checkin

打开所谓的`flag.txt`给了一串编码，`base64`解出来说要去玩弹球，输掉之后就出现了`flag`（貌似focus不在窗体上就不会一闪而过......）

flag{f5dfd0f5-0343-4642-8f28-9adbb74c4ede}

EzMachine

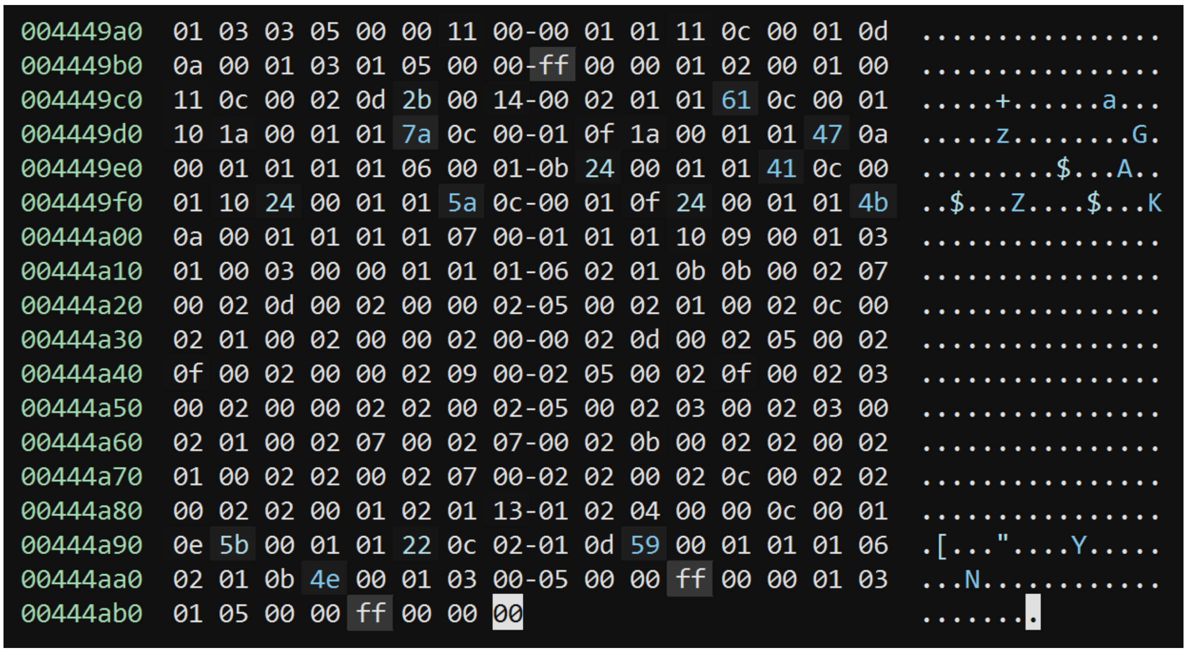
打开发现是个虚拟机，找到每个`opcode`对应的指令，大概的意思如下，不是特别精确

```
op=0:
index++

op=1 a,b:
index+=3
regs[a]=b

op=2 a:
index+=3
push a


op=3 a:
index+=3
push reg[a]

op=4 a:
index+=3
pop reg[a]

op=5:
index+=3
check:
reg[3]=0:
    right
    7c00(445ba8)
reg[3]=1:
    wrong
    7c00(445ba8)
reg[3]=3:
    input
    7c00(445ba8)
reg[3]=4:
    hacker

op=6 a,b:
index+=3
add reg[a],reg[b]

op=7 a,b:
index+=3
sub reg[a],reg[b]

op=8 a,b:
```

```
index+=3
imul reg[a],reg[b]


op=9 a,b:
index+=3
idiv reg[a],reg[b]
eax=>reg[0]
edx=>reg[1]

op=0xa a,b:
index+=3
xor reg[a],reg[b]

op=0xb a:
index=3*a-3

op=0xc a,b:
index+=3
reg[3]=reg[a]-reg[b]

op=0xd a:
if reg[3]!=0:
    index+=3
else:
    index=3*a-3

op=0xe a:
if reg[3]==0:
    index+=3
else:
    index=3*a-3

op=0xf a:
if reg[3]<=0:
    index+=3
else:
    index=3*a-3

op=0x10 a:
if reg[3]>=0:
    index+=3
else:
    index=3*a-3

op=0x11 :
index+=3
input flag
flag in *445ba8
length in reg[0]

op=0x12 a,b:
index+=3
2d90(*445ba8+a,0,b)

op=0x13 a,b:
index+=3
reg[a]=*(reg[b]+*445bd0+*ebp)
```

```
op=0x14 a,b:
index+=3
reg[a]=*(reg[b]+*445ba8)
# get char from input flag

op=0xff:
end
```

所有的`opcode`如下图



```
004449a0  01 03 03 05 00 00 11 00-00 01 01 11 0c 00 01 0d  ................
004449b0  0a 00 01 03 01 05 00 00-ff 00 00 01 02 00 01 00  ................
004449c0  11 0c 00 02 0d 2b 00 14-00 02 01 01 61 0c 00 01  .....+......a...
004449d0  10 1a 00 01 01 7a 0c 00-01 0f 1a 00 01 01 47 0a  .....z........G.
004449e0  00 01 01 01 01 06 00 01-0b 24 00 01 01 41 0c 00  .........$...A..
004449f0  01 10 24 00 01 01 5a 0c-00 01 0f 24 00 01 01 4b  ..$...Z....$...K
00444a00  0a 00 01 01 01 01 07 00-01 01 01 10 09 00 01 03  ................
00444a10  01 00 03 00 00 01 01 01-06 02 01 0b 0b 00 02 07  ................
00444a20  00 02 0d 00 02 00 00 02-05 00 02 01 00 02 0c 00  ................
00444a30  02 01 00 02 00 00 02 00-00 02 0d 00 02 05 00 02  ................
00444a40  0f 00 02 00 00 02 09 00-02 05 00 02 0f 00 02 03  ................
00444a50  00 02 00 00 02 02 00 02-05 00 02 03 00 02 03 00  ................
00444a60  02 01 00 02 07 00 02 07-00 02 0b 00 02 02 00 02  ................
00444a70  01 00 02 02 00 02 07 00-02 02 00 02 0c 00 02 02  ................
00444a80  00 02 02 00 01 02 01 13-01 02 04 00 00 0c 00 01  ................
00444a90  0e 5b 00 01 01 22 0c 02-01 0d 59 00 01 01 01 06  .[..."....Y.....
00444aa0  02 01 0b 4e 00 01 03 00-05 00 00 ff 00 00 01 03  ...N............
00444ab0  01 05 00 00 ff 00 00 00                          ........
```

不算很多，感觉直接看可能比写`decompiler`还要快，干脆粗略浏览一下，开始进行输入然后判断长度是0x11，看到右面的`a,z,A,Z`可以想到区分大小写进行处理，在其中找到`06`发现分别使用不同的值进行异或，然后分别自增和自减，然后除0x10，余数和得到的结果分别进“栈”，下面一长串`02`压进"栈"很多数据，刚好是0x22个，很容易想到和之前计算的结果是一一对应的关系，可以直接算出`flag`

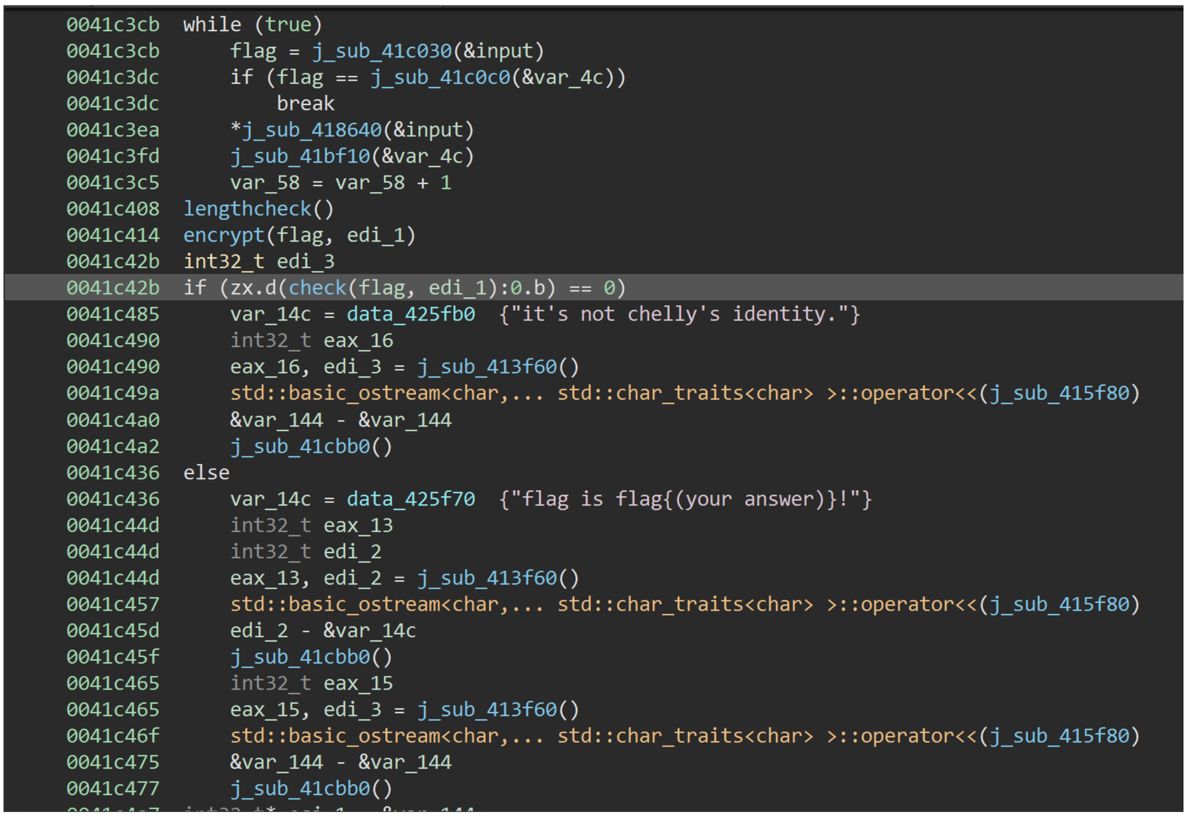
写`solution`的时候发现这两个用来异或的值恰到好处，把所有的情况刚好分开，可以直接根据算出来的值分类

```
target =
b"\x02\x07\x00\x02\r\x00\x02\x00\x00\x02\x05\x00\x02\x01\x00\x02\x0c\x00\x02\x01
\x00\x02\x00\x00\x02\x00\x00\x02\r\x00\x02\x05\x00\x02\x0f\x00\x02\x00\x00\x02\t
\x00\x02\x05\x00\x02\x0f\x00\x02\x03\x00\x02\x00\x00\x02\x02\x00\x02\x05\x00\x02
\x03\x00\x02\x03\x00\x02\x01\x00\x02\x07\x00\x02\x07\x00\x02\x0b\x00\x02\x02\x00
\x02\x01\x00\x02\x02\x00\x02\x07\x00\x02\x02\x00\x02\x0c\x00\x02\x02\x00\x02\x02
\x00"

# print(len(target))
tmp = []
for i in range(0, 0x22, 2):
    tmp.append(target[3 * i + 1] * 0x10 + target[3 * (i + 1) + 1])
flag = ""
# print(tmp)
tmp.reverse()
for i in tmp:
    if i <= 31:
        flag += chr((i + 1) ^ 0x4B)
    elif i <= 63:
```

```python
            flag += chr((i - 1) ^ 0x47)
        else:
            flag += chr(i)
    print(flag)
    # flag{Such_A_EZVM}
```

Chellys_identity

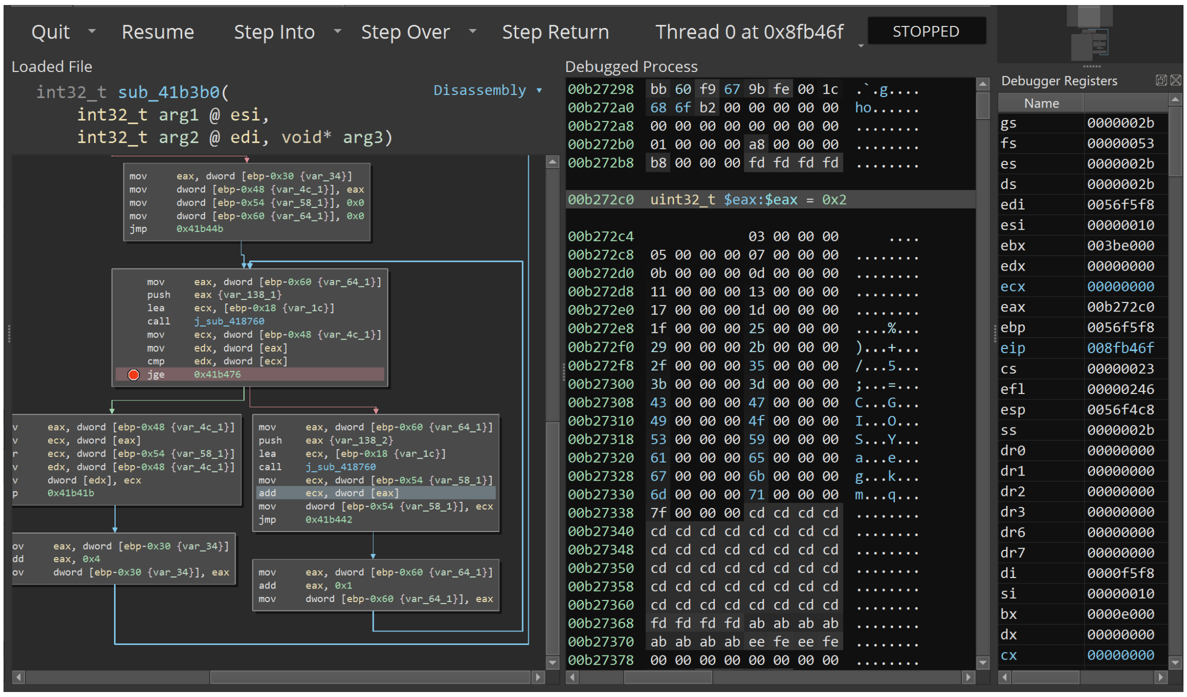
几乎压哨交了上去，在驱动题哪里浪费了太多时间，比赛结束了才知道有键盘扫描码这种东西......



```
0041c3cb  while (true)
0041c3cb      flag = j_sub_41c030(&input)
0041c3dc      if (flag == j_sub_41c0c0(&var_4c))
0041c3dc          break
0041c3ea      *j_sub_418640(&input)
0041c3fd      j_sub_41bf10(&var_4c)
0041c3c5      var_58 = var_58 + 1
0041c408  lengthcheck()
0041c414  encrypt(flag, edi_1)
0041c42b  int32_t edi_3
0041c42b  if (zx.d(check(flag, edi_1):0.b) == 0)
0041c485      var_14c = data_425fb0  {"it's not chelly's identity."}
0041c490      int32_t eax_16
0041c490      eax_16, edi_3 = j_sub_413f60()
0041c49a      std::basic_ostream<char,... std::char_traits<char> >::operator<<(j_sub_415f80)
0041c4a0      &var_144 - &var_144
0041c4a2      j_sub_41cbb0()
0041c436  else
0041c436      var_14c = data_425f70  {"flag is flag{(your answer)}!"}
0041c44d      int32_t eax_13
0041c44d      int32_t edi_2
0041c44d      eax_13, edi_2 = j_sub_413f60()
0041c457      std::basic_ostream<char,... std::char_traits<char> >::operator<<(j_sub_415f80)
0041c45d      edi_2 - &var_14c
0041c45f      j_sub_41cbb0()
0041c465      int32_t eax_15
0041c465      eax_15, edi_3 = j_sub_413f60()
0041c46f      std::basic_ostream<char,... std::char_traits<char> >::operator<<(j_sub_415f80)
0041c475      &var_144 - &var_144
0041c477      j_sub_41cbb0()
```

程序的主要处理就集中在这里，输入之后检查一下长度应该是16位，否则输出`bad long!`（奇怪的英语增加了），然后对输入进行一些变换，最后对比已有的数据

在处理flag的时候先生成了一组新的数据，通过调试发现是素数集

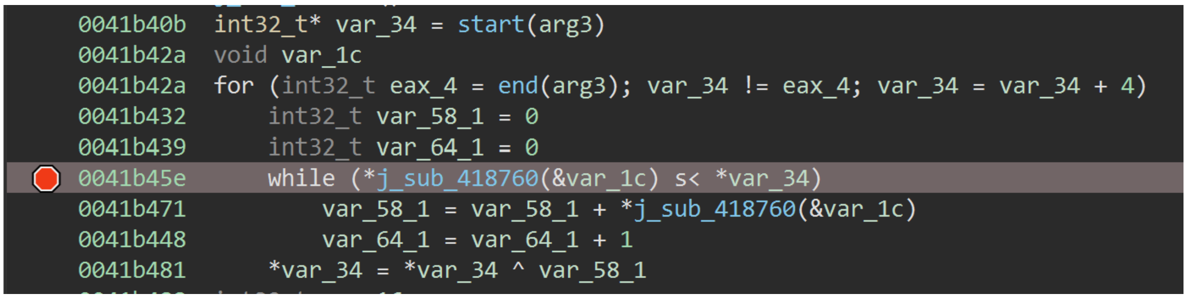


```
int32_t sub_41b3b0(
    int32_t arg1 @ esi,
    int32_t arg2 @ edi, void* arg3)

mov     eax, dword [ebp-0x30 {var_34}]
mov     dword [ebp-0x48 {var_4c_1}], eax
mov     dword [ebp-0x54 {var_58_1}], 0x0
mov     dword [ebp-0x60 {var_64_1}], 0x0
jmp     0x41b44b

mov     eax, dword [ebp-0x60 {var_64_1}]
push    eax {var_138_1}
lea     ecx, [ebp-0x18 {var_1c}]
call    j_sub_418760
mov     ecx, dword [ebp-0x48 {var_4c_1}]
mov     edx, dword [eax]
cmp     edx, dword [ecx]
jge     0x41b476

v       eax, dword [ebp-0x48 {var_4c_1}]
v       ecx, dword [eax]
r       ecx, dword [ebp-0x54 {var_58_1}]
v       edx, dword [ebp-0x48 {var_4c_1}]
v       dword [edx], ecx
p       0x41b41b

ov      eax, dword [ebp-0x30 {var_34}]
dd      eax, 0x4
ov      dword [ebp-0x30 {var_34}], eax

mov     eax, dword [ebp-0x60 {var_64_1}]
push    eax {var_138_2}
lea     ecx, [ebp-0x18 {var_1c}]
call    j_sub_418760
mov     ecx, dword [ebp-0x54 {var_58_1}]
add     ecx, dword [eax]
mov     dword [ebp-0x54 {var_58_1}], ecx
jmp     0x41b442

mov     eax, dword [ebp-0x60 {var_64_1}]
add     eax, 0x1
mov     dword [ebp-0x60 {var_64_1}], eax
```

```
00b27298  bb 60 f9 67 9b fe 00 1c  .`.g....
00b272a0  68 6f b2 00 00 00 00 00  ho......
00b272a8  00 00 00 00 00 00 00 00  ........
00b272b0  01 00 00 00 a8 00 00 00  ........
00b272b8  b8 00 00 00 fd fd fd fd  ........

00b272c0  uint32_t $eax:$eax = 0x2

00b272c4              03 00 00 00      ....
00b272c8  05 00 00 00 07 00 00 00  ........
00b272d0  0b 00 00 00 0d 00 00 00  ........
00b272d8  11 00 00 00 13 00 00 00  ........
00b272e0  17 00 00 00 1d 00 00 00  ........
00b272e8  1f 00 00 00 25 00 00 00  ....%...
00b272f0  29 00 00 00 2b 00 00 00  )...+...
00b272f8  2f 00 00 00 35 00 00 00  /...5...
00b27300  3b 00 00 00 3d 00 00 00  ;...=...
00b27308  43 00 00 00 47 00 00 00  C...G...
00b27310  49 00 00 00 4f 00 00 00  I...O...
00b27318  53 00 00 00 59 00 00 00  S...Y...
00b27320  61 00 00 00 65 00 00 00  a...e...
00b27328  67 00 00 00 6b 00 00 00  g...k...
00b27330  6d 00 00 00 71 00 00 00  m...q...
00b27338  7f 00 00 00 cd cd cd cd  ........
00b27340  cd cd cd cd cd cd cd cd  ........
00b27348  cd cd cd cd cd cd cd cd  ........
00b27350  cd cd cd cd cd cd cd cd  ........
00b27358  cd cd cd cd cd cd cd cd  ........
00b27360  cd cd cd cd cd cd cd cd  ........
00b27368  fd fd fd fd ab ab ab ab  ........
00b27370  ab ab ab ab ee fe ee fe  ........
00b27378  00 00 00 00 00 00 00 00  ........
```

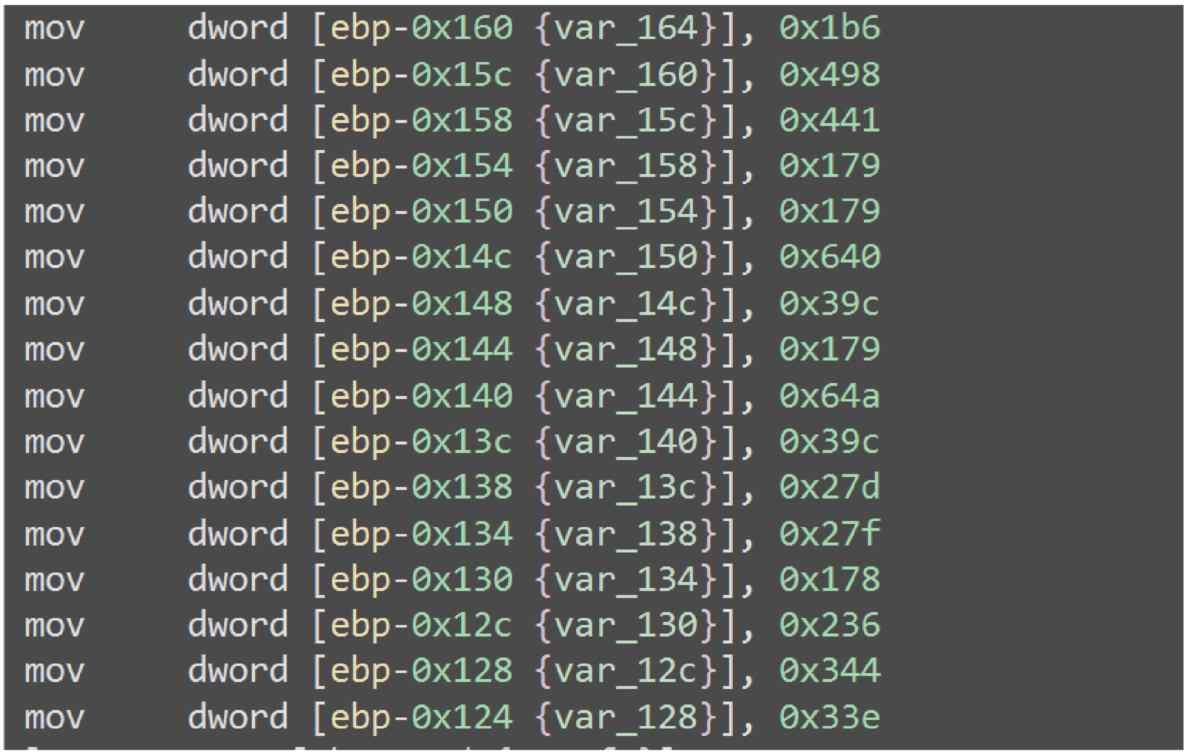
```
gs    0000002b
fs    00000053
es    0000002b
ds    0000002b
edi   0056f5f8
esi   00000010
ebx   003be000
edx   00000000
ecx   00000000
eax   00b272c0
ebp   0056f5f8
eip   008fb46f
cs    00000023
efl   00000246
esp   0056f4c8
ss    0000002b
dr0   00000000
dr1   00000000
dr2   00000000
dr3   00000000
dr6   00000000
dr7   00000000
di    0000f5f8
si    00000010
bx    0000e000
dx    00000000
cx    00000000
```

具体的算法就是异或，不过异或的值是小于原数的所有素数的和



```
0041b40b  int32_t* var_34 = start(arg3)
0041b42a  void var_1c
0041b42a  for (int32_t eax_4 = end(arg3); var_34 != eax_4; var_34 = var_34 + 4)
0041b432      int32_t var_58_1 = 0
0041b439      int32_t var_64_1 = 0
0041b45e      while (*j_sub_418760(&var_1c) s< *var_34)
0041b471          var_58_1 = var_58_1 + *j_sub_418760(&var_1c)
0041b448          var_64_1 = var_64_1 + 1
0041b481      *var_34 = *var_34 ^ var_58_1
```

最后对比的时候对比这些数据



```
mov     dword [ebp-0x160 {var_164}], 0x1b6
mov     dword [ebp-0x15c {var_160}], 0x498
mov     dword [ebp-0x158 {var_15c}], 0x441
mov     dword [ebp-0x154 {var_158}], 0x179
mov     dword [ebp-0x150 {var_154}], 0x179
mov     dword [ebp-0x14c {var_150}], 0x640
mov     dword [ebp-0x148 {var_14c}], 0x39c
mov     dword [ebp-0x144 {var_148}], 0x179
mov     dword [ebp-0x140 {var_144}], 0x64a
mov     dword [ebp-0x13c {var_140}], 0x39c
mov     dword [ebp-0x138 {var_13c}], 0x27d
mov     dword [ebp-0x134 {var_138}], 0x27f
mov     dword [ebp-0x130 {var_134}], 0x178
mov     dword [ebp-0x12c {var_130}], 0x236
mov     dword [ebp-0x128 {var_12c}], 0x344
mov     dword [ebp-0x124 {var_128}], 0x33e
```

直接逆不太好逆，干脆直接遍历

```python
def Num(num):
    value = 0
    for i in range(2, num):
        for j in range(2, i):
            if i % j == 0:
                break
        else:
            value += i
    return value


v = [438, 1176, 1089, 377, 377, 1600, 924, 377, 1610, 924, 637, 639, 376, 566,
836, 830]
for j in range(16):
    for i in range(255):
        if i ^ Num(i) == v[j]:
            print(chr(i), end='')
# Che11y_1s_EGOIST
```