  for ( j = 0; j <= 21; ++j )
  {
    if ( v5[j] != *(_DWORD *)&Str[4 * j + 112] )
      v11 = 0;
  }
  if ( !v11 )
    printf("wrong!");
  if ( v11 == 1 )
    printf("right flag!");
```

```
  }
  else
  {
    printf("wrong lenth!");
  }
  return 0;
}
```

主要看下面的这一段代码

```
v9 = 1;
v8 = 1;
for ( j = 0; j <= 21; ++j )
{
  if ( (j & 1) != 0 )
  {
    v8 += v9;
    v3 = (v8 + j + *(_DWORD *)&Str[4 * j + 112]) % 64 + 64;
  }
  else
  {
    v9 += v8;
    v3 = (v9 + j + *(_DWORD *)&Str[4 * j + 112]) % 64 + 64;
  }
  *(_DWORD *)&Str[4 * j + 112] = v3;
}
```

这其实就对应了题目的名字，是个斐波那契数列`*(_DWORD *)&Str[4 * j + 112]`这是一个字符，是未进行变换前的flag，这里应该是可以直接按位爆破的，会省去很多的算法分析，但是我们追求的就是困难的道路，这里写个python脚本逆一下。

```
slist = ['d', 'y','n', 'v', 'F', 'U', '{', 'm', '@', '^', 'm', 'c', 't', 'Q', 'm' ,'V' ,'S' , '~' ,'w' ,'e' ,'n'
 ,'r' ]

flag = ''
v9 = 1
v8 = 1
for j in range(22):
    if (j & 1) != 0:
        v8 += v9
        tmp = ord(slist[j])-v8-j
    else:
        v9 += v8
        tmp = ord(slist[j])-v9-j
    tmp = tmp % 64 + 64
    flag += chr(tmp)
print(flag)
# bugku{So_Ez_Fibon@cci}
```

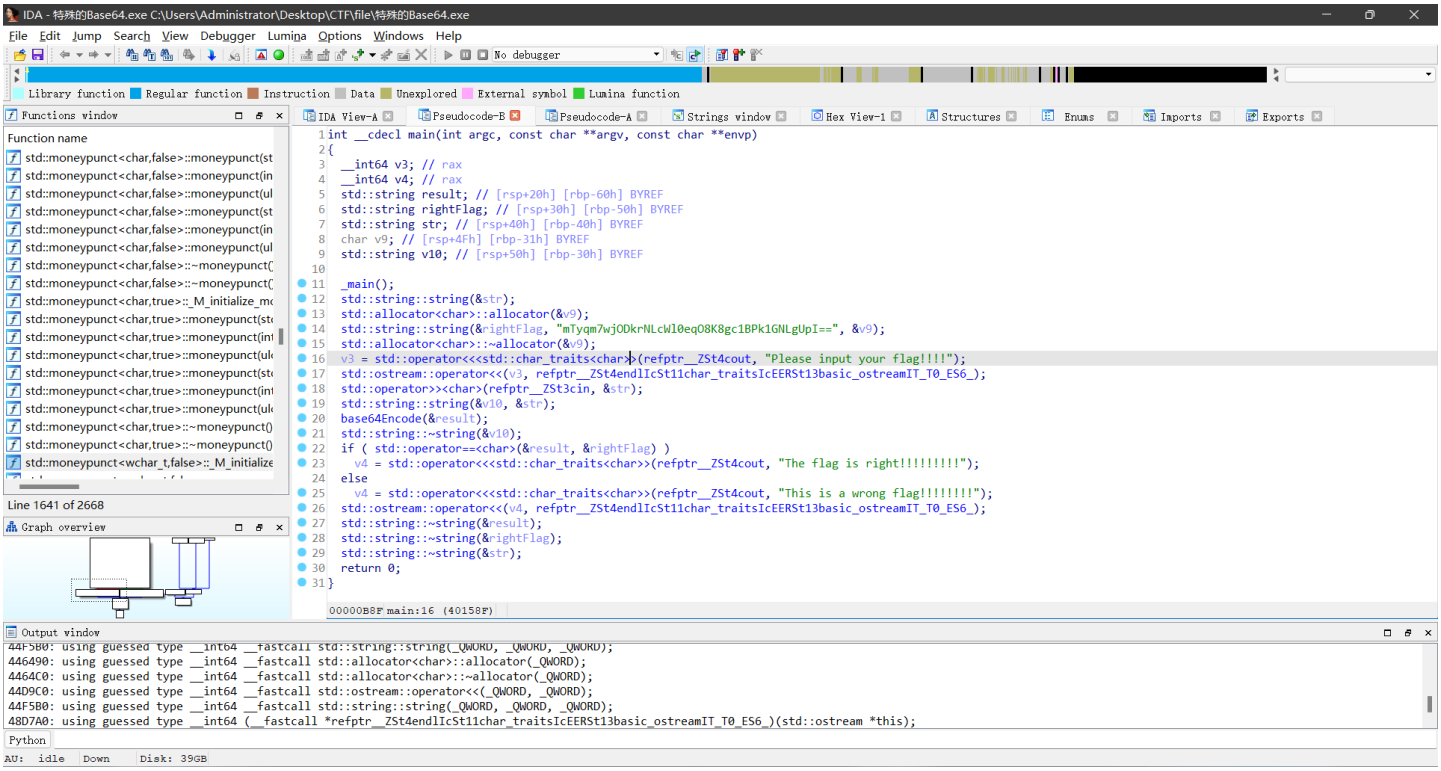
前面恢复什么的都没什么含金量，主要是下面的这个`tmp = tmp % 64 + 64`，这是因为我们使用的大多数的ASCII字符都是分布在这个`64 ~ 128`之间的。

```
>>> print(ord('a'))
97
>>> print(ord('A'))
65
>>> print(ord('z'))
122
>>>
```

十一、特殊的Base64

1. 功能分析

64位无壳，直接`shift+f12`看字符串，发现`换表base64表`，还有密文，直接省去对程序分析了



2. 换表base64

直接使用换表base64的脚本带入数据即可

```
import base64

str1 = "mTyqm7wjODkrNLcWl0eqO8K8gc1BPk1GNLgUpI=="

string1 = "AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz0987654321/+"
string2 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"

print (base64.b64decode(str1.translate(str.maketrans(string1,string2))))
# b'flag{Special_Base64_By_Lich}'
```

十二、不好用的ce

1. 没什么好说的，没用ce

转换前:

DeZmqMUhRcP8NgJgzLPdXa

编码Base58> 解码Base58>

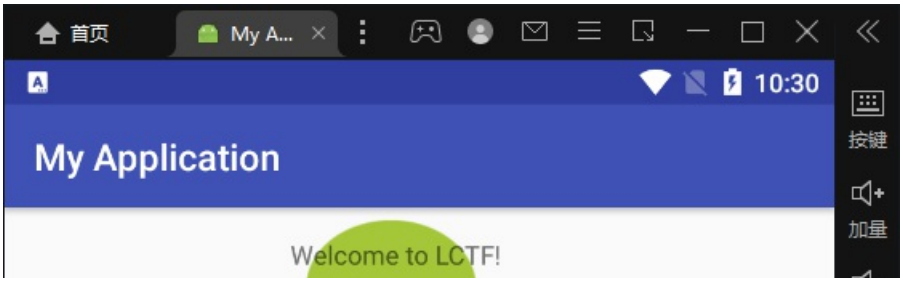
转换后:

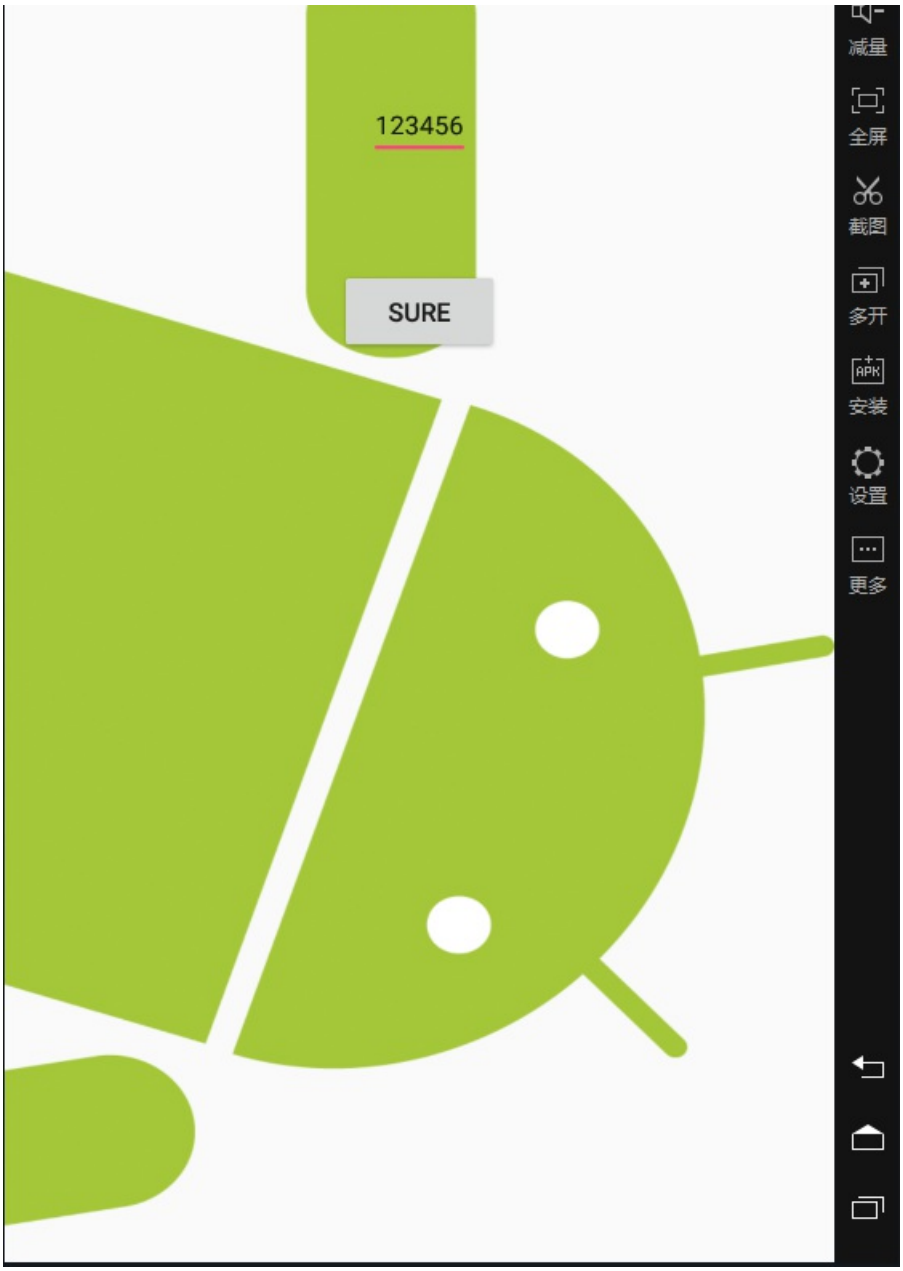
flag{c1icktimes}

```
flag{c1icktimes}
```

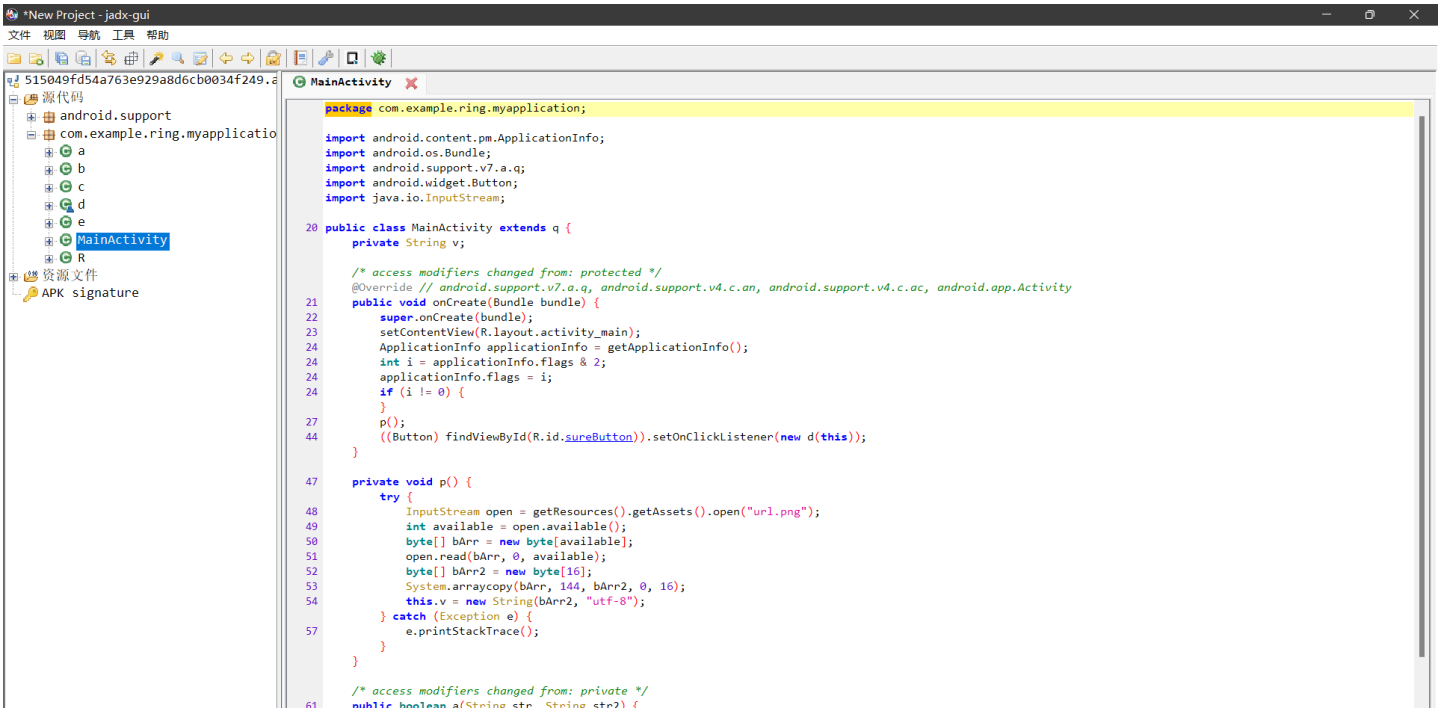
十三、easy-100(LCTF)

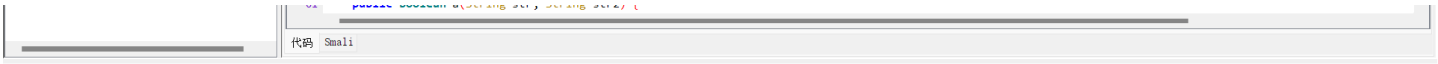
1. 分析





安装打开程序看一下界面，然后用jadx打开看看。





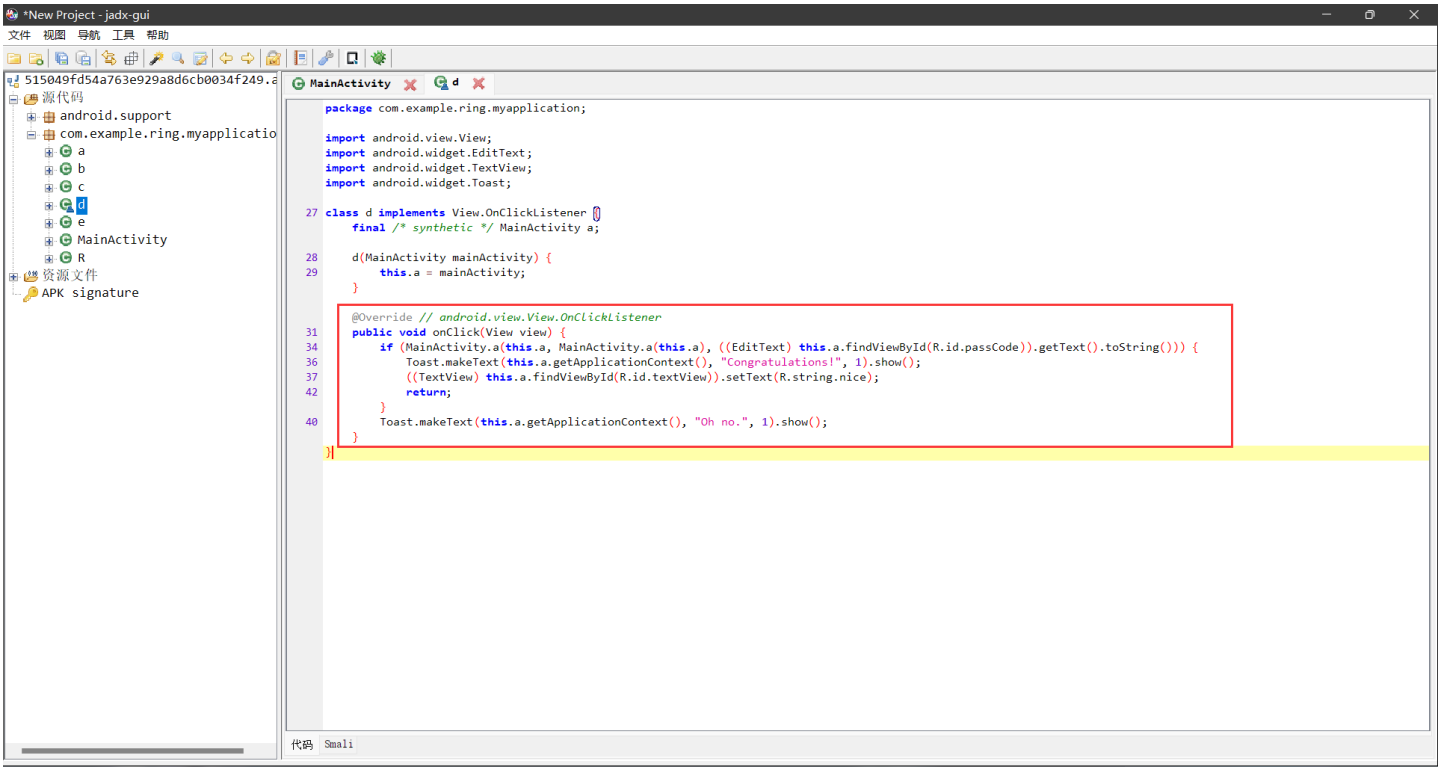
看起来好复杂啊晕，注意到

```
p();
((Button) findViewById(R.id.sureButton)).setOnClickListener(new d(this));
```

可见需要先进性一个p函数，就在本类之中

```
private void p() {
    try {
        InputStream open = getResources().getAssets().open("url.png");
        int available = open.available();
        byte[] bArr = new byte[available];
        open.read(bArr, 0, available);
        byte[] bArr2 = new byte[16];
        System.arraycopy(bArr, 144, bArr2, 0, 16);
        this.v = new String(bArr2, "utf-8");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

然后本类中的`v`的值就被赋为了这个`url.png`图片的`144 ~ 160`位的值字符串了。然后就是`setOnClickListener(new d(this));`这里了，这个是点击按钮的事件，看看`类d`中都有什么。

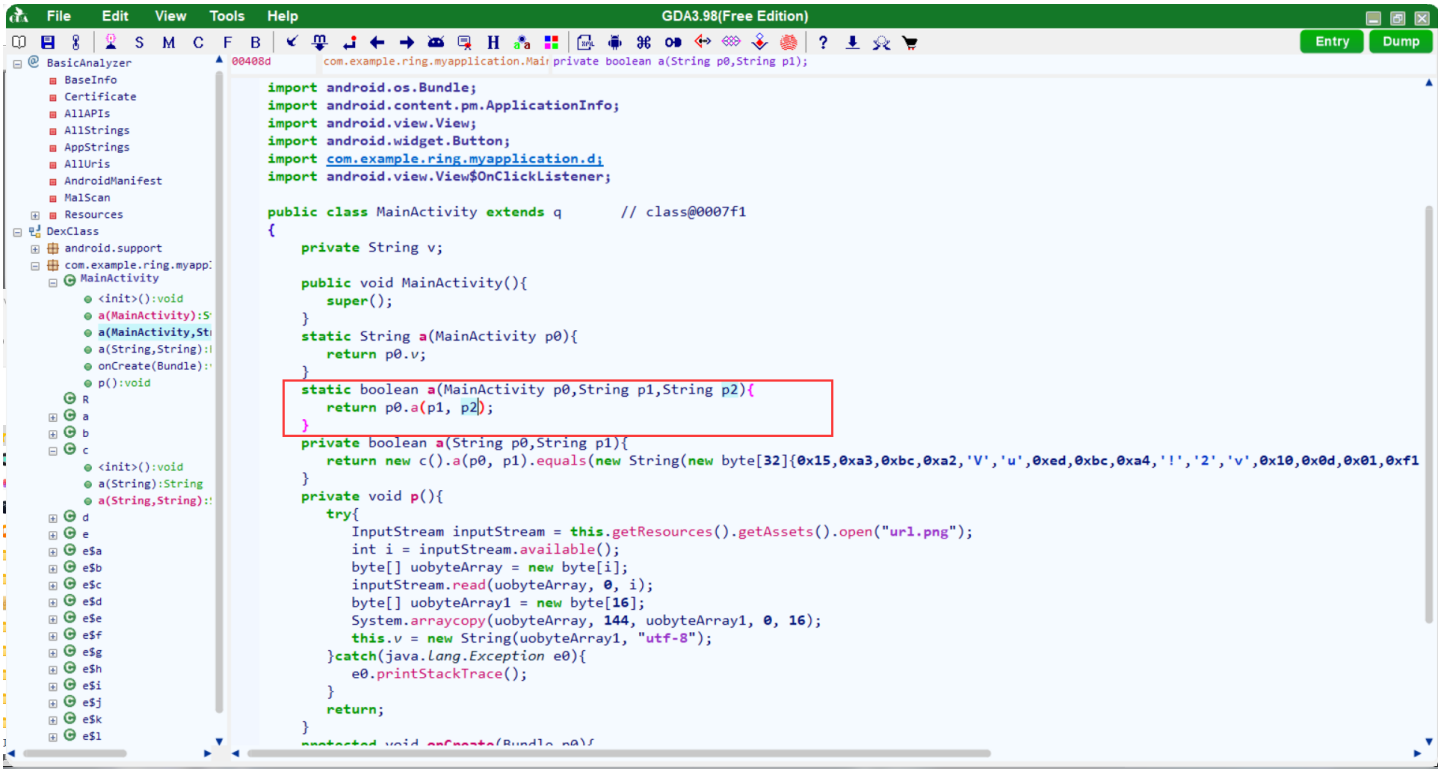


看来主要的代码在`类d`之中，满足以下条件则判断为成功，输入的即是正确的flag

```
MainActivity.a(this.a, MainActivity.a(this.a), ((EditText) this.a.findViewById(R.id.passCode)).getText().toStrin
g())
```

> 第一个参数是一个句柄
> 第二个参数是调用了mainactivity的a函数，返回一个字符串
> 第三个参数是输入的flag

一看jadx没有三个参数的函数重载形式，果断换用gda3.98



找到了，但是没什么用，还是将后两个参数传到了两个字符串参数的a函数里面了。
注意：一个参数的a函数返回的是刚才分析的v参数的值
然后下面的`str`就是`v`、`str2`就是`flag`，后面的字节数组就是`进行c类中的a函数加密之后的密文`了

```
public boolean a(String str, String str2) {
    return new c().a(str, str2).equals(new String(new byte[]{21, -93, -68, -94, 86, 117, -19, -68, -92, 33, 50,
118, 16, 13, 1, -15, -13, 3, 4, 103, -18, 81, 30, 68, 54, -93, 44, -23, 93, 98, 5, 59}));
}
```

看看c类中的a函数

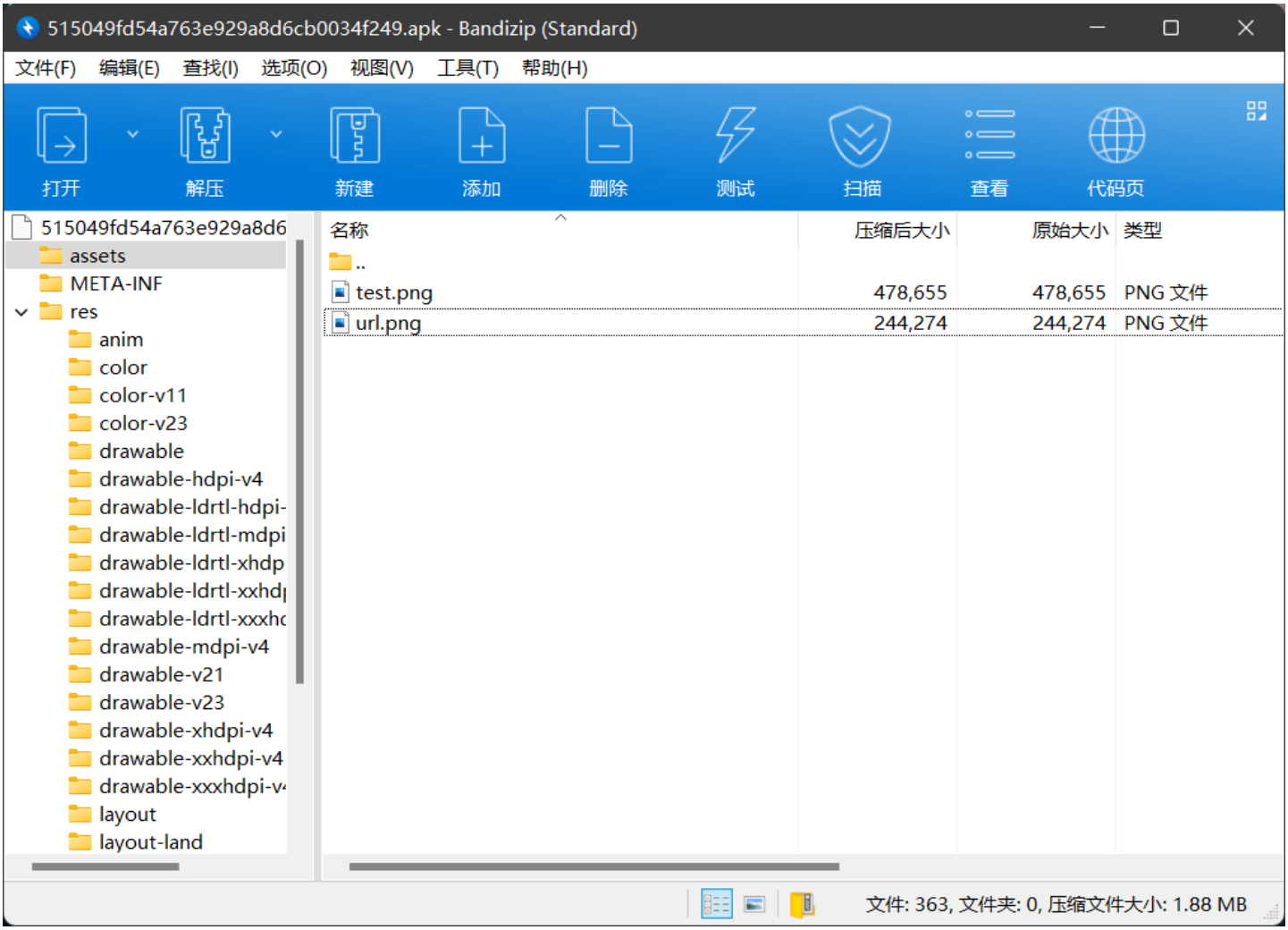
```
//首先是对v进行了一个变换
String a = a(str);
```

然后a类中以`v`为key，以`flag`为明文，进行了下面的这种类型的加密。

```
AES/ECB/PKCS5Padding
```

好了，直接可以试着写脚本了。

2. getflag


515049fd54a763e929a8d6cb0034f249.apk - Bandizip (Standard)
文件(F) 编辑(E) 查找(I) 选项(O) 视图(V) 工具(T) 帮助(H)
打开
解压
新建
添加
删除
测试
扫描
查看
代码页
515049fd54a763e929a8d6
assets
META-INF
res
anim
color
color-v11
color-v23
drawable
drawable-hdpi-v4
drawable-ldrtl-hdpi-
drawable-ldrtl-mdpi
drawable-ldrtl-xhdp
drawable-ldrtl-xxhd
drawable-ldrtl-xxxh
drawable-mdpi-v4
drawable-v21
drawable-v23
drawable-xhdpi-v4
drawable-xxhdpi-v4
drawable-xxxhdpi-v
layout
layout-land
名称
压缩后大小
原始大小
类型
..
test.png
478,655
478,655
PNG 文件
url.png
244,274
244,274
PNG 文件
文件: 363, 文件夹: 0, 压缩文件大小: 1.88 MB

```python
from os import path
from Crypto.Cipher import AES
from binascii import a2b_hex


cipher =[21, -93, -68, -94, 86, 117, -19, -68, -92, 33, 50, 118, 16, 13, 1, -15, -13, 3, 4, 103, -18, 81, 30, 68
, 54, -93, 44, -23, 93, 98, 5, 59]
v = ''
with open('url.png', 'rb') as file:
    data = file.read()[144:160]
for i in range(0, len(data), 2):
    v += chr(data[i+1])
    v += chr(data[i])


def AES_decrypt(secret_key, encrypted_text_hex):
    """
    :param secret_key [str] : 加密秘钥
    :param encrypted_text_hex [str]: # 加密后的 data 字符串
    :return [str]:
    """
    # 去掉 PKCS5Padding 的填充
    unpad = lambda s: s[:-ord(s[len(s) - 1:])]
    # 通过 key 值进行
    cipher = AES.new(secret_key.encode(), AES.MODE_ECB)
    data_response = unpad(cipher.decrypt(a2b_hex(encrypted_text_hex))).decode('utf8')
    return data_response


cipertext = ''
for i in cipher:
    s = str(hex((i+256)%256))
    print(s)
    if len(s) < 4:
        cipertext = cipertext + '0' + s[2:]
    else:
        cipertext += s[2:]


flag = AES_decrypt(v, cipertext)
print(flag)
# LCTF{1t's_rea1ly_an_ea3y_ap4}
```

总结

本来就是打算一天一道题的，现在忙里偷闲也算是把自己设置的任务完成了，还行，就差15分钟就完不成了哈哈...

一天一道，已完成