reverse

check in

IDA打开，shift+f12打开字符串界面，看到flag

在010里面也可以找到

```
<---Welcome to m
oectf2022!--->.T
his challenge is
 very easy~.....
Input your flag,
 and I will chec
k for you~.Input
:.%s....moectf{E
njoy_yourself_in
_Reverse_Enginee
ring!!!}........
.Good job!!! ttt
ttqqqqqllllll!!!.
.QwQ. Something
wrong. Please tr
y again. >_<....
```

Hex

010打开，拖到文件尾，得到flag

```
app_type.     mo
ectf{Hello_Hex}
```

逆向工程之入门指北

begin

IDA打开，F5反编译

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char Str[108]; // [rsp+20h] [rbp-60h] BYREF
  int i; // [rsp+8Ch] [rbp+Ch]

  sub_4016D0(argc, argv, envp);
  puts("<---Welcome to moectf2022!--->");
  puts("Xor is very interesting and useful! You can learn it by various search engines.\n");
  printf("Input your flag, and I will check for you:");
  scanf("%s", Str);
  for ( i = 0; i < strlen(Str); ++i )
    Str[i] ^= 0x19u;
  if ( !strcmp(Str, Str2) )
    puts("\nGood job!!! You know how to decode my flag by xor!");
  else
    puts("\nQwQ. Something wrong. Please try again. >_<");
  return 0;
}
```

第11-12行对输入进行异或后与Str2比较，由于异或运算的对称性，直接对Str2每一项异或得到flag

```
>>> str2 = ['\x74','\x76','\x7c','\x7a','\x6d','\x7f'] + list('bA)kF(jFj)Fpwm*k*jmpw~88888d')
>>> for i in str2:
        print(chr(ord(i)^0x19),end="")


moectf{X0r_1s_s0_int3r3sting!!!!!}
```

Base

base64编码首先将A-Z、a-z、0-9和“+/”这64个可打印字符组成一张表。

将待加密数据进行处理，每个字符的ascii编码都有8个bit位，将这些比特位按照字符的顺序进行排列，每次取6个比特进行计算，得到的数作为上述表中的索引得到可打印字符串，最后的比特位不足6的倍数会补=号（对应bit位为0）。

IDA打开文件

```
__int64 __fastcall main()
{
  char a[50]; // [rsp+20h] [rbp-A0h] BYREF
  char inp[20]; // [rsp+60h] [rbp-60h] BYREF
  char de64[20]; // [rsp+80h] [rbp-40h] BYREF
  char base64[29]; // [rsp+A0h] [rbp-20h] BYREF

  _main();
  strcpy(base64, "1wX/yRrA4RfR2wj72Qv52x3L5qa=");
  text_46("Welcome to moectf,plz input your flag!\n");
  gets(inp);
  base64_decode(base64, de64);
  if ( !strcmp(de64, inp) )
    text_46("great!");
  else
    text_46("wrong!");
  gets(a);
  return 0i64;
}
```

虽然看到密文就在这里，但是用在线工具怎么都解不出来

进入base64_decode函数中，找到base64char（就是base64编码的那张索引表），再到数据区查看，发现A-Z被放在了后面

```
.rdata:00007FF660289000 61 62 63 64 65 66 67 68 69 6A+aAbcdefghijklmn db 'abcdefghijklmnopqrstuvwxyz0123456789+/ABCDEFGHIJKLMNOPQRSTUVWXYZ',0
.rdata:00007FF660289000 6B 6C 6D 6E 6F 70 71 72 73 74+                          ; DATA XREF: .data:base64char↑o
```

只好自己写脚本，得到flag

```
>>> import base64
>>> import string
>>> enc_str = '1wX/yRrA4RfR2wj72Qv52x3L5qa='
>>> string1 = 'abcdefghijklmnopqrstuvwxyz0123456789+/ABCDEFGHIJKLMNOPQRSTUVWXYZ'
>>> string2 = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
>>> base64.b64decode(enc_str.translate(str.maketrans(string1, string2)))
b'moectf{qwqbase_qwq}\x00'
```

直接调试也可以得到flag

```
1  __int64 __fastcall main()
2  {
3    char a[50]; // [rsp+20h] [rbp-A0h] BYREF
4    char inp[20]; // [rsp+60h] [rbp-60h] BYREF
5    char de64[20]; // [rsp+80h] [rbp-40h] BYREF
6    char base64[29]; // [rsp+A0h] [rbp-20h] BYREF
7
8    _main();
9    strcpy(base64, "1wX/yRrA4RfR2wj72Qv52x3L5qa=");
10   text_46("Welcome to moectf,plz input your flag!\n");
11   gets(inp);
12   base64_decode(base64, de64);
13   if ( !strcmp(de64, inp) )
14     text_46("great!");
15   else
16     text_46("wrong!");
17   gets(a);
18   return 0i64;
19 }
```

Str1: char de64[20]; // [rsp+80h] [rbp-40h] BYREF
"moectf{qwqbase_qwq}"

ezTea

这道题给了C源码，要求进行逆向

在所给pdf中可以得到密文，对encrypt函数倒着还原即可

另外这道题的输出部分，采用小端序(即对应数据的低字节端存储在低地址处)的方式输出，内层每次只会输出最低字节位然后右移8位

```
for (int j = 0; j < 2; j++) {
    for (int k = 0; k < 4; k++) {
        printf("%c", v[j] & 0xff);
        v[j] >>= 8;
    }
}
```

解密函数如下

```
void decrypt(uint32_t* v, uint32_t* k)
{
    uint32_t v0 = v[0], v1 = v[1], sum = 0;
    uint32_t delta = 0xd33b470;
    sum += delta * 32;
```

```
    for (int i = 0; i < 32; i++)
    {
        v1 -= ((v0 << 4) + k[2]) ^ (v0 + sum) ^ ((v0 >> 5) + k[3]);
        v0 -= ((v1 << 4) + k[0]) ^ (v1 + sum) ^ ((v1 >> 5) + k[1]);
        sum -= delta;
    }
    v[0] = v0;
    v[1] = v1;
}
```

将密文作为输入，即可得到明文

EquationPy

给了pyc文件 (pyc文件就是由Python文件经过编译后所生成的文件，py文件编译成pyc文件后加载速度更快而且提高了代码的安全性。---百度结果)

找反编译工具得到py源码（python反编译 - 在线工具 (tool.lu)）

```
print('Maybe z3 can help you solve this challenge.')
print('Now give me your flag, and I will check for you.')
flag = input('Input your flag:')
if len(flag) == 22 and ord(flag[0]) * 7072 + ord(flag[1]) * 2523 + ord(flag[2]) * 6714 + ord(flag[3]) * 8810 +
ord(flag[4]) * 6796 + ord(flag[5]) * 2647 + ord(flag[6]) * 1347 + ord(flag[7]) * 1289 + ord(flag[8]) * 8917 +
ord(flag[9]) * 2304 + ord(flag[10]) * 5001 + ord(flag[11]) * 2882 + ord(flag[12]) * 7232 + ord(flag[13]) * 3192 +
ord(flag[14]) * 9676 + ord(flag[15]) * 5436 + ord(flag[16]) * 4407 + ord(flag[17]) * 6269 + ord(flag[18]) * 9623 +
ord(flag[19]) * 6230 + ord(flag[20]) * 6292 + ord(flag[21]) * 57 == 10743134 and ord(flag[0]) * 3492 +
ord(flag[1]) * 1613 + ord(flag[2]) * 3234 + ord(flag[3]) * 5656 + ord(flag[4]) * 9182 + ord(flag[5]) * 4240 +
ord(flag[6]) * 8808 + ord(flag[7]) * 9484 + ord(flag[8]) * 4000 + ord(flag[9]) * 1475 + ord(flag[10]) * 2616 +
ord(flag[11]) * 2766 + ord(flag[12]) * 6822 + ord(flag[13]) * 1068 + ord(flag[14]) * 9768 + ord(flag[15]) * 1420 +
ord(flag[16]) * 4528 + ord(flag[17]) * 1031 + ord(flag[18]) * 8388 + ord(flag[19]) * 2029 + ord(flag[20]) * 2463 +
ord(flag[21]) * 32 == 9663091 and ord(flag[0]) * 9661 + ord(flag[1]) * 1108 + ord(flag[2]) * 2229 + ord(flag[3]) *
1256 + ord(flag[4]) * 7747 + ord(flag[5]) * 5775 + ord(flag[6]) * 5211 + ord(flag[7]) * 2387 + ord(flag[8]) * 1997
+ ord(flag[9]) * 4045 + ord(flag[10]) * 7102 + ord(flag[11]) * 7853 + ord(flag[12]) * 5596 + ord(flag[13]) * 6952
+ ord(flag[14]) * 8883 + ord(flag[15]) * 5125 + ord(flag[16]) * 9572 + ord(flag[17]) * 1149 + ord(flag[18]) * 7583
+ ord(flag[19]) * 1075 + ord(flag[20]) * 9804 + ord(flag[21]) * 72 == 10521461 and ord(flag[0]) * 4314 +
ord(flag[1]) * 3509 + ord(flag[2]) * 6200 + ord(flag[3]) * 5546 + ord(flag[4]) * 1705 + ord(flag[5]) * 9518 +
```

看到一堆条件，题目中有提示要用 z3 求解，然后百度z3是什么，可知是一种约束求解器，就是来解这道题的方程组的。

经过一番学习后写出求解脚本

```
from z3 import *

a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z = Reals('a b c d e f g h i j k l m n o p q r s t u v w x y z')
solver = Solver()
```

```
equ = [ a * 7072 + b * 2523 + c * 6714 + d * 8810 + e * 6796 + f * 2647 + g *
1347 + h * 1289 + i * 8917 + j * 2304 + k * 5001 + l * 2882 + m * 7232 + n *
3192 + o * 9676 + p * 5436 + q * 4407 + r * 6269 + s * 9623 + t * 6230 + u *
6292 + v * 57 == 10743134  , a * 3492 + b * 1613 + c * 3234 + d * 5656 + e *
9182 + f * 4240 + g * 8808 + h * 9484 + i * 4000 + j * 1475 + k * 2616 + l *
2766 + m * 6822 + n * 1068 + o * 9768 + p * 1420 + q * 4528 + r * 1031 + s *
8388 + t * 2029 + u * 2463 + v * 32 == 9663091  , a * 9661 + b * 1108 + c * 2229
+ d * 1256 + e * 7747 + f * 5775 + g * 5211 + h * 2387 + i * 1997 + j * 4045 + k
* 7102 + l * 7853 + m * 5596 + n * 6952 + o * 8883 + p * 5125 + q * 9572 + r *
1149 + s * 7583 + t * 1075 + u * 9804 + v * 72 == 10521461  , a * 4314 + b *
3509 + c * 6200 + d * 5546 + e * 1705 + f * 9518 + g * 2975 + h * 2689 + i *
2412 + j * 8659 + k * 5459 + l * 7572 + m * 3042 + n * 9701 + o * 4697 + p *
9863 + q * 1296 + r * 1278 + s * 5721 + t * 5116 + u * 4147 + v * 52 == 9714028
, a * 2310 + b * 1379 + c * 5900 + d * 4876 + e * 5329 + f * 6485 + g * 6610 + h
* 7179 + i * 7897 + j * 1094 + k * 4825 + l * 8101 + m * 9519 + n * 3048 + o *
3168 + p * 2775 + q * 4366 + r * 4066 + s * 7490 + t * 5533 + u * 2139 + v * 87
== 10030960  , a * 1549 + b * 8554 + c * 6510 + d * 6559 + e * 5570 + f * 1003 +
g * 8562 + h * 6793 + i * 3509 + j * 4965 + k * 6111 + l * 1229 + m * 5654 + n *
2204 + o * 2217 + p * 5039 + q * 5657 + r * 9426 + s * 7604 + t * 5883 + u *
5285 + v * 17 == 10946682  , a * 2678 + b * 4369 + c * 7509 + d * 1564 + e *
7777 + f * 2271 + g * 9696 + h * 3874 + i * 2212 + j * 6764 + k * 5727 + l *
5971 + m * 5876 + n * 9959 + o * 4604 + p * 8461 + q * 2350 + r * 3564 + s *
1831 + t * 6088 + u * 4575 + v * 9 == 10286414  , a * 8916 + b * 8647 + c * 4522
+ d * 3579 + e * 5319 + f * 9124 + g * 9535 + h * 5125 + i * 3235 + j * 3246 + k
* 3378 + l * 9221 + m * 1875 + n * 1008 + o * 6262 + p * 1524 + q * 8851 + r *
4367 + s * 7628 + t * 9404 + u * 2065 + v * 9 == 11809388  , a * 9781 + b * 9174
+ c * 3771 + d * 6972 + e * 6425 + f * 7631 + g * 8864 + h * 9117 + i * 4328 + j
* 3919 + k * 6517 + l * 7165 + m * 6895 + n * 3609 + o * 3878 + p * 1593 + q *
9098 + r * 6432 + s * 2584 + t * 8403 + u * 4029 + v * 30 == 13060508  , a *
2511 + b * 8583 + c * 2428 + d * 9439 + e * 3662 + f * 3278 + g * 8305 + h *
1100 + i * 7972 + j * 8510 + k * 8552 + l * 9993 + m * 6855 + n * 1702 + o *
1640 + p * 3787 + q * 8161 + r * 2110 + s * 5320 + t * 3313 + u * 9286 + v * 74
== 10568195  , a * 4974 + b * 4445 + c * 7368 + d * 9132 + e * 5894 + f * 7822 +
g * 7923 + h * 6822 + i * 2698 + j * 3643 + k * 8392 + l * 4126 + m * 1941 + n *
6641 + o * 2949 + p * 7405 + q * 9980 + r * 6349 + s * 3328 + t * 8766 + u *
9508 + v * 65 == 12514783  , a * 4127 + b * 4703 + c * 6409 + d * 4907 + e *
5230 + f * 3371 + g * 5666 + h * 3194 + i * 5448 + j * 8415 + k * 4525 + l *
4152 + m * 1467 + n * 5254 + o * 2256 + p * 1643 + q * 9113 + r * 8805 + s *
4315 + t * 8371 + u * 1919 + v * 2 == 10299950  , a * 6245 + b * 8783 + c * 6059
+ d * 9375 + e * 9253 + f * 1974 + g * 8867 + h * 6423 + i * 2577 + j * 6613 + k
* 2040 + l * 2209 + m * 4147 + n * 7151 + o * 1011 + p * 9446 + q * 4362 + r *
3073 + s * 3006 + t * 5499 + u * 8850 + v * 23 == 11180727  , a * 1907 + b *
9038 + c * 3932 + d * 7054 + e * 1135 + f * 5095 + g * 6962 + h * 6481 + i *
7049 + j * 5995 + k * 6233 + l * 1321 + m * 4455 + n * 8181 + o * 5757 + p *
6953 + q * 3167 + r * 5508 + s * 4602 + t * 1420 + u * 3075 + v * 25 == 10167536
, a * 1489 + b * 9236 + c * 7398 + d * 4088 + e * 4131 + f * 1657 + g * 9068 + h
* 6420 + i * 3970 + j * 3265 + k * 5343 + l * 5386 + m * 2583 + n * 2813 + o *
7181 + p * 9116 + q * 4836 + r * 6917 + s * 1123 + t * 7276 + u * 2257 + v * 65
== 10202212  , a * 2097 + b * 1253 + c * 1469 + d * 2731 + e * 9565 + f * 9185 +
g * 1095 + h * 8666 + i * 2919 + j * 7962 + k * 1497 + l * 6642 + m * 4108 + n *
6892 + o * 7161 + p * 7552 + q * 5666 + r * 4060 + s * 7799 + t * 5080 + u *
8516 + v * 43 == 10435786  , a * 1461 + b * 1676 + c * 4755 + d * 7982 + e *
3860 + f * 1067 + g * 6715 + h * 4019 + i * 4983 + j * 2031 + k * 1173 + l *
2241 + m * 2594 + n * 8672 + o * 4810 + p * 7963 + q * 7749 + r * 5730 + s *
9855 + t * 5858 + u * 2349 + v * 71 == 9526385  , a * 9025 + b * 9536 + c * 1515
```

```
+ d * 8177 + e * 6109 + f * 4856 + g * 6692 + h * 4929 + i * 1010 + j * 3995 + k
* 3511 + l * 5910 + m * 3501 + n * 3731 + o * 6601 + p * 6200 + q * 8177 + r *
5488 + s * 5957 + t * 9661 + u * 4956 + v * 48 == 11822714  , a * 4462 + b *
1940 + c * 5956 + d * 4965 + e * 9268 + f * 9627 + g * 3564 + h * 5417 + i *
2039 + j * 7269 + k * 9667 + l * 4158 + m * 2856 + n * 2851 + o * 9696 + p *
5986 + q * 6237 + r * 5845 + s * 5467 + t * 5227 + u * 4771 + v * 72 == 11486796
, a * 4618 + b * 8621 + c * 8144 + d * 7115 + e * 1577 + f * 8602 + g * 3886 + h
* 3712 + i * 1258 + j * 7063 + k * 1872 + l * 9855 + m * 4167 + n * 7615 + o *
6298 + p * 7682 + q * 8795 + r * 3856 + s * 6217 + t * 5764 + u * 5076 + v * 93
== 11540145  , a * 7466 + b * 8442 + c * 4822 + d * 7639 + e * 2049 + f * 7311 +
g * 5816 + h * 8433 + i * 5905 + j * 4838 + k * 1251 + l * 8184 + m * 6465 + n *
4634 + o * 5513 + p * 3160 + q * 6720 + r * 9205 + s * 6671 + t * 7716 + u *
1905 + v * 29 == 12227250  , a * 5926 + b * 9095 + c * 2048 + d * 4639 + e *
3035 + f * 9560 + g * 1591 + h * 2392 + i * 1812 + j * 6732 + k * 9454 + l *
8175 + m * 7346 + n * 6333 + o * 9812 + p * 2034 + q * 6634 + r * 1762 + s *
7058 + t * 3524 + u * 7462 + v * 11 == 11118093 ]
for i in equ:
    solver.add(i)
if solver.check() == sat:
    result = solver.model()
print(result)
```

D flat

看到提示之后才逐渐会做。。。

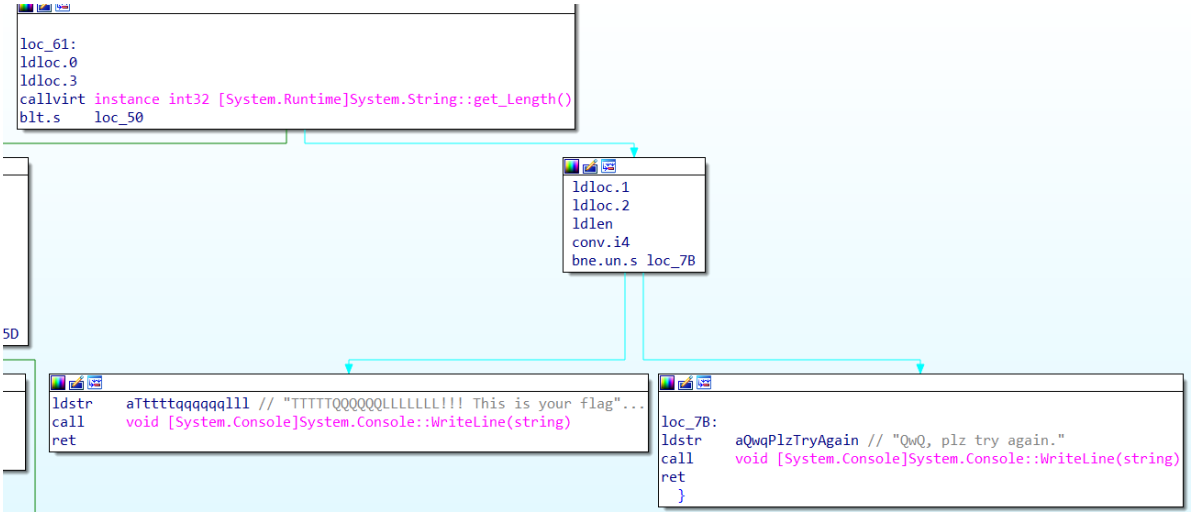
IDA打开exe文件，里面什么都没有，查字符串也没有应该有的字符

那肯定就是在dll里面，dll是程序运行过程中的动态链接库，便于将程序划分为不同模块，方便修改，且可以多个程序共享。

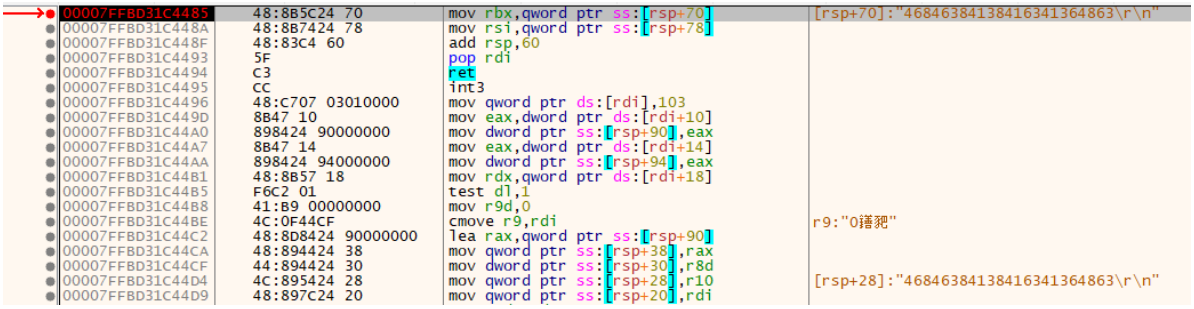
IDA载入dll文件，看到的熟悉的moectf字符串

```
ldtoken  valuetype __StaticArrayInitTypeSize=108 <PrivateImplementationDetails>::B9C5F381FBFC7DEC1C410EDB39E7AC7B11CC4F4E1FDCC874346BD16E7501BB81
call     void [System.Runtime]System.Runtime.CompilerServices.RuntimeHelpers::InitializeArray(class [System.Runtime]System.Array, valuetype [System.Runtime]System.F
stloc.2
ldstr    aInMusicTheoryT // "In music theory, there is a note that h"...
call     void [System.Console]System.Console::WriteLine(string)
ldstr    aDoYouKnowItNow // "Do you know it?\nNow plz input your fla"...
call     void [System.Console]System.Console::WriteLine(string)
call     string [System.Console]System.Console::ReadLine()
stloc.3
```

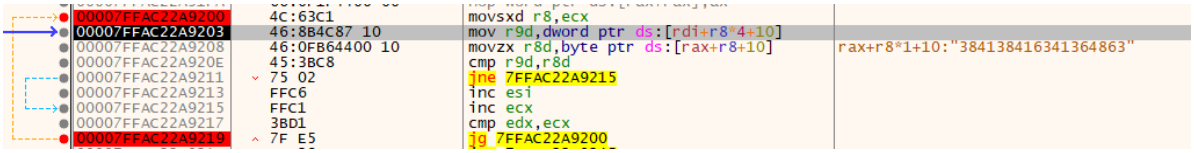
再往下看，看到了判断的位置，但是还是没有具体逻辑



用x64dbg调试下试试吧，一直运行直到输入flag部分，先随便输下，此时dll已经载入，现在就是dll里面的逻辑，f8持续单步

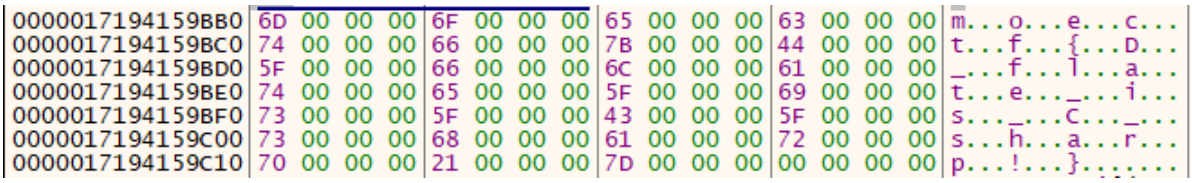


当到达某个位置时会进入循环，且注释位置每次循环都会减少一位，这里就是flag验证的部分，会把flag和输入分别取出到r9和r8进行比较



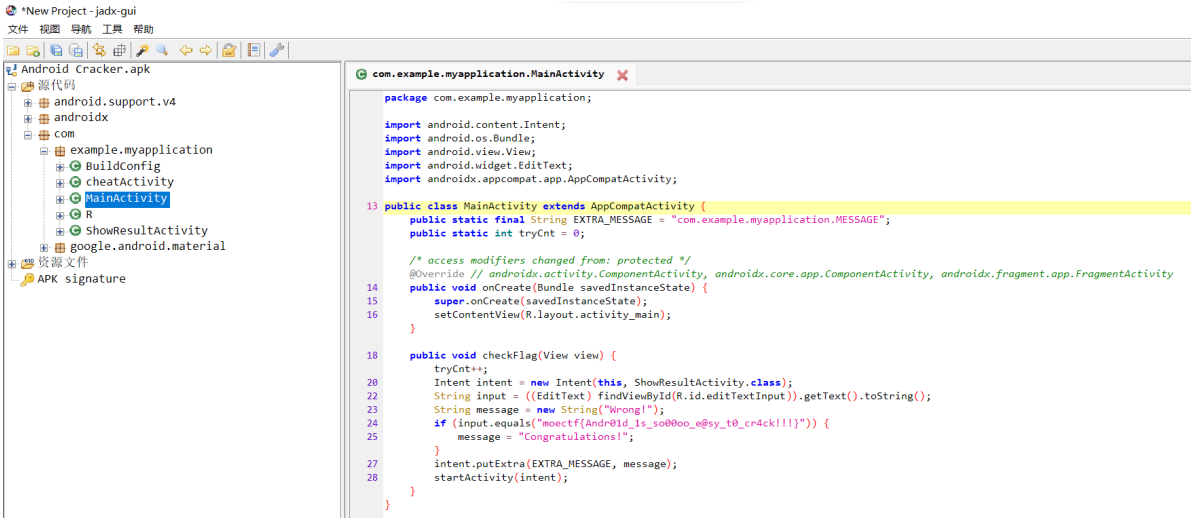
其中[rax+r8+10]指向输入部分，[rdi+r8*4+10]指向flag部分

在内存窗口中转到flag位置



Android Cracker

apk逆向，先下个jadx，找到main函数就找到了flag



chicken soup

IDA打开

```
.text:00401009 74 03                    jz      short near ptr loc_40100D+1
.text:00401009
.text:0040100B 75 01                    jnz     short near ptr loc_40100D+1
.text:0040100B
.text:0040100D
.text:0040100D                          loc_40100D:                     ; CODE XREF: .text:00401009↑j
.text:0040100D                                                          ; .text:0040100B↑j
.text:0040100D E9 C7 45 F8 00           jmp     near ptr 13855D9h
.text:0040100D
.text:0040100D                          ; ---------------------------------------------------------------------------
.text:00401012 00 00                    align 4
.text:00401014 00 EB 09 8B 45 F8 83 C0 01 89+dd 8B09EB00h, 0C083F845h, 0F8458901h, 89084D8Bh, 558BF44Dh, 1C283F4h, 8BF055
.text:00401014 45 F8 8B 4D 08 89 4D F4 8B 55+dd 8001F445h, 7500FF7Dh, 0F4558BEEh, 89F0552Bh, 458BEC55h, 1E883ECh, 73F8453
.text:00401014 F4 83 C2 01 89 55 F0 8B 45 F4+dd 8B0151B6h, 45030845h, 8B60FF8h, 558BCA03h, 0F8550308h, 0A3EB0A88h, 8B5B5E
.text:00401080                          ; ---------------------------------------------------------------------------
.text:00401080
.text:00401080                          loc_401080:                     ; CODE XREF: _main+91↓p
.text:00401080 55                       push    ebp
.text:00401081 8B EC                    mov     ebp, esp
.text:00401083 83 EC 14                 sub     esp, 14h
.text:00401086 53                       push    ebx
.text:00401087 56                       push    esi
.text:00401088 57                       push    edi
.text:00401089 74 03                    jz      short near ptr loc_40108D+1
.text:00401089
.text:0040108B 75 01                    jnz     short near ptr loc_40108D+1
.text:0040108B
.text:0040108D
.text:0040108D                          loc_40108D:                     ; CODE XREF: .text:00401089↑j
.text:0040108D                                                          ; .text:0040108B↑j
.text:0040108D E9 C7 45 F8 00           jmp     near ptr 1385659h
.text:0040108D
```

重要加密函数两片飘红，程序中插入了E9，导致后面的机器码被认为是其操作数，将其改为90(nop)即可。

修改后看到第一个函数是将每个字节都加上下一个字节。

```
1  unsigned int __cdecl sub_401000(const char *a1)
2  {
3    unsigned int result; // eax
4    unsigned int i; // [esp+18h] [ebp-8h]
5
6    for ( i = 0; ; ++i )
7    {
8      result = strlen(a1) - 1;
9      if ( i >= result )
10       break;
11     a1[i] += a1[i + 1];
12   }
13   return result;
14 }
```

第二个函数是将每个字节的16倍（考虑溢出）和该字节的高四个比特位进行或运算。

```
1  unsigned int __cdecl sub_401080(const char *a1)
2  {
3    unsigned int result; // eax
4    unsigned int i; // [esp+18h] [ebp-8h]
5
6    for ( i = 0; ; ++i )
7    {
8      result = i;
9      if ( i >= strlen(a1) )
10       break;
11     a1[i] = (16 * a1[i]) | ((int)(unsigned __int8)a1[i] >> 4);
12   }
13   return result;
14 }
```

解密脚本如下

```
int main()
{
    char key[] = {
0xcd,0x4d,0x8c,0x7d,0xad,0x1e,0xbe,0x4a,0x8a,0x7d,0xbc,0x7c,0xfc,0x2e,0x2a,0x79,
0x9d,0x6a,0x1a,0xcc,0x3d,0x4a,0xf8,0x3c,0x79,0x69,0x39,0xd9,0xdd,0x9d,0xa9,0x69,
0x4c,0x8c,0xdd,0x59,0xe9,0xd7,0x00};
    int len = strlen(key);

    char result[39] =
{'m','o',0x8c,0x7d,0xad,0x1e,0xbe,0x4a,0x8a,0x7d,0xbc,0x7c,0xfc,0x2e,0x2a,0x79,0
x9d,0x6a,0x1a,0xcc,0x3d,0x4a,0xf8,0x3c,0x79,0x69,0x39,0xd9,0xdd,0x9d,0xa9,0x69,0
x4c,0x8c,0xdd,0x59,0xe9,0xd7,0x00};
    for (int i = 0; i < 38; i++)
    {
        for (int j = 0x20; j < 0xff; j++)
        {
            char x = (result[i] + j) * 16;
            char y = (int)(unsigned __int8)(result[i] + j) >> 4;
            if ((x | y) == key[i])
            {
                result[i + 1] = j;
            }
        }
    }
    puts(result);
    return 0;
}
```

fake key

IDA打开，f5反编译

```
11  puts("I changed the key secretly, you can't find the right key!");
12  puts("And I use random numbers to rot my input, you can never guess them!");
13  puts("Unless you debug to get the key and random numbers...");
14  puts("Now give me your flag:");
15  scanf("%s", Str);
16  v5 = strlen(Str);
17  for ( i = 0; i < v5; ++i )
18    Str[i] ^= ::Str[i % v6];
19  for ( j = 0; j < v5; ++j )
20    Str[j] += rand() % 10;
21  if ( (unsigned int)sub_4015A2(Str, &unk_403020) )
22    puts("\nRight! TTTTTQQQQQLLLLL!!!");
23  else
24    puts("QwQ, plz try again.");
```

看到加密部分，同时还有提示，调试可以看到key和rand()产生的随机数，但该题没有随机种子，所以直接rand()和题目中产生的值相同。

```
ata:0000000000403040 Str db 'yunzh1junT'          ; DATA XREF: sub_401550+5↑o
ata:0000000000403040                              ; sub_401550+2A↑o
ata:0000000000403040                              ; main+10↑o
ata:0000000000403040                              ; main+8F↑o
ata:000000000040304A db  43h ; C
ata:000000000040304B db  4Ch ; L
ata:000000000040304C db  2Ch ; ,
ata:000000000040304D db  74h ; t
ata:000000000040304E db  72h ; r
ata:000000000040304F db  61h ; a
ata:0000000000403050 db  63h ; c
ata:0000000000403051 db  6Bh ; k
ata:0000000000403052 db  59h ; Y
ata:0000000000403053 db  59h ; Y
ata:0000000000403054 db  44h ; D
ata:0000000000403055 db  53h ; S
```

脚本如下

```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#include<string.h>

int main()
{
    char des[] = {
0x15,0x21,0xf,0x19,0x25,0x5b,0x19,0x39,0x5f,0x3a,0x3b,0x30,0x74,0x7,0x43,0x3f,0x
9,0x5a,0x34,0xc,0x74,0x3f,0x1e,0x2d,0x27,0x21,0x12,0x16,0x1f,0x00};
    char key[] = "yunzh1junTCL,trackYYDS";

    int len = strlen(des);

    for (int i = 0; i < len; i++)
        des[i] -= rand() % 10;
    for (int i = 0; i < len; i++)
        des[i] ^= key[i % 22];

    puts(des);
    return 0;
}
```

EzRisc-V

IDA打不开，提示没有risc-v相关脚本

百度搜索IDA相关risc-v脚本，github有一个py脚本，按照提示配置IDA

再次尝试打开

```
# public main
main:

var_s0=  0
var_s8=  8
arg_0=  10h

addi            sp, sp, -70h # Alternative name is '$x'
sd              ra, 60h+var_s8(sp)
sd              s0, 60h+var_s0(sp)
addi            s0, sp, 60h+arg_0
li              a5, unk_58508
ld              a1, 0(a5)
ld              a2, 8(a5)
ld              a3, 10h(a5)
ld              a4, 18h(a5)
sd              a1, -70h(s0)
sd              a2, -68h(s0)
sd              a3, -60h(s0)
sd              a4, -58h(s0)
lw              a4, 20h(a5)
sw              a4, -50h(s0)
lhu             a4, 24h(a5)
sh              a4, -4Ch(s0)
lbu             a5, 26h(a5)
sb              a5, -4Ah(s0)
lui             a5, 58h # 'X'
addi            a0, a5, 480h
jal             _IO_puts
lui             a5, 58h # 'X'
addi            a0, a5, 498h
jal             _IO_puts
addi            a5, s0, -48h
mv              a1, a5
lui             a5, 58h # 'X'
addi            a0, a5, 4B0h
jal             __isoc99_scanf
sw              zero, -14h(s0)
j               loc_106B6
```

main函数看不懂，跟着字符串也找不到该有的东西。

看群友水群说还有个ghidra，于是安装之。

用ghidra打开，找到main函数，如愿看到了伪代码

```
puts("Welcome to moeCTF 2022");
puts("Plz input your flag:");
__isoc99_scanf(&DAT_000584b0,abStack72);
local_14 = 0;
while( true ) {
  if (0x26 < local_14) {
    printf("congratulations!!! you are right");
    gp = (undefined1 *)((longlong)_dl_static_dtv + 0x1c8);
    return 0;
  }
  if ((abStack72[local_14] ^ 0x39) != *(byte *)((longlong)&local_70 + (longlong)local_14)) bre...
  ;
  local_14 = local_14 + 1;
}
printf("ops, wrong input!\nPlease try again");
```

可以看到只是简单的异或，脚本如下

```c
#include<stdio.h>

int main()
{
    char var[] = {
0x54,0x56,0x5c,0x5a,0x4d,0x5f,0x42,0x4b,0x08,0x4a,0x5a,0x14,0x4f,0x66,0x08,0x4a,
0x66,0x4a,0x56,0x09,0x09,0x56,0x56,0x66,0x08,0x57,0x4d,0x5c,0x4b,0x5c,0x4a,0x4d,
0x08,0x57,0x00,0x18,0x18,0x18,0x44,0x00};
    for (int i = 0; i<40; i++)
    {
        printf("%c", var[i] ^ 0x39);
    }

    return 0;
}
```

fake code

这个pdf完全就是在教怎么做这道题，找到对应位置后直接看反汇编就可以知道算法

程序在运行中出现了除零异常，因此跳转到了except异常处理

该位置汇编如下，dword_7FF667445000就是你的输入

```
.text:00007FF6674411E9
.text:00007FF6674411E9 loc_7FF6674411E9:
.text:00007FF6674411E9 ;   __except(loc_7FF6674420D0) // owned by 7FF6674411B8
.text:00007FF6674411E9 imul    eax, cs:dword_7FF667445000, 61h ; 'a'
.text:00007FF6674411F0 add     eax, 65h ; 'e'
.text:00007FF6674411F3 cdq
.text:00007FF6674411F4 mov     ecx, 0E9h
.text:00007FF6674411F9 idiv    ecx
.text:00007FF6674411FB mov     eax, edx
.text:00007FF6674411FD mov     cs:dword_7FF667445000, eax
.text:00007FF667441203 mov     eax, cs:dword_7FF667445000
.text:00007FF667441209 xor     eax, 29h
.text:00007FF66744120C mov     cs:dword_7FF667445000, eax
```

转成c就是　　((0x65 + (n * 0x61)) % 0xe9)^0x29

回到主程序

```
 9  puts("Can you read my assembly in exception?");
10  puts("Give me your flag:");
11  sub_7FF667441290("%s", v7);
12  v6 = -1i64;
13  do
14    ++v6;
15  while ( v7[v6] );
16  if ( v6 == 51 )
17  {
18    for ( i = 0; i < 51; ++i )
19    {
20      v5 = (127 * v5 + 102) % 255;
21      v7[i] ^= byte_7FF667445010[dword_7FF667445000];
22    }
23    if ( (unsigned int)sub_7FF667441020(&unk_7FF667445110, v7) )
24      puts("\nTTTTTTTTTTQQQQQQQQQQQQQLLLLLLLLL!!!!");
25    else
26      puts("\nQwQ, please try again.");
27    return 0;
28  }
29  else
```

上面的汇编计算的是 byte_7FF667445010 数组的下标，因为是异或运算，所以直接将 byte_7FF667445010 和 sub_7FF667445110 异或即可得到flag （调试时发现有些时候不会进入except，但是不知道是哪些情况，就一个一个试出来了，发现在 i == 4*k + 2 时不会进入except，也就是下标不变）

脚本如下

```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#include<string.h>

int main()
{
    char key[] = {
0x1e,0x70,0x7a,0x6e,0xea,0x83,0x9e,0xef,0x96,0xe2,0xb2,0xd5,0x99,0xbb,0xbb,0x78,
0xb9,0x3d,0x6e,0x38,0x42,0xc2,0x86,0xff,0x63,0xbd,0xfa,0x79,0xa3,0x6d,0x60,0x94,
0xb3,0x42,0x11,0xc3,0x90,0x89,0xbd,0xef,0xd4,0x97,0xf8,0x7b,0x8b,0xb,0x2d,0x75,0
x7e,0xdd,0xcb,0x00};
    unsigned int n = 0x19;
    char array[] = { 0xAC , 0x4, 0x58 , 0xB0 , 0x45 , 0x96 , 0x9F , 0x2E , 0x41
, 0x15 , 0x18 ,
                    0x29 , 0x0B1 , 0x33 , 0x0AA , 0x12 , 0x0D , 0x89 , 0x0E6 ,
0x0FA , 0x0F3 ,
                    0x0C4 , 0x0BD , 0x0E7 , 0x70 , 0x8A , 0x94 , 0x0C1 , 0x85 ,
0x9D , 0x0A3 ,
                    0x0F2 , 0x3F , 0x82 , 0x8E , 0x0D7 , 0x3, 0x93 , 0x3D , 0x13
, 0x5, 0x6B ,
                    0x41 , 0x3, 0x96 , 0x76 , 0x0E3 , 0x0B1 , 0x8A , 0x4A , 0x22
, 0x55 , 0x0C4 ,
                    0x19 , 0x0F5 , 0x55 , 0x0A6 , 0x1F , 0x0E , 0x61 , 0x27 ,
0x0CB , 0x1F ,
                    0x9E , 0x5A , 0x7A , 0x0E3 , 0x15 , 0x40 , 0x94 , 0x47 ,
0x0DE , 0x0, 0x1,
                    0x91 , 0x66 , 0x0B7 , 0x0CD , 0x22 , 0x64 , 0x0F5 , 0x0A5 ,
0x9C , 0x68 ,
```

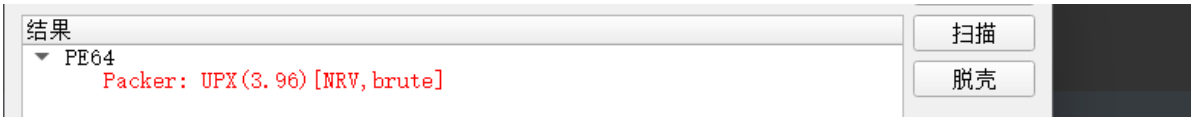
```
                    0x0A5 , 0x52 , 0x86 , 0x0BD , 0x0B0 , 0x0DD , 0x76 , 0x28 ,
0x0AB , 0x16 ,
                    0x95 , 0x0C5 , 0x26 , 0x2C , 0x0F6 , 0x39 , 0x0BE , 0x0,
0x0A5 , 0x0AD ,
                    0x0E3 , 0x93 , 0x9E , 0x0E3 , 0x5, 0x0A0 , 0x0B0 , 0x1D ,
0x0B0 , 0x16 ,
                    0x0B , 0x5B , 0x33 , 0x95 , 0x0A4 , 0x9, 0x16 , 0x87 , 0x56
, 0x1F , 0x83 ,
                    0x4E , 0x4A , 0x3C , 0x55 , 0x36 , 0x6F , 0x0BB , 0x4C ,
0x4B , 0x9D , 0x0B1 ,
                    0x0AE , 0x0E5 , 0x8E , 0x0C8 , 0x0FB , 0x0E , 0x29 , 0x8A ,
0x0BB , 0x0FC ,
                    0x20 , 0x62 , 0x4, 0x2D , 0x80 , 0x61 , 0x0D6 , 0x0C1 ,
0x0CC , 0x3B , 0x89 ,
                    0x0C5 , 0x8B , 0x0D5 , 0x26 , 0x58 , 0x0D6 , 0x0B6 , 0xA0 ,
0x50 , 0x75 ,
                    0x0AB , 0x17 , 0x83 , 0x7F , 0x37 , 0x2B , 0x0A0 , 0x1D ,
0x2C , 0x0CF ,
                    0x0C7 , 0x0E0 , 0x0E5 , 0x49 , 0x0C9 , 0x0FA , 0x6B , 0x0C0
, 0x98 , 0x66 ,
                    0x99 , 0x92 , 0x0, 0x2, 0x0D4 , 0x75 , 0x46 , 0x22 , 0x5,
0x35 , 0x0D1 , 0x4B ,
                    0x0C5 , 0x0AD , 0x0E0 , 0x8E , 0x45 , 0x3B , 0x50 , 0x15 ,
0x0B5 , 0x2E ,
                    0x85 , 0x30 , 0x89 , 0x54 , 0x12 , 0x0DE , 0x0F1 , 0x5A ,
0x0F0 , 0x2B ,
                    0x0A7 , 0x1B , 0x4A , 0x26 , 0x5D , 0x98 , 0x0D4 , 0x0A1 ,
0x0BE , 0x0D1 ,
                    0x4D , 0x7E , 0x38 , 0xDE , 0x0B , 0x0A , 0x54 , 0x0B8 ,
0x73 , 0x6D ,
                    0x0AD , 0x8C , 0x1E , 0x0D9 , 0x31 , 0x5F , 0x56 , 0x7E ,
0x0BD , 0x48 ,
                    0x32 , 0x98 , 0x2E , 0x3E , 0x0EB , 0x0A2 , 0x1D };

    for (int i = 0; key[i]; i++)
    {
        if(i!=2&&i!=6&&i!=10&&i!=14&&i!=18&&i!=22&& i != 26 && i != 30 && i !=
34 && i != 38 && i != 42 && i != 46 &&i!=50)
            n = ((0x65 + (n * 0x61)) % 0xe9)^0x29;
        for (int j = 0x20; j < 0x80; j++)
        {
            if (j == (key[i]^array[n]))
            {
                printf("%c", j);
            }
        }
    }

    return 0;
}
```

Art

程序加壳，UPX



直接脱

也可以手动脱，但是我是懒狗

分析脱壳后的程序

```
  puts("Do you know UPX???");
  puts("Oh no...Something seems to be wrong...My equations has multiple solutions...");
  puts("May be I can check it by a hash algorithm. You can never reverse it!!!");
  printf("Input your flag:");
  scanf("%s", Str1);
  for ( i = 0; i <= 27; ++i )
    v4[i] = Str1[i];
  for ( i = 1; i <= 27; ++i )
    Str1[i - 1] ^= (Str1[i - 1] % 17 + Str1[i]) ^ 0x19;
  if ( !strcmp(Str1, &Str2) && (unsigned int)sub_401550(v4) )
    puts("\nGood job!!! You know UPX and hash!!!");
  else
    puts("\nQwQ. Something wrong. Please try again. >_<");
  return 0;
```

sub_401550应该就是hash校验吧（或许是吧），那具体的加密就只有第二个for了

在写脚本的过程中发现每个值会对应多个结果，写不出一次性脚本，就只能一个一个试了

试的脚本如下

```
char key[] =
{2,0x18,0xf,0xf8,0x19,4,0x27,0xd8,0xe8,0x0,0x35,0x48,0x4d,0x2a,0x45,0x6b,0x59,0x
2e,0x43,1,0x18,0x5c,9,9,9,9,0xb5,0x7d };

        for (int j = 0x20; j < 0x80; j++)
        {
            if ((j ^ ((j % 17 + '}') ^ 0x19)) == 0xb5)
            {
                printf("%c\n", j);
            }
        }
```

我是从后面往前面试的，每次替换下 '}' 和 0xb5 就行了

broken hash

ida打开看到

```
 9  v7 = (void (*)(void))sub_140001BE0;
10  if ( dword_1400053F0 )
11    v7 = (void (*)(void))sub_1400010A0;
12  puts("This is a surprise!");
13  sub_140001E80("Give me your flag: ");
14  sub_140001F00("%s", v8);
15  v6 = -1i64;
16  do
17    ++v6;
18  while ( v8[v6] );
19  if ( v6 == 88 )
20  {
21    sub_1400010C0(v8);
22    for ( i = 0; i < 88; ++i )
23    {
24      v5 = dword_140005184 && dword_140005260[i] == dword_140005000[i];
25      dword_140005184 = v5;
26      if ( !v5 )
27        break;
28    }
29    v7();
30    if ( dword_140005184 )
31      sub_140001E80("%s", aTtttqqqqqqqlll);
32    else
33      sub_140001E80("%s", aWhatAPityPlzTr);
34    return 0;
35  }
36  else
37  {
    puts("Wrong length!");
```

改好看点

```
 9  v7 = (void (*)(void))sub_140001BE0;
10  if ( dword_1400053F0 )
11    v7 = (void (*)(void))sub_1400010A0;
12  puts("This is a surprise!");
13  printf("Give me your flag: ");
14  scanf("%s", v8);
15  v6 = -1i64;
16  do
17    ++v6;
18  while ( v8[v6] );
19  if ( v6 == 88 )
20  {
21    encode((__int64)v8);
22    for ( i = 0; i < 88; ++i )
23    {
24      v5 = judge && var_1[i] == var_2[i];
25      judge = v5;
26      if ( !v5 )
27        break;
28    }
29    v7();
30    if ( judge )
31      printf("%s", aTtttqqqqqqqlll);
32    else
33      printf("%s", aWhatAPityPlzTr);
34    return 0;
35  }
36  else
37  {
38    puts("Wrong length!");
```

encode函数里面还挺复杂的，看了好长时间看不懂，最后看到hint说是要patch后爆破，就去搜了搜爆破方法

先patch程序，使其能输出正确的长度

改一下 if(judge)......else...... 的else部分语句就行

patch前

```
.text:0000000140001E47                          loc_140001E47:                 ; CODE XREF: main+179↑j
.text:0000000140001E47 48 8B 54 24 40           mov     rdx, [rsp+0E8h+var_A8]
.text:0000000140001E4C 48 8D 0D 3D 14 00 00     lea     rcx, aS_0              ; "%s"
.text:0000000140001E53 E8 28 00 00 00           call    printf
```

patch后

```
.text:0000000140001E47                          loc_140001E47:                 ; CODE XREF: main+179↑j
.text:0000000140001E47 48 8B 54 24 20           mov     rdx, qword ptr [rsp+0E8h+var_C8] ; Keypatch modified this from:
.text:0000000140001E47                                                         ;   mov rdx, [rsp+0E8h+var_A8]
.text:0000000140001E4C 48 8D 0D 3D 14 00 00     lea     rcx, aS_0              ; "%d"
.text:0000000140001E53 E8 28 00 00 00           call    printf
```

测试输入输出

```
This is a surprise!
Give me your flag: moectf{AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA}
7
```

然后写爆破脚本

```python
from subprocess import Popen,PIPE

flag = 
'moectf{AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAA'

print(len(flag))
ans = ''
k = 7
while(k<=88):
    for i in range(33,127):
        p = Popen("此处为绝对路径",shell = True,stdin = PIPE,stdout = PIPE, 
text=True, bufsize=0)
        #p.wait()
        ans = list(flag)
        ans[k] = chr(i)
        ans = ''.join(ans)
        #ans += '}'
        p.stdin.write((ans+'\n'))
        stdout,stderr = p.communicate()
        #print(ans)
        if k<=8:
            check = stdout[-1]
        else:
            check = stdout[-2] + stdout[-1]
        if(int(check) >= k+1):
            flag = list(flag)
            flag[k] = chr(i)
            flag = ''.join(flag)
            k += 1
            print(flag)
            print(k)
            break
```