这周是我比较遗憾的一周，因为这周七天我只有三天是在做题的，而且这三天身体还不是特别好。其余的时间帮家里做事 and 匹配我的新电脑（是真的麻烦）

一．Reverse

1 before_main

在 IDA 里面打开后直接 F5

```
7
8   v7 = __readfsqword(0x28u);
9   printf("input your flag:");
10  __isoc99_scanf("%s", v6);
11  s2 = (char *)sub_12EB(v6);
12  strcpy(s1, "AMHo7dLxUEabf6Z3PdWr6cOy75i4fdfeUzL17kaV7rG=");
13  if ( !strcmp(s1, s2) )
14    puts("congratulations!");
15  else
16    puts("sorry!");
17  return 0LL;
18}
```

可以看到就一个 sub_12EB 加密函数，而且从密文的形式上可以推测到是一个 base64 加密。那么进入这个函数看一看：

```
7   _BYTE *v6; // [rsp+28h] [rbp-8h]
8
9   v5 = strlen(a1);
10  if ( v5 % 3 )
11    v4 = 4 * (v5 / 3 + 1);
12  else
13    v4 = 4 * (v5 / 3);
14  v6 = malloc(v4 + 1);
15  v6[v4] = 0;
16  v2 = 0;
17  v3 = 0;
18  while ( v2 < v4 - 2 )
19  {
20    v6[v2] = *((_BYTE *)&qword_4020 + ((unsigned __int8)a1[v3] >> 2));
21    v6[v2 + 1] = *((_BYTE *)&qword_4020 + ((16 * a1[v3]) & 0x30 | (unsigned int)((unsigned __int8)a1[v3 + 1] >> 4)));
22    v6[v2 + 2] = *((_BYTE *)&qword_4020 + ((4 * a1[v3 + 1]) & 0x3C | (unsigned int)((unsigned __int8)a1[v3 + 2] >> 6)))
23    v6[v2 + 3] = *((_BYTE *)&qword_4020 + (a1[v3 + 2] & 0x3F));
24    v3 += 3;
25    v2 += 4;
26  }
27  if ( v5 % 3 == 1 )
28  {
29    v6[v2 - 2] = 61;
30    v6[v2 - 1] = 61;
31  }
32  else if ( v5 % 3 == 2 )
33  {
34    v6[v2 - 1] = 61;
35  }
36  return v6;
37}
```

很明显的 base64 加密，shift+F12 看一看字符串，找到 base64 表。嗯……没有找到。那就把这个函数里面的 qword_4020 点开看一看

```
qword_4020 dq 6D654F7357784330h




qword_4028 dq 326B647A34714A76h
qword_4030 dq 396A72416C513656h
qword_4038 dq 664E317462486E77h
qword_4040 dq 796844332B2F5845h
qword_4048 dq 7038594C52426F50h
qword_4050 dq 75615A696346354Bh
qword_4058 dq 47535467494D5537h
byte_4060 db 0
```

转化为字符串的样式，先转化为 64 大小的数组，再按 A 转化为字符串

```
align 20h
a0cxwsoemvjq4zd db '0CxWsOemvJq4zdk2V6QlArj9wnHbt1NfEX/+3DhyPoBRLY8pK5FciZau7UMIgTSG',
```

去找个网站解密一下

```
S¦©àÛ.æ
.}.d ÐÕG1gã..|×ÆäË.àí.á_
```

很明显表不对，结合题目分析可知程序在某个地方提前改了表，按 X 找到程序其他的引用了的地方。

```
if ( result ≠ -1 )
{
  strcpy(a0cxwsoemvjq4zd, "qaCpwYM2tO/RP0XeSZv8kLd6nfA7UHJ1No4gF5zr3VsBQbl9juhEGymc+WTxIiDK");
  return 0x636D79474568756ALL;
}
return result;
```

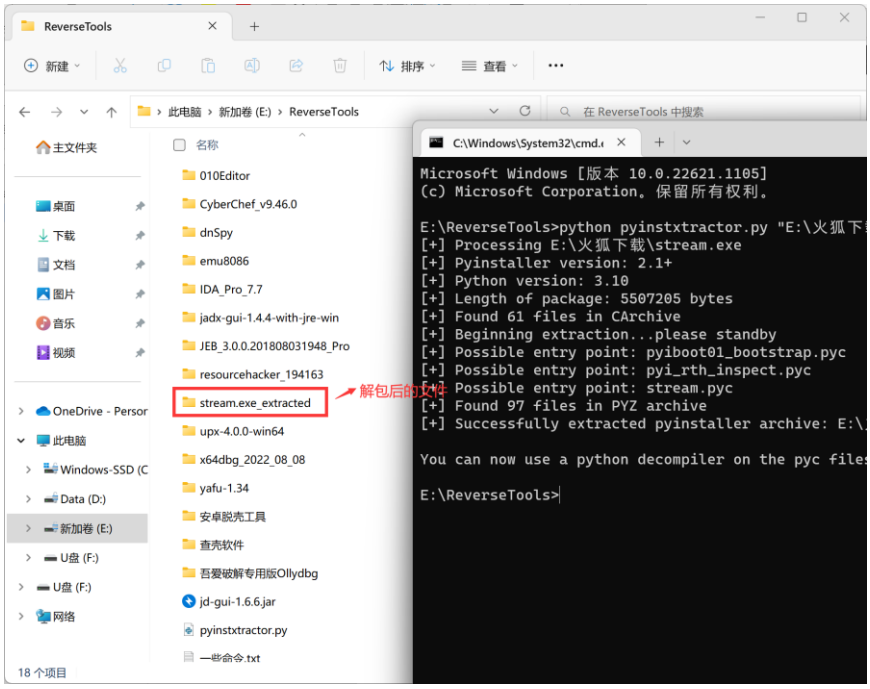
果然改了表，再拿去试一试

```
hgame{s0meth1ng_run_befOre_m@in}
```

flag 直接就出来了。

2 stream

从解压缩的后的图标中可以得知这个题目是一个 python 逆向题目，对于这个题目的思路是，先用 pyinstxtractor.py（github 上可以下载到源码）用 python pyinstxtractor.py "E:\火狐下载\stream.exe"对这个程序进行解包



在将这个文件里面的 stream.pyc 文件拿到 python 的反编译网站去就可以看到源码

```python
# Version: Python 3.10

import base64

def gen(key):
    s = list(range(256))
    j = 0
    for i in range(256):
        j = (j + s[i] + ord(key[i % len(key)])) % 256
        tmp = s[i]
        s[i] = s[j]
        s[j] = tmp
    i = j = 0
    data = []
    for _ in range(50):
        i = (i + 1) % 256
        j = (j + s[i]) % 256
        tmp = s[i]
        s[i] = s[j]
        s[j] = tmp
```

典型的 RC4 + base64 那么思路就很清晰了，先用 base64 进行解密，再用 RC4 加密的数字异或一遍就可以解决。因为 RC4 用于加密的数据是不变的，那么我们就可以将源码重新跑一边得到加密数据（同样也是解密数据）。那按照思路开始吧

首先 base64 解密

```
¼.W.ù.Ü. íí2KÏ.zj.É.¼"»(Îjëv.×.éª½!ý5/.{íµå
```

好家伙！乱了。其实这道题我花了很长时间在上面，因为按照我的常规思路，base64 解出乱码就是错了，但是我万万没想到这次解出的就是不可见字符（这次经验害了我）。于是我在网上找了一个将 base64 解码成 byte 字节的脚本

```
import binascii
import struct
import base64
encoded = b'wr3ClVcSw7nCmMOcHcKgacOtMkvDjxZ6asKWw4nChMK8IsK7KMOOasOrdgbDlx3DqcKqwr0hw701Ly57w63CtcOl'
print(binascii.a2b_base64(encoded))
```

得到结果

```
b'\xc2\xbd\xc2\x95W\x12\xc3\xb9\xc2\x98\xc3\x9c\x1d\xc2\xa0i\xc3\xad2K\xc3\x8f\x16zj\xc2\x96\xc3\x89\xc2\x84\xc2\xbc"\xc2\xbb(\xc3\x8ej\xc3\xabv\x06\xc3
 \x97\x1d\xc3\xa9\xc2\xaa\xc2\xbd!\xc3\xbd5/.{\xc3\xad\xc2\xb5\xc3\xa5'
```

然后我用 RC4 的解密数据异或一遍，结果仍然是错的。到了这里我其实已经被卡死了，因为我不知道自己那一步做错了才会导致这个结果（后面明白过来知道之前没有错），于是我就开始“胡乱”的做，在“机缘巧合”之下我看到了，\bd 被 encode()函数加密成了\c2\bd 多了一个\c2。于是我灵光一现，enconde()加密之后的字节数据和之前的不一样了，于是我一个一个对比，只找到了\bd 变为了\c2\bd，\xf9 变为了\xc3\xb9，于是我手动恢复过来后得到 b'\xbd\x95W\x12\xf9\x98\xdc\x1d\xa0i\xed2K\xcf\x16zj\x96\xc9\x84\xbc"\xbb(\xcej\xebv\x06\xd7\x1d\xe9\xaa\xbd!\xfd5/.{\xed\xb5\xe5'密文，再跑一遍得到了

```
data1 = [213, 242, 54, 127, 156, 227, 172, 100, 212, 1, 130, 92, 20, 189, 115, 12, 15, 228, 186, 225, 227, 75, 200, 119, 171, 11, 152, 15, 89, 160, 116, 157, 194, 226, 72, 147, 65,
enc = b'\xbd\x95W\x12\xf9\x98\xdc\x1d\xa0i\xed2K\xcf\x16zj\x96\xc9\x84\xbc"\xbb(\xcej\xebv\x06\xd7\x1d\xe9\xaa\xbd!\xfd5/.{\xed\xb5\xe5'
for c,k in zip(data1,enc):
    print(chr(c ^ k),end='')
flag = hgame{python_reverse_is_easy_with_internet}
```

这个题目我做的挺失败的，所以之后需要看官方 wp 好好复盘。

3 math

将程序拖进 IDA 按 F5 反编译一下

```
  for ( i = 0; i <= 4; ++i )
  {
    for ( j = 0; j <= 4; ++j )
    {
      for ( k = 0; k <= 4; ++k )
        v11[5 * i + j] += *((char *)&savedregs + 5 * i + k - 368) * v10[5 * k + j];
    }
  }
  for ( m = 0; m <= 24; ++m )
  {
    if ( v11[m] != v12[m] )
    {
      printf("no no no, your match is terrible...");
      exit(0);
    }
  }
  printf("yes!");
  return 0LL;
}
```

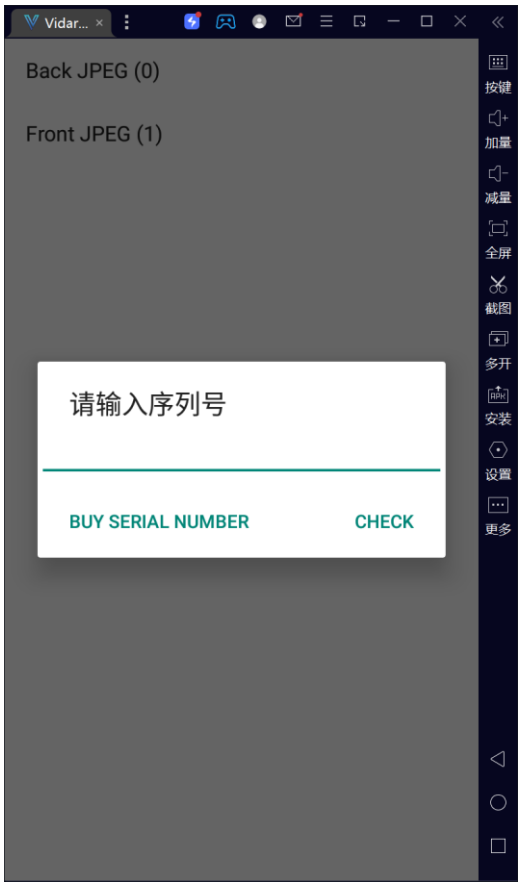
非常明显的矩阵特征，那就按照矩阵直接求解就行

```
from sympy import *
B = Matrix([
    [126, 225, 62, 40, 216],
    [253, 20, 124, 232, 122],
    [62, 23, 100, 161, 36],
    [118, 21, 184, 26, 142],
    [59, 31, 186, 82, 79]
])
C = Matrix([
    [63998, 33111, 67762, 54789, 61979],
    [69619, 37190, 70162, 53110, 68678],
    [63339, 30687, 66494, 50936, 60810],
    [48784, 30188, 60104, 44599, 52265],
    [43048, 23660, 43850, 33646, 44270]
])
print(C*B**(-1))
```

就可以得到 flag 了 hgame{y0ur_m@th_1s_gO0d}

4 VidarCamera

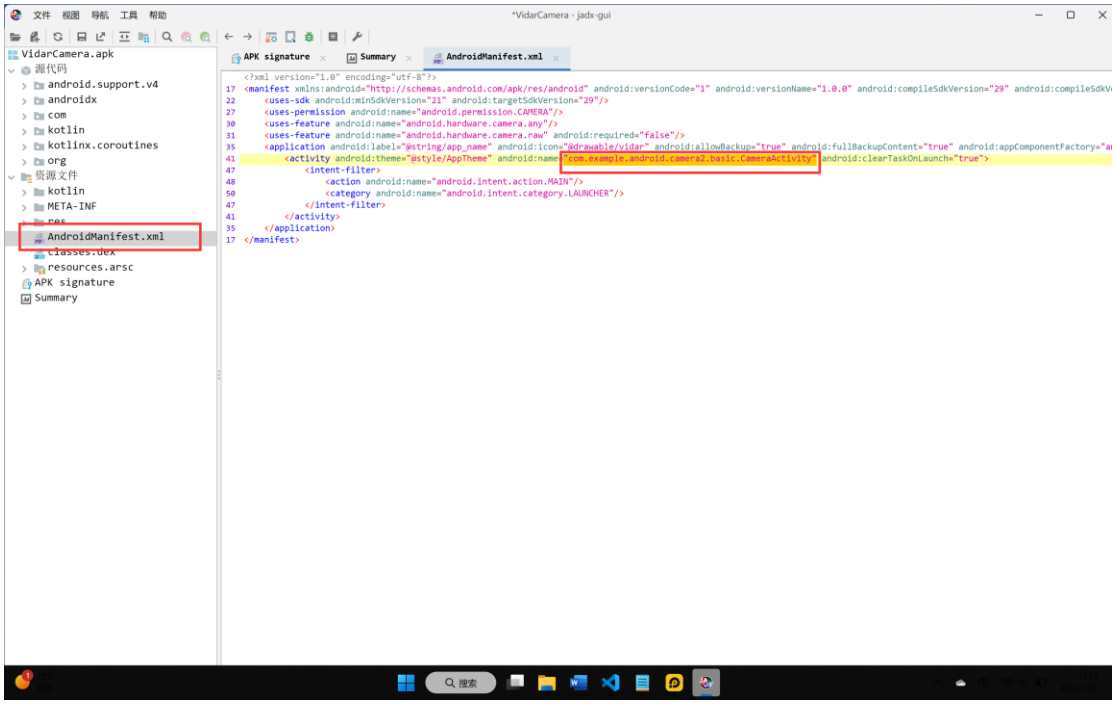
解压缩发现是一个 apk 文件，那这道题目就是安卓逆向了。我先在模拟器中运行一下



功能很简单，输入 check 一下，错误退出。

用 jadx-gui 打开，首先去到 AndroidManifest.xml 下，因为这里面有一些 apk 文件的配置信息比如是否能调试、文件的入口。是

com.example.android.camera2.basic.CameraActivity



那么去到这里就可以看到主要的代码逻辑了

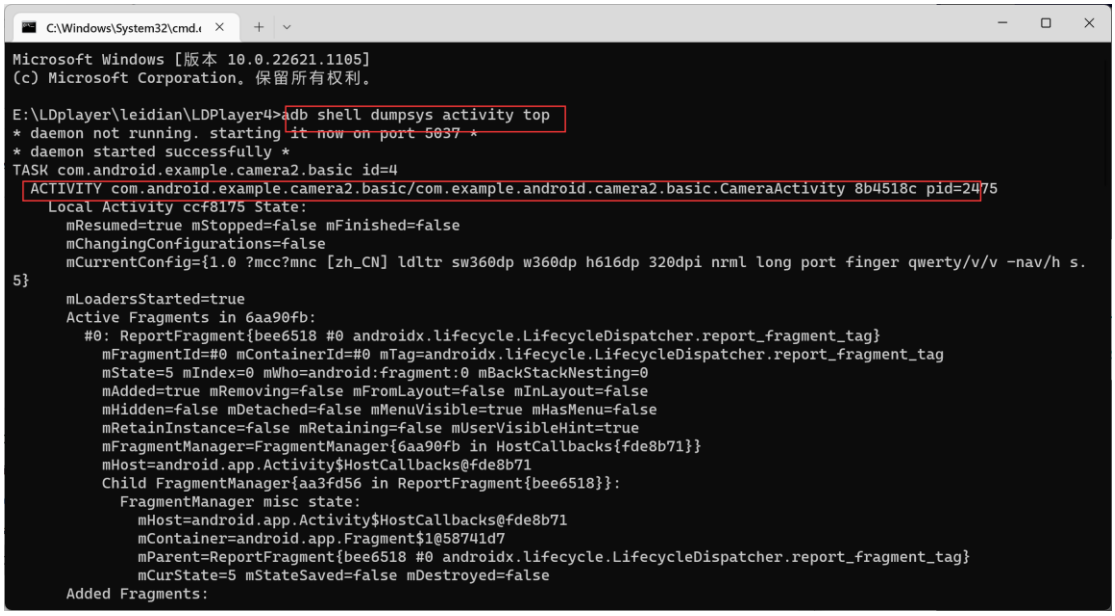
```
int i;
int[] m446constructorimpl = UIntArray.m446constructorimpl(4);
UIntArray.m457setVXSXFK8(m446constructorimpl, 0, 2233);
UIntArray.m457setVXSXFK8(m446constructorimpl, 1, 4455);
UIntArray.m457setVXSXFK8(m446constructorimpl, 2, 6677);
UIntArray.m457setVXSXFK8(m446constructorimpl, 3, 8899);    XTEA加密
int i2 = 0;
while (i2 < 9) {
    int i3 = 0;
    int i4 = 0;
    do {
        i3++;
        i = i2 + 1;
        UIntArray.m457setVXSXFK8(iArr, i2, UInt.m393constructorimpl(UIntArray.m452getpVg5ArA(iArr, i2) + UInt.m393constructorimpl(UInt.m393constructorimpl(UInt.m393constructori
        UIntArray.m457setVXSXFK8(iArr, i, UInt.m393constructorimpl(UIntArray.m452getpVg5ArA(iArr, i) + UInt.m393constructorimpl(UInt.m393constructorimp
        i4 = UInt.m393constructorimpl(i4 + 878077251);
    } while (i3 <= 32);
    i2 = i;
}
return iArr;
```

```
* JADX INFO: Access modifiers changed from: protected */
```

```
if (obj.length() != 40) {
    Toast.makeText(this$0, "序列号不正确", 0).show();
    return;
}
int[] m446constructorimpl = UIntArray.m446constructorimpl(10);
for (int i = 0; i < 40; i += 4) {
    UIntArray.m457setVXSXFK8(m446constructorimpl, i / 4, UInt.m393constructorimpl(UInt.m393constructorimpl(UInt.m393constructorimpl(UInt.m393constru
}
int[] m229encrypthkIa6DI = this$0.m229encrypthkIa6DI(m446constructorimpl);
UInt[] uIntArr = {UInt.m387boximpl(637666042), UInt.m387boximpl(457511012), UInt.m387boximpl(-2038734351), UInt.m387boximpl(578827205), UInt.m387box
int i2 = 0;
while (true) {
    int i3 = i2 + 1;
    if (uIntArr[i2].m444unboximpl() != UIntArray.m452getpVg5ArA(m229encrypthkIa6DI, i2)) {
        Toast.makeText(this$0, "序列号不正确", 0).show();
        return;
    } else if (i3 > 9) {
        alertDialog.dismiss();
        return;
    } else {
```

主要的代码逻辑

这里要定位到某个界面的包位置还可以用 adb shell dumpsys activity top 命令看当前界面包名，这个对于类特别多的程序有效，而且不仅仅可以看程序的入口点还可以看其他的界面。



其实这个题目的主程序代码逻辑很好理解，就是将数据进行一次 XTEA 加密。


```
hgame{d8c1d7d34573434ea8dfe5db40fbb25c0}请按任意键继续. .
```

这里注意一下大端序和小端序的问题就行。