

# 2022DASCTF X SU Re WriteUp

原创

Whitebird\_ 已于 2022-04-01 20:35:09 修改 350 收藏 1

分类专栏: [DASCTF](#) 文章标签: [安全](#)

于 2022-04-01 20:29:55 首次发布

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/jxnu\\_666/article/details/123906920](https://blog.csdn.net/jxnu_666/article/details/123906920)

版权



[DASCTF 专栏收录该内容](#)

1 篇文章 0 订阅

订阅专栏

## 2022DASCTF X SU Re WriteUp

easyre

StarGate

### easyre

一道签到题,老样子先查壳

Exeinfo PE - ver.0.0.6.7 by A.S.L - 1102+114 sign 2021.10.02

File: easyre.exe

Entry Point: 000DA001 oo < EP Section: .aspack

File Offset: 0003E401 First Bytes: 60.E8.03.00.00

Linker Info: 2.24 SubSystem: Win Console PE

File Size: 0003F600h < NET Overlay: NO 00000000

Image is 32bit executable RES/OVL: 0 / 0 % 2022

Generic check - Aspack v2.1x - 2.39 -> Alexey Solodovnikov - www.aspack.  
Lamer Info - Help Hint - Unpack info 0 ms.  
Big sec. 1 .text , unpack Stripper.exe v.2.07 (not 2.11) or AspackDie 1.4 - f

asp壳比较容易脱,可以百度了解一下,我直接通过ESP定律脱的,然后把内存dump下来用IDA分析

```

1 int __cdecl sub_401771(char *Str)
2 {
3     int v2[50]; // [esp+1Ch] [ebp-DCh] BYREF
4     int v3; // [esp+E4h] [ebp-14h]
5     int j; // [esp+E8h] [ebp-10h]
6     int i; // [esp+ECh] [ebp-Ch]
7
8     v3 = strlen(Str);
9     sub_401500();
10    sub_40152B();
11    sub_401593();
12    sub_401619(Str, v3);
13    for ( i = 0; i < v3; ++i )
14        byte_492A60[i] = (LOBYTE(dword_492940[i]) ^ Str[i]) + 71;
15    memset(v2, 0, sizeof(v2));
16    v2[0] = -61;
17    v2[1] = -128;
18    v2[2] = -43;
19    v2[3] = -14;
20    v2[4] = -101;
21    v2[5] = 48;
22    v2[6] = 11;
23    v2[7] = -76;
24    v2[8] = 85;
25    v2[9] = -34;
26    v2[10] = 34;
27    v2[11] = -125;
28    v2[12] = 47;
29    v2[13] = -105;
30    v2[14] = -72;
31    v2[15] = 32;
32    v2[16] = 29;
33    v2[17] = 116;
34    v2[18] = -47;
35    v2[19] = 1;
36    v2[20] = 115;
37    v2[21] = 26;
38    v2[22] = -78;
39    v2[23] = -56;
40    v2[24] = -59;
41    v2[25] = 116;

```

通过字符串定位来到加密函数，下面的v2数组是密文，str数组是我们输入的flag，分析了一下sub\_401500、sub\_40152B、sub\_401593，是一个变种的RC4。

如果是正常RC4，我们可以用脚本或者网站解，变种虽然也可以硬逆，但是比较麻烦。我这里是直接动调出密钥流，RC4虽然内部有改变，但是每次XOR的密钥流是一样的。

我们用OD进行动态调试，记住要先脱壳。

```

004017B3 > 8B45 F4      mov     eax, dword ptr [ebp-C]
004017B6 . 8B0485 402940 mov     eax, dword ptr [eax*4+492940]
004017BD . 89C1        mov     ecx, eax
004017BF . 8B55 F4      mov     edx, dword ptr [ebp-C]
004017C2 . 8B45 08      mov     eax, dword ptr [ebp+8]
004017C5 . 01D0        add     eax, edx
004017C7 . 0FB600      movzx  eax, byte ptr [eax]
004017CA . 31C8        xor     eax, ecx
004017CC . 83C0 47      add     eax, 47
004017CF . 89C2        mov     edx, eax
004017D1 . 8B45 F4      mov     eax, dword ptr [ebp-C]
004017D4 . 05 602A4900 add     eax, 00492A60
004017D9 . 8810        mov     byte ptr [eax], dl
004017DB . 8345 F4 01   add     dword ptr [ebp-C], 1
004017DF > 8B45 F4      mov     eax, dword ptr [ebp-C]
004017E2 . 3B45 EC      cmp     eax, dword ptr [ebp-14]
004017E5 . ^ 7C CC       jl     short 004017B3
004017E7 . 8D95 24FFFF lea     edx, dword ptr [ebp-DC]
004017ED . B8 00000000 mov     eax, 0
004017F2 . B9 32000000 mov     ecx, 32
004017F7 . 89D7        mov     edi, edx
004017F9 . F3:AB      rep     stos dword ptr es:[edi]
004017FB . C785 24FFFF mov     dword ptr [ebp-DC], -3D
00401805 . C785 28FFFF mov     dword ptr [ebp-D8], -80
0040180F . C785 2CFFFF mov     dword ptr [ebp-D4], -2B
00401819 . C785 30FFFF mov     dword ptr [ebp-D0], -0E
00401823 . C785 34FFFF mov     dword ptr [ebp-CC], -65
0040182D . C785 38FFFF mov     dword ptr [ebp-C8], 30
00401837 . C785 3CFFFF mov     dword ptr [ebp-C4], 0B
00401841 . C785 40FFFF mov     dword ptr [ebp-C0], -4C
0040184B . C785 44FFFF mov     dword ptr [ebp-BC], 55
00401855 . C785 48FFFF mov     dword ptr [ebp-B8], -22
0040185F . C785 4CFFFF mov     dword ptr [ebp-B4], 22
00401869 . C785 50FFFF mov     dword ptr [ebp-B0], -7D
00401873 . C785 54FFFF mov     dword ptr [ebp-AC], 2F
0040187D . C785 58FFFF mov     dword ptr [ebp-A8], -69
00401887 . C785 5CFFFF mov     dword ptr [ebp-A4], -48
00401891 . C785 60FFFF mov     dword ptr [ebp-A0], 20
0040189B . C785 64FFFF mov     dword ptr [ebp-9C], 1D

```

这一块就是对应我们IDA分析的代码区域，上面的循环是xor，下面是密文

在加密的地方下个断点，然后开始调试，前提得先知道flag的格式

格式为:DASCTF{11111112222222333333334444444455}

```

Paused
004017B3 > 8B45 F4 mov eax, dword ptr [ebp-8]
004017B6 . 8B0485 402944 mov eax, dword ptr [eax*4+492940]
004017BD . 89C1 mov ecx, eax
004017BF . 8B55 F4 mov edx, dword ptr [ebp-C]
004017C2 . 8B45 08 mov eax, dword ptr [ebp+8]
004017C5 . 01D0 add eax, edx
004017C7 . 0FB600 movzx eax, byte ptr [eax]
004017CA . 31C8 xor eax, ecx
004017CC . 83C0 47 add eax, 47
004017CF . 89C2 mov ecx, eax
004017D1 . 8B45 F4 mov eax, dword ptr [ebp-C]
004017D4 . 05 602A4900 add eax, 00492A60
004017D9 . 8810 mov byte ptr [eax], dl
004017DB . 8345 F4 01 add dword ptr [ebp-C], 1
004017DF > 8B45 F4 mov eax, dword ptr [ebp-C]
004017E2 . 3B45 EC cmp eax, dword ptr [ebp-14]
004017E5 . ^ 7C CC jnz short 004017B3
004017E7 . 8D95 24FFFFFF lea edx, dword ptr [ebp-DC]
004017ED . B8 00000000 mov eax, 0
004017F2 . B9 32000000 mov ecx, 32
004017F7 . 89D7 mov edi, edx
004017F9 . F3:AD rep stos dword ptr es:[edi]
004017FB . C780 24FFFFFF mov dword ptr [ebp-DC], -3D

```

Stack ds:[0060E2C]=44 ('D')  
 eax=006DFE2F (ASCII "DASCTF{111111122222222333333334444444455}")

00492940	38 00 00 00	78 00 00 00	DD 00 00 00	E8 00 00 00	8...x...?..?..
00492950	00 00 00 00	AF 00 00 00	BF 00 00 00	3A 00 00 00	....?..?..:...
00492960	6B 00 00 00	FB 00 00 00	B8 00 00 00	0C 00 00 00	k...?..?..:...
00492970	85 00 00 00	35 00 00 00	5C 01 00 00	AD 00 00 00	?..5... \f..?..
00492980	E6 00 00 00	00 00 00 00	E0 00 00 00	8A 00 00 00	?.....?..?..
00492990	1D 00 00 00	BD 00 00 00	46 01 00 00	D2 FF FF FF	■...?..Ff..?üü
004929A0	2B 00 00 00	00 00 00 00	15 00 00 00	24 00 00 00	+.....■...\$...
004929B0	C6 00 00 00	AD 00 00 00	A1 00 00 00	C9 00 00 00	?..?..?..?..
004929C0	7B 00 00 00	12 00 00 00	28 00 00 00	00 00 00 00	{...■...{.....
004929D0	05 00 00 00	00 00 00 00	72 00 00 00	3E 00 00 00	Ÿ.....r...>...
004929E0	10 00 00 00	A1 00 00 00	00 00 00 00	00 00 00 00	■...?.....■
004929F0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....
00492A00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....
00492A10	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....

找到密钥流，现在拿密文XOR密钥流就是flag，贴一下脚本

```

xor=[0x38,0x78,0xDD,0xE8,0X00,0XAF,0XBF,0X3A,0X6B,0XFB,0XB8,0XC,0X85,0X35,0X15C,0XAD,0XE6,0X00,0XE0,0X8A,0X1D,0X
BD,0X146,0XFFFFFFD2,0X2B,0X00,0X15,0X24,0XC6,0XAD,0XA1,0XC9,0X7B,0X12,0X28,0X00,0X05,0X00,0X72,0X3E,0X10,0XA1]

decode=[0xFFFFFC3,0xFFFFF80,0xFFFFFD5,0xFFFFF2,0xFFFFF9B,0x30,0x0b,0xFFFFFB4,0x55,0xFFFFFDE,0x22,0xFFFF
F83,0x2f,0xFFFFF97,0xFFFFFB8,0x20,0x1d,0x74,0xFFFFFD1,0x1,0x73,26,0xFFFFFB2,0xFFFFFC8,0xFFFFF5,0x74,0xFFF
FFC0,0x5B,0xFFFFF7,0xF,0xFFFFFD3,0x1,0x55,0xFFFFFB2,0xFFFFFA4,0xFFFFFAE,0x7B,0xFFFFFAC,0x5C,0x56,0xFFFF
FBC,0x23]
flag=""
for i in range(len(xor)):
    flag+=chr(((decode[i]-0x47)^xor[i])&0x7f)
print(flag)

```

这题比赛的时候没写出来，后面复现了下，还是挺有意思的。

首先这题与常规的re不同，没有给文件，给了个靶机，那就先nc一下

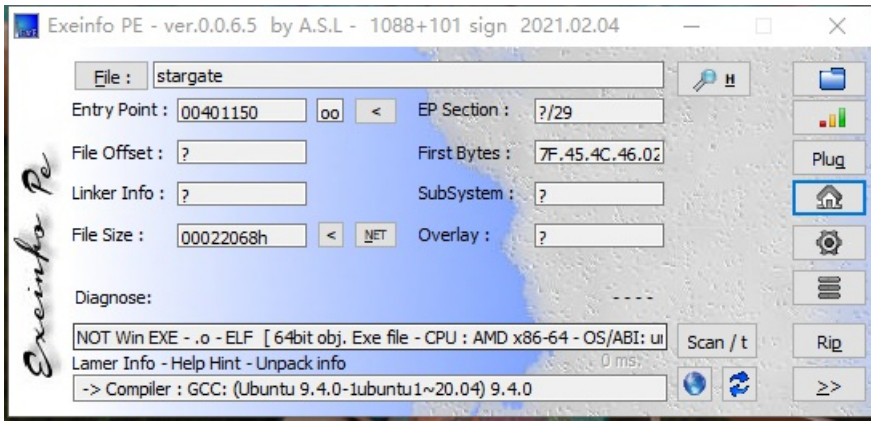
```
AAAAAAAAAOYFQAAAAAAAA5gUAAAAAAAAeAAAAAAAAAYAAAAAAAAAgAAAAAAAACAAAAAAAAAAG4AAD+//9vAgAAAAAAAAIBkA
AAAAAAAAgGAAAAAAAAQAAAAAAAAAHAAAAAQAAAAgAAAAAAAAAAAAAAAAAB9AAAAABAAAAATAAAAAAAAASAZAAAAABIBgAAAA
AAHgAAAAAAAAABgAAAAAAAAATIAAAAAAAAAABgAAAAAAAAAhwAAAAQAAABCAAAAAAAAAAMAGQAAAAAAAAAwYAAAAAAAADYAAAAAAAA
AYAAAAAYAAAACAAAAAAAAAAYAAAAAAAAAJEAAAAABAAAABgAAAAAAAAAEAAAAAAAAAQAAAAAAAAAGwAAAAAAAAAAAAAAAAAAAAQA
AAAAAAAAAAAAAAAAAAAAACMAAAAAAQAAAAAYAAAAAAAAAIBBAAAAAAAAAgEAAAAAAAAAKAAAAAAAAAAAAAAAAAAAAQAAAAAAAAABAAAA
AAAAA\lwAAAAEAAAAGAAAAAAAAAMAQQAAAAAAAAwBAAAAAAAACQAAAAAAAAAAAAAAAAAAAAEAAAAAAAAAQAAAAAAAAAKAAAAABAA
AABgAAAAAAAAABQEUAAAAAAFARAAAAAAAAAVXYBAAAAAAAAAAAAAAAAABAAAAAAAAAAAAAAAAAAAAACmAAAAAQAAAAAYAAAAAAAAq
IdBAAAAACohwEAAAAAAAA0AAAAAAAAAAAAAAAAAAAAEAAAAAAAAAAAAAAAAAAAAArAAAAEAAAACAAAAAAAAAAACQQQAAAAAAAAJAB
AAAAADqWAAAAAAAAAAAAAAAAAAAACAAAAAAAAAAAAAAAAAAAAAALQAAAABAAAAAgAAAAAADs6EEAAAAA0zoAQAAAAAzAUAAAA
AAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAADCAAAAAQAAAAIAAAAAAAAAAu05BAAAAAAC47gEAAAAABgXAAAAAAAAAAAAAAAA
AIAAAAAAAAAAAAAAAAAAAAAzAAAAA4AADAAAAAAAAABAEQgAAAAAAEA4CAAAAAAIAAAAAAAAAAAAAAAAAAAAACAAAAAAAAAAIA
AAAAAANgAAAAPAAAAAwAAAAAAAAAYHkIAAAAAABgOAgAAAAACAAAAAIAAAAAAAAAAAAAAgAAAAAAACAAAAAAADkAAAA
BgAAAAMAAAAAAAAAIB5CAAAAAAAGDgIAAAAAANABAAAAAAABwAAAAAAAAAIAAAAAAAAAABAAAAAAA7QAAAAEAAADAAAAAAA
AAPAfQgAAAAA8A8CAAAAAAQAAAAAAAAAAAAAAAAAAAAACAAAAAAAATAAAAAAAPIAAAAABAAAAAwAAAAAAAAAAIEIAAAAAA
AQAgAAAAAYAAAAAAAAAAAAAAAAAAAAAgAAAAAAACAAAAAAAAD7AAAAAQAAAAAMAAAAAAAAYBCAAAAABgEAIAAAAANgHA
AAAAAAAAAAAAAAAAAAAAIAAAAAAAAAAAAAAAAAAAAAQEAAGAAADAAAAAAAAAAEAOQgAAAAAABgCAAAAAAwAAAAAAAAAAAAAAAA
AAAAIAAAAAAAAAAAAAAAAAAAAAAYBAABAAAAAMAAAAAAAADgYAgAAAAAAKQAAAAAAAAAAAAAAAAAAAAEAAAAAAA
AAQAAAAAAAAABAAAAAwAAAAAAAAAAAAAAAAAAAAABhGAIAAAAAAB8BAAAAAAAAAAAAAAAAAAAAABAAAAAAAAAAAAAAAAAAAA
==end==
Legend has it that there are hidden treasures in the universe.
Explorer, can you find the treasure through the stargate?
Passing through each stargate requires the correct code, can you find the correct way through the s
tar gate?
The first stargate is right in front of you.
Password : Wrong password!
You hava be stuck in first star forever!!
Bye Bye!
```

上面好像是个base64编码的文件，先用网站的base64解密下看看

我这里直接用py写了个脚本交互下载文件，后面也会用到

```
from pwn import*
import base64
import sys
import subprocess
p=remote('node4.buuoj.cn',26237)
p.recvuntil("Gate")#将Wellcome to Star Gate接收
code=p.recvuntil("==end==")#接收中间的base64
with open("stargate",'wb') as f:
    f.write(base64.b64decode(code[:-8]))#code 包含==end==，所以只编码到倒数第八个字符
p.interactive()
```

用IDA分析一下这个文件，ELF、无壳、64位的程序



```

1 void __fastcall __noreturn main(__int64 a1, char **a2, char **a3)
2 {
3   char psw[40]; // [rsp+0h] [rbp-30h] BYREF
4   unsigned __int64 v4; // [rsp+28h] [rbp-8h]
5
6   v4 = __readfsqword(0x28u);
7   sub_401254(a1, a2, a3);
8   puts("Legend has it that there are hidden treasures in the universe. ");
9   puts("Explorer, can you find the treasure through the stargate? ");
10  puts("Passing through each stargate requires the correct code, can you find the correct way through the star gate? ");
11  puts("The first stargate is right in front of you. ");
12  printf("Password: ");
13  isoc99_scanf("%20s", psw);
14  if ( !strcmp("pTIIOPTYFfi", psw) )
15    sub_4012C3();
16  if ( !strcmp("JvteSmWoH", psw) )
17    sub_4013AE();
18  if ( !strcmp("KltuRVDhNEpA", psw) )
19    sub_401599();
20  if ( !strcmp("WgDrztg", psw) )
21    sub_401704();
22  if ( !strcmp("DccwEiB", psw) )
23    sub_4018EF();
24  if ( !strcmp("afOgUcwpT", psw) )
25    sub_401ADA();
26  if ( !strcmp("GTonSAHTMlF", psw) )
27    sub_401C45();
28  if ( !strcmp("IhCBxd", psw) )
29    sub_401E30();
30  if ( !strcmp("tG0cyntXeBkva", psw) )
31    sub_401F9B();
32  if ( !strcmp("nYnG0tyrRfJO", psw) )
33    sub_402186();
34  if ( !strcmp("pVIMHiCDMyvcUwc", psw) )
35    sub_4024F1();
36  if ( !strcmp("oJoHGUE", psw) )
37    sub_40265C();
38  if ( !strcmp("dnoSkEMyB", psw) )
39    sub_4027C7();
40  if ( !strcmp("cbRNHqBlveM", psw) )
41    sub_402932();
42  if ( !strcmp("irtYDTaqQqhqT", psw) )
43    sub_402A9D();
44  if ( !strcmp("dGnOkwvBijXBx", psw) )

```

会发现让我们输入一个password，然后进入一个函数

```
LUA View-A 4 fseudocode-A Strings window Hex View-1 Structures Enums
1 int64 sub_4012C3()
2 {
3 char s2[40]; // [rsp+0h] [rbp-30h] BYREF
4 unsigned int64 v2; // [rsp+28h] [rbp-8h]
5
6 v2 = __readfsqword(0x28u);
7 printf("Now you in universe p1TIOPTYFfi, please enter the password to enter the next universe through the stargate. \n: ");
8 __isoc99_scanf("%20s", s2);
9 if ( !strcmp("xhgrNMZizKMMW", s2) && dword_4221C8 == 1 )
10 {
11 dword_4221C8 = 0;
12 sub_40A60A();
13 exit(0);
14 }
15 if ( !strcmp("acJOFKGoRXppqN", s2) && dword_42211C == 1 )
16 {
17 dword_42211C = 0;
18 sub_413591();
19 exit(0);
20 }
21 puts("Wrong password!\nYou're stuck in this universe forever!!");
22 return 0LL;
23 }
```

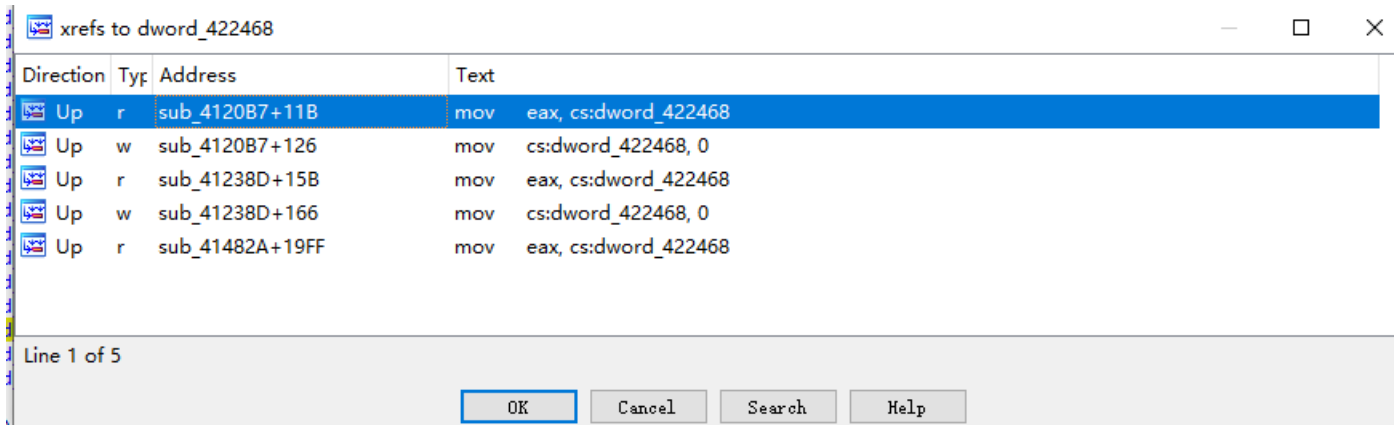
进去之后，会让你继续输入一个password，再调用其他函数，有点套娃，并且把dword\_4221C8这种全局变量由1变成了0

然后我在字符串中看到了cat flag，这个命令是用来获取远程主机上的flag，莫非是个pwn？

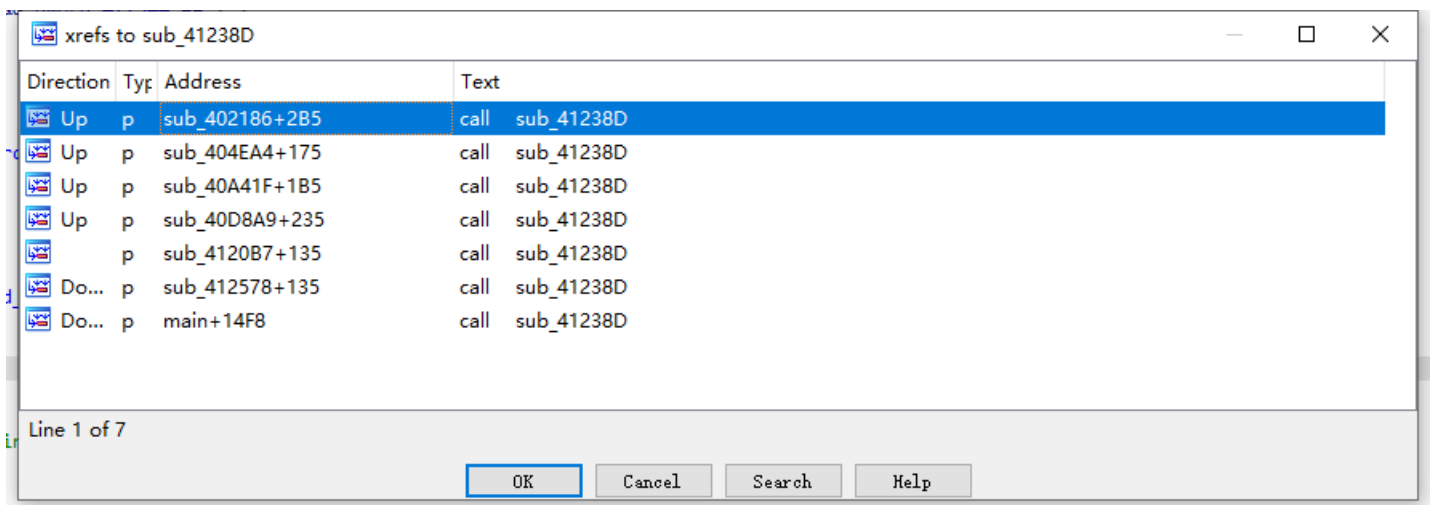
对字符串进行交叉引用，找到了调用该命令的函数

```
457 && !dword_42231C
458 && !dword_4226A8
459 && !dword_4221B8
460 && !dword_42216C
461 && !dword_422774
462 && !dword_422380
463 && !dword_422608
464 && !dword_4227D4
465 && !dword_4226F4
466 && !dword_422440
467 && !dword_4220A0
468 && !dword_422704
469 && !dword_422498
470 && !dword_422754
471 && !dword_4220D8
472 && !dword_4222DC
473 && !dword_422290
474 && !dword_42276C
475 && !dword_422178
476 && !dword_422444
477 && !dword_422138
478 && !dword_422690
479 && !dword_422180
480 && !dword_4226AC
481 && !dword_4227C8
482 && !dword_422524
483 && !dword_422468
484 && !dword_42271C
485 && !dword_422314 )
486 {
487 catflag();
488 }
489 puts("There seems to be treasure in this universe but you're not qualified to get it ");
```

上面是个if(!xxxx)的条件判断，大概有四五百个dword\_xxxx这种变量并且初始值都是1，并且我们需要将所有的1变成0，这样就可以获得flag了



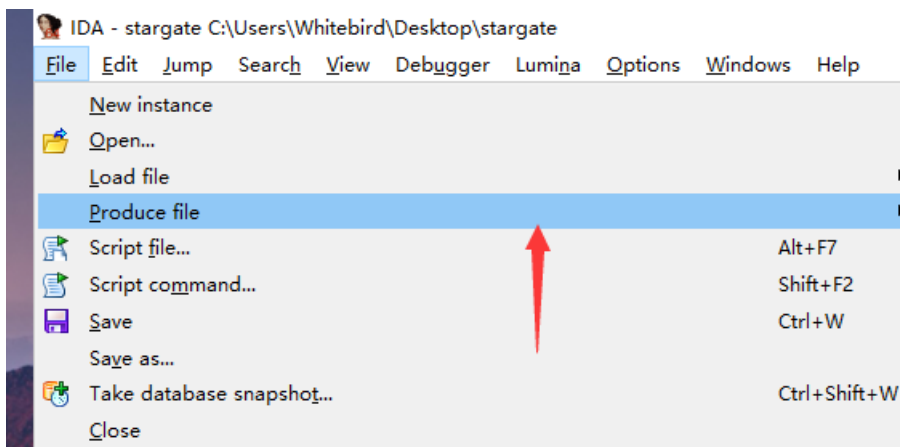
发现一个变量对应了多个函数，而一个函数又有可能会出现在多个函数里，并且也对应了多个变量



需要注意的是如果这个变量已经是0的话, 你是不能走这条路的.

通过简单分析后, 就知道这个类似于一个图. 变量类似于是图中的边, 而函数就是图中的点

这样子题目的意思是让我们以某种顺序来输入password, 并且没有重复, 最后将所有的变量变为0, 出题人也提示了欧拉路径, 其实我感觉像一个迷宫



先生成一个asm文件, 然后用py脚本把所有的password提取出来



```

import os
points = []
now, rt = -1, 0
flag = False
edges = []
with open("stargate.asm") as f :
    tmp = f.readlines()
    for i in range(len(tmp)) :
        if 'lea    rdi, aLegendHasItTha' in tmp[i] :#起点
            break
        if 'lea    rdi, aThere' in tmp[i] :#终点
            flag = True
        if 'call    _strcmp' in tmp[i] :#获取_strcmp上一行的password, 也就是进入下一个星球的password
            s = tmp[i-1].find('')
            p = tmp[i-1][s+1:-2]
            if p not in points :#如果之前没走过, 就新加一个
                points.append(p)
            if now == -1 :
                continue
            print(points[now], p, i)
            edges.append([now, points.index(p)])
        elif 'lea    rdi, aNowYouIn' in tmp[i] :
            s = tmp[i].find('se ')#获取"Now you in universe xxxxx"中的password
            t = tmp[i][s:].find(', ')
            p = tmp[i][s+3:s+t]
            if p not in points :
                points.append(p)
            now = points.index(p)#now为p在点中的索引
            if flag :
                rt = now#最后终点的index
                flag = False#后面还要继续遍历
print(points)
ff = open("tt.in", "w")
ff.write(str(len(points)) + " " + str(len(edges)) + " " + str(rt) + '\n')
for i in edges :
    ff.write(str(i[0]) + " " + str(i[1]) + "\n")#以两个索引之间作为边, 每个password就是索引, 也是点
ff.close()
os.system(".\dfs.exe")#利用dfs跑欧拉
s = input().split(' ')#将跑完的路径给作为输入带回, 用来以索引生成真正的password
with open('tt.ans', "w") as f :
    for i in range(len(s)) :
        if s[i] == s[i-1] :#结束
            continue
        try :
            f.write(points[int(s[i])-1]+" ")#将index替换成password
        except :
            print(i)
print(len(edges)//2, len(s))

```

我们利用password来当作点，两点的索引组成一条边，剩下的就是找个脚本跑这个欧拉图就行了

```

185 958 145
2 3
2 4
2 5
2 6
2 7
2 8
9 10
9 11
9 12
9 13
14 15
14 16
14 17
14 18
14 19
14 20
21 22
21 23
21 24
21 25
21 26
21 19
3 2
3 27
3 28
3 29
27 3
27 30
27 31
27 32
27 33
27 34
30 27
30 35
30 18
30 36
37 38
37 39
37 40
37 41
37 42
37 43
38 37
38 44
38 45
38 46

```

这个是将password变成了索引，一对为边

```

#include <cstdio>
#include <iostream>
#include <vector>
using namespace std;
vector<int> v[1000];
int x,y,n,m,du[5020],sta=0x3f3f3f,mp[2520][2520],ans[5200],top;
inline int read()
{
    int x=0,f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=(x<<1)+(x<<3)+(c^48);c=getchar();}
    return x*f;
}
int cnt1 = 0;
void dfs(int x)
{
    for(int i=1;i<=n;i++)
    {
        if(mp[x][i])
        {

```

```

        mp[x][i]--;
        mp[i][x]--;
        cnt1 ++;
        dfs(i);
    }
}
ans[++top]=x;
}
void dfs1(int p, int t) {
    v[t].push_back(p);
    for(int i = 1; i <= n; i ++ ) {
        if(mp[p][i]) {
            mp[p][i]--;
            cnt1 ++;
            mp[i][p]--;
            dfs1(i, t);
        }
    }
}
int main()
{
    freopen("tt.in", "r", stdin);
    freopen("tt.out", "w", stdout);
    int rt;
    scanf("%d%d%d", &n, &m, &rt);
    rt ++;
    for(int i=1;i<=m;i++)
    {
        scanf("%d%d", &x, &y);
        x ++, y ++;
        mp[x][y] = 1;
        mp[y][x] = 1;
        du[x]++;
        du[y]++;
    }
    dfs(rt);
    while(true) {
        for(int i = 1; i <= n; i ++ ) {
            dfs1(i, i);
        }
        bool flag = 0;
        for(int i = 1; i <= n; i ++ ) for(int j = 1; j <= n; j ++ ) flag |= mp[i][j];
        if(!flag) break;
    }
    for(int i = 1; i <= top; i ++ )
        printf("%d ", ans[i]);
    printf("%d", rt);
    printf("\n%d\n", cnt1);
    return 0;
}

```

```
100 102 110 134 148 142 147 179 149 126 133 134 114 64 78 177 130 160 139 184 123 178 171 181 111 160 162 161 157 135 145 152 158 159 173 174 150 130 115 178 166 70 145 151 155 146 172 182 127
180 124 121 122 146 131 125 118 110 124 161 163 97 178 95 169 81 122 25 104 105 184 54 177 116 113 36 114 58 118 117 119 37 177 51 157 156 98 171 129 137 139 167 168 109 102 111 58 134 139 108
129 172 41 171 99 174 183 141 140 7 131 74 169 9 183 165 86 102 108 130 89 128 94 110 133 135 167 1 176 2 113 115 82 146 90 83 91 69 94 93 92 158 50 180 57 93 80 69 160 73 75 147 19 122 96 61
130 71 81 68 82 72 67 74 71 67 70 109 135 144 173 84 149 120 150 56 121 50 135 43 150 26 141 42 117 84 108 87 97 105 56 69 67 75 161 54 134 179 154 159 49 175 65 80 68 67 73 66 62 64 68 79 84
77 63 78 57 94 129 103 152 13 161 9 148 185 27 78 181 14 180 132 126 182 48 66 63 62 65 50 51 72 29 108 43 58 80 18 57 48 91 137 136 138 153 154 96 45 96 127 142 106 101 100 103 44 156 21 114
112 32 81 132 5 129 1 109 54 128 5 130 20 155 143 76 126 89 88 40 58 164 170 41 166 52 42 70 47 63 76 101 107 1 101 8 184 53 165 34 51 39 54 126 46 127 23 144 162 95 45 97 5 131 49 39 52 46 39
47 34 33 28 34 25 27 22 25 142 99 17 79 83 92 44 38 43 45 39 48 64 26 52 143 59 89 21 117 120 20 95 11 102 30 92 8 100 9 57 16 56 9 3 8 121 123 136 24 145 13 63 5 117 19 72 83 85 87 61 60 59 124
6 78 30 4 29 44 115 151 41 38 42 39 50 13 85 86 11 54 2 9 43 82 106 23 22 26 33 1 41 6 60 7 110 37 90 178 14 10 13 69 19 64 32 28 35 42 11 35 46 5 3 7 120 24 70 21 15 18 48 36 31 28 4 3 6 58 55
59 20 19 59 17 15 19 31 37 40 38 39 53 12 10 11 55 16 15 20 22 24 146 146
480
```

这是跑完欧拉后的结果，依次输入对应索引的password就是真正的路径，但是不知道为什么dfs后的值都+1了，懂得师傅可以评论告诉我一下谢谢

```
aGheqUNxbhQFLC hXgVcGfB WxTiyPfqhUdlg WYgxmt fmPuhKpgPYQEjCj jAQEnriM ZllVDCuu puOxDKyxv gVYCopRcznutYJ pewzLoBxM OrWtMkMwCRO WYgxmt XRFHmR JXtVomwxyFw
glhZCMTWqe oDSOIQSZdm hHKTKJLVyfqEK lbPgduOHctx cnYwnZip JlQvqWrydNgwBLq mAaIJlyFBAyf ItATthoGc hYnTDUooPmliEoO CbulBTJ zxpDvZbsWOAgNH lbPgduOHctx eryBDtEAQ
VltPAFiQAc vBQDukqYUDkwj xcazvlcBbChpgmS IcWOrv WBeGSuASyNGpcFb ahtjEVI MkJbnoR nKMDIqMnDJyjk KIYJtPnxhO wheOBPHkCw hHKTKJLVyfqEK SIbKySpErhCr ItATthoGc
KaAtcOwUmVN JShaWwciPo IcWOrv MvFhRyqNnHo hImfUwabGNuzv tmuqpinRnu pgrqzJA cmvnuPnQwpsQuY wEGoElEn voSHHINIPMIQ kHgnSursXl XKRKJYDFUqBfry yJmVpxil tmuqpinRnu
lbXpciSwZfjBF HbVxYb1S POMouWpuZf WxTiyPfqhUdlg kHgnSursXl VltPAFiQAc HdEgADNNWxfdu FnxotQqyYbHii ItATthoGc QnMIZZPwAir JfBYjvmrowT sBDuCsYt yJmVpxil YruHIXE
bBVuLcJLpBOHibb XuigMvtFw JlQvqWrydNgwBLq OzfsOfhq oDSOIQSZdm dGxMrrlnXIJQHT uWDYzOgXeHlJ FKTFWgPX XRFHmR XdOMfRtMpPpJ POMouWpuZf xSTzYLSdVe VntSSkPrpF HJz1bYKEfc
oDSOIQSZdm xPRZXmoOk1PO vBQDukqYUDkwj jMaAIWLR EdIoTPnJWQJnUN hYnTDUooPmliEoO ldkFxad ttLrCbGSyIgoI cnYwnZip lqsNhb mxkOwqgkzKJFOJn dKhfjtABcgXtUfg hXgVcGfB
zxpDvZbsWOAgNH XdOMfRtMpPpJ WYgxmt cnYwnZip uKptJUOKJuSs ldkFxad pgrqzJA JMmbrkUpQofPxxU hYnTDUooPmliEoO VVSpzHQ KIYJtPnxhO aOUzHueNIQExdUB obuIqMnGV IMBiEQDxyWjIpe
cpazDfrTFlrpaq lbXpciSwZfjBF svdkXpnKz JfBYjvmrowT kJAVjbcE aOUzHueNIQExdUB wSQjhRopvvr1WT GPxitiY hXgVcGfB uKptJUOKJuSs hHKTKJLVyfqEK ohNwxyfNiIhryd lkYrgVfkUgDdcX
TAFPEPHUvps WxTiyPfqhUdlg OrWtMkMwCRO xcazvlcBbChpgmS lqsNhb xhgrNMZizKMWwM pLIIOPTyffi acJOFKGoRkppqN uWDYzOgXeHlJ SIbKySpErhCr pnPequaU tmuqpinRnu AtgeQYKKVxnF
qJvVHX MIoTBR tboOiqkLZ TAFPEPHUvps LaFqjayomE AChrkQNYUpwCy ahtjEVI AHPHdSL voSHHINIPMIQ OTPxcEPzxfGd LaFqjayomE PnrWmP tboOiqkLZ lbPgduOHctx HPEFNPcDjPUB EJJsKsxPARqD
ZllVDCuu qnbzWRDYRA yJmVpxil FfKwSvDisEILsG zYbsbSGzOt hHKTKJLVyfqEK bXHeEXyApBMGSen sBDuCsYt YhFBSor pnPequaU ZZhkHISyrDpMgX irtYDTaqQhgqT svdkXpnKz bXHeEXyApBMGSen
irtYDTaqQhgqT JShaWwciPo dKhfjtABcgXtUfg xcazvlcBbChpgmS yt1KhpTjSkbo nKMDIqMnDJyjk rRNsmYx gVYCopRcznutYJ sKTFcQaMD wheOBPHkCw NntzHPUhl XKRKJYDFUqBfry AHPHdSL xcazvlcBbC
gbSoYpnwXoAdthB wheOBPHkCw vnrwThDvtVAFtJ obuIqMnGV zvWmuvNdnndgJFv xSTzYLSdVe rRNsmYx uKptJUOKJuSs yxKgatqVWDbO FnxotQqyYbHii XuigMvtFw NntzHPUhl tboOiqkLZ irtYDTaqQhgqT
EJJsKsxPARqD VltPAFiQAc OzfsOfhq WYgxmt puOxDKyxv OLQPocVMtrkKSwO MkJbnoR RTxPwFhkStopR NfMNLDOpeU oLtzrsYyxnnzME PnrWmP YhFBSor irtYDTaqQhgqT HPEFNPcDjPUB CqzbzHldx
cbRRNHqBlveM JXtVomwxyFw YhFBSor VjOQKTIKO rRNsmYx pyYBEcQCoHYI dGnOkWvBijXBX glhZCMTWqe OTPxcEPzxfGd TAFPEPHUvps ldkFxad oBnrIdkMfNKALD WBeGSuASyNGpcFb PMNZeHaLbzHBY
VltPAFiQAc kJAVjbcE fmPuhKpgPYQEjCj CQYHeGKhdh CLmNhtsNncT glhZCMTWqe CbulBTJ NqQrGKUL1Z voSHHINIPMIQ vmgDDFMQwmO pewzLoBxM cmvnuPnQwpsQuY aJWSnHm CqzbzHldx
dGnOkWvBijXBX cbRRNHqBlveM oLtzrsYyxnnzME AHPHdSL xPRZXmoOk1PO ZzhkHISyrDpMgX mPahPbnUvAjd uKptJUOKJuSs gbSoYpnwXoAdthB XdOMfRtMpPpJ PnrWmP TAQIduWLogrSoy OTPxcEPzxfGd
aJWSnHm MIoTBR ttLrCbGSyIgoI UwIUVXPCeuwFC PrAlNmokb zJecaZMpbVhyv OLQPocVMtrkKSwO EdIoTPnJWQJnUN yYWTsvm FfKwSvDisEILsG wEGoElEn jAQEnriM awyXfbcExww mpSmqzYvzNvg
aGheqUNxbhQFLC oBnrIdkMfNKALD YlPrLWfmgqyd jMaAIWLR BnPhWQYzhWuHNv XRFHmR DOoxgrwboB ggsTEwgg sBDuCsYt vmgDDFMQwmO xIqehhoWQ ldkFxad xhgrNMZizKMWwM dKhfjtABcgXtUfg
OzfsOfhq lkYrgVfkUgDdcX xIqehhoWQ hHKTKJLVyfqEK FEIoWCryCTkaA hImfUwabGNuzv XPfoOoPqLoaTT ESXISQZkKdM pewzLoBxM ohNwxyfNiIhryd tFGLkVrWtOnbTQN zWwspWPFJiISF XdOMfRtMpPpJ
RbyuRQPZM RdeZEIClNH JMmbrkUpQofPxxU KaAtcOwUmVN zwYyIatV zvWmuvNdnndgJFv JShaWwciPo NLIQLKqH5JHPNFB dGnOkWvBijXBX ESXISQZkKdM mpSmqzYvzNvg nuDjIEK xhgrNMZizKMWwM mpSmqzYvz
SiaCVVqFQ JlQvqWrydNgwBLq bkcvEIGxS wSQjhRopvvr1WT aJOahZmL xPRZXmoOk1PO nYnG0tYrFJJO OzfsOfhq pewzLoBxM sJffJkzX wEGoElEn omBnHhHlNBpUc yt1KhpTjSkbo eryBDtEAQ QnMIZZPwAir
```

替换成password，这么多肯定不是我们手动输入，必须还得使用交互式脚本来

```
print("when ready, press enter to continue.")
ans = input().split(' ')[:-1]
print(ans)
r.recvuntil('Password: ')
for i in range(len(ans)):
    print(i, ans[i])
    r.sendline(ans[i])
    try:
        print(r.recvuntil(':'))
    except:
        break
r.interactive()
```

最后可以把这个与前面的脚本结合一下：

```

from pwn import*
import base64
import sys
import subprocess
p=remote('node4.buuoj.cn',26237)
p.recvuntil("Gate")#将Wellcome to Star Gate接收
code=p.recvuntil("==end==")#接收中间的base64
with open("stargate",'wb') as f:
    f.write(base64.b64decode(code[:-8]))#code包含==end==, 所以只编码到倒数第八个字符
print("when ready, press enter to continue.")
ans = input().split(' ')[:-1]
print(ans)
p.recvuntil('Password : ')
for i in range(len(ans)) :
    print(i, ans[i])
    p.sendline(ans[i])
    try :
        print(p.recvuntil(':'))
    except :
        break
p.interactive()

```

总体流程:

第一个py生成ELF->IDA生成ASM->第二个py生成边调用dfs, dfs结果输入回第二个py拿到点对应的password, 把password给第一个py和服务端交互, 最后拿到flag

```

482 JEsWYUQAgac
b' Now you in universe JEsWYUQAgac, please enter the password to enter the next universe through the stargate. \n:'
483 tsvUIxBTZR
b' Now you in universe tsvUIxBTZR, please enter the password to enter the next universe through the stargate. \n:'
484 kbavLM
b' Now you in universe kbavLM, please enter the password to enter the next universe through the stargate. \n:'
485 aOfclGDrs
b' Now you in universe aOfclGDrs, please enter the password to enter the next universe through the stargate. \n:'
486 wvYCbBKyDFsW
b' Now you in universe wvYCbBKyDFsW, please enter the password to enter the next universe through the stargate. \n:'
487 JEsWYUQAgac
b' Now you in universe JEsWYUQAgac, please enter the password to enter the next universe through the stargate. \n:'
488 MXHFBhdceaixHo
b' Now you in universe MXHFBhdceaixHo, please enter the password to enter the next universe through the stargate. \n:'
489 bfSxFrsROEyDIil
b' Now you in universe bfSxFrsROEyDIil, please enter the password to enter the next universe through the stargate. \n:'
490 OJlmcjQ
[*] Switching to interactive mode
flag{e5143a52-3a0d-4d67-ac82-10f3b674fdd2}
Bye Bye!
[*] Got EOF while reading in interactive
$
[*] Interrupted
[*] Closed connection to node4.buuoj.cn port 26237

(root@kali)-[~/桌面/re]
└─#

```

这题主要还是学习交互式脚本的写法, 平时对于这种脚本的使用比较少, 导致做题时没什么思路