攻防世界### Reversing-x64Elf-100

下载附件后先用ida打开,查看了strings部分,发现该文件需要在linux中打开,
之后查看main函数的c语言伪代码,并且main中显示了返回1时为错误,返回1时才正确,并且根据代码中显示调用了sub_4006FD,

```
int64 fastcall main(int a1, char a2, char a3)
{
char s[264]; // [rsp+0h] [rbp-110h] BYREF
unsigned __int64 v5; // [rsp+108h] [rbp-8h]

v5 = __readfsqword(0x28u);
printf("Enter the password: ");
if ( !fgets(s, 255, stdin) )
return 0;
if ( (unsigned int)sub_4006FD(s) )
{
puts("Incorrect password!");
return 1;
}
else
{
puts("Nice!");
return 0;
}
}
```

查看这个函数的代码后发现if的判断语句为*(char*)(v3[n11 % 3] + 2 *(n11 / 3))* - (char *)(n11 + p_s) != 1

```
int64 fastcall sub_4006FD(__int64 p_s)
{
int n11; // [rsp+14h] [rbp-24h]
_QWORD v3[4]; // [rsp+18h] [rbp-20h]

v3[0] = "Dufhbmf";
v3[1] = "pG`imos";
v3[2] = "ewUglpt";
for ( n11 = 0; n11 <= 11; ++n11 )
{
if ( *(char* )(v3[n11 % 3] + 2 *(n11 / 3))* - (char *)(n11 + p_s) != 1 )
```

return 1;
}
return 0;
}

由此可得知需要该判断语句在12次循环中都要等于1才能返回0得出正确flag,由于我自身还不会写脚本,并且对于该判断语句中的二级指针也不太熟练,于是结合ai得出了脚本,运行后得出了正确密码

def generate_password():
"""
This function replicates the password validation logic from the reverse-engineered C code to generate the correct password. """ # These are the hardcoded strings from the binary
v3 = [
"Dufhbmf",
"pG`imos",
"ewUglpt"
]

```
password = []

# The loop runs 12 times, for n11 from 0 to 11
for n11 in range(12):
    # This part calculates the index to get the source character,
    # exactly matching the C code logic:        # *(v3[n11 % 3] + 2 * (n11 /
3))        string_index = n11 % 3
    char_index = 2 * (n11 // 3)  # Use integer division

    source_char = v3[string_index][char_index]

    # The condition for a correct password character is:
    # ASCII(source_char) - ASCII(input_char) == 1        # Therefore, the
correct input character's ASCII value is source_char's ASCII - 1
correct_char_code = ord(source_char) - 1

    password.append(chr(correct_char_code))

return "".join(password)
```

--- Main execution ---

if **name** == "**main**":
correct_password = generate_password()

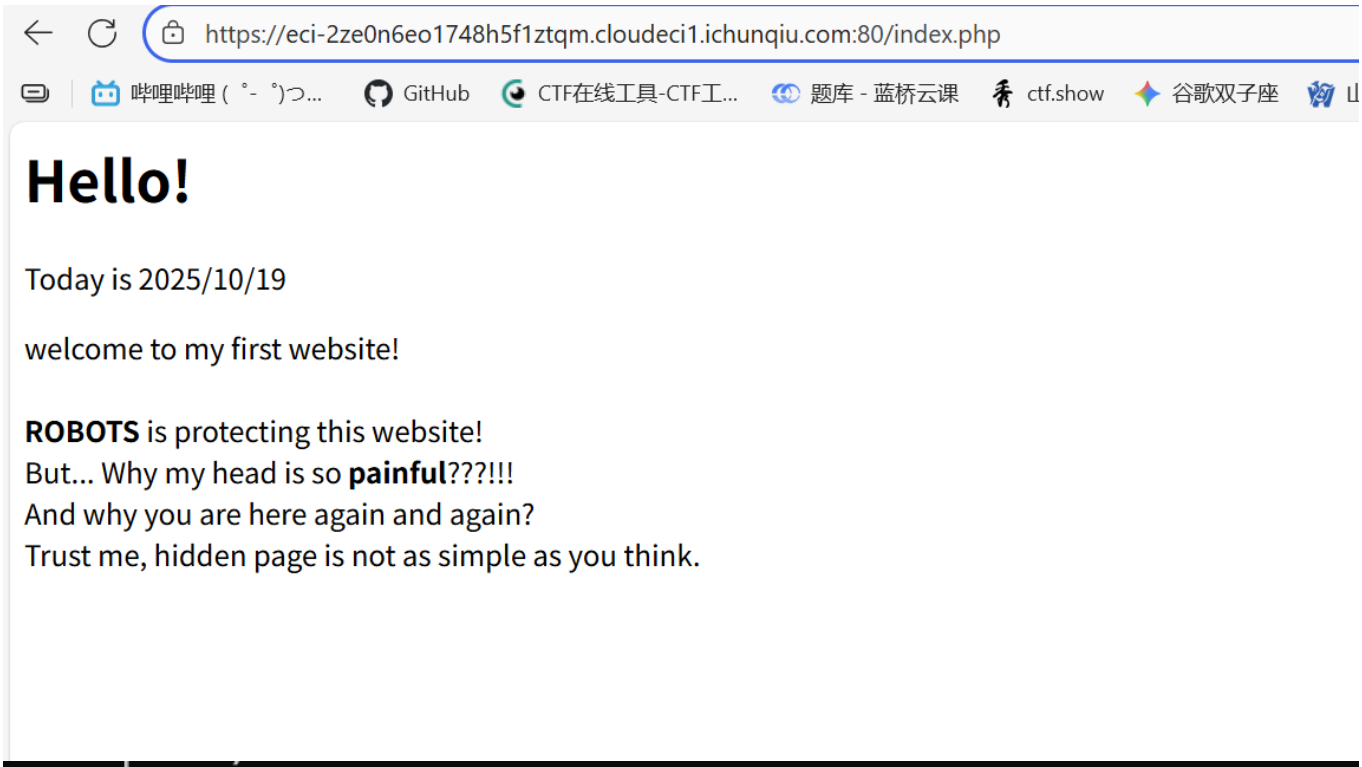
print(f"[*] The script has derived the password.")
print(f"[*] Correct password is: {correct_password}")

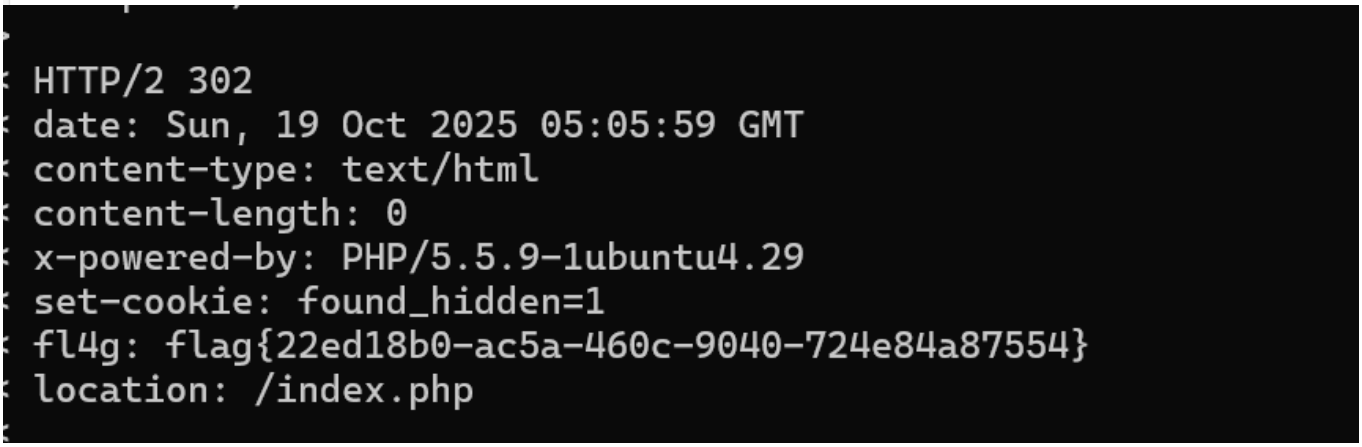
得出密码为Code_Talkers并且在linux中运行也正确

由此可得flag为改密码本身.

NEWSTAR

1,
multi-headach3





根据网页显示的提示先打开robots.txt,之后根据显示打开index.php,最后通过 `curl -v` 来查看重定向过程中的详细响应头,得到了flag

2,
strange_login
打开网页后发现需要输入账号和密码,尝试sql注入,admin'#为账号,密码随意;
得出了flag,拼凑语句大致为SELECT * FROM users WHERE username = 'admin'#' AND password = '123'

3,
宇宙的中心是php
打开网页后是一个黑洞的效果,并且我点击右键和f12没有效果,于是怀疑网页源代码里有线索,于是用 `curl` 命令绕过了限制,查看html的源代码,在注释中发现了线索/s3kret.php;于是打开这个新网页;

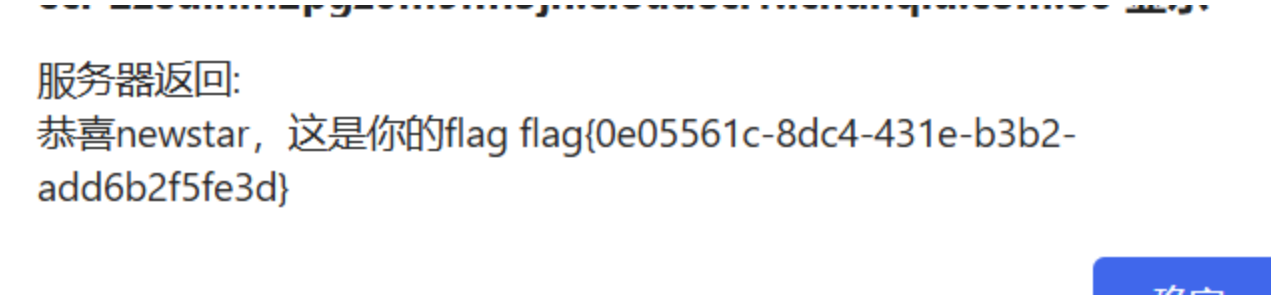
分析if的判断语句,根据intval函数的特性,十进制47转化为八进制57;于是得出curl指令curl -d "newstar2025=057" http://47.94.239.26:34820/s3kret.php; 运行后得出flag

4,

别笑,你也过不了第二关

哪吒接魔丸,第二关要求一万分,这个直接在控制台修改分数就行score = 1000000;

然后得出flag



basectf(逆向)

1,#### You are good at IDA

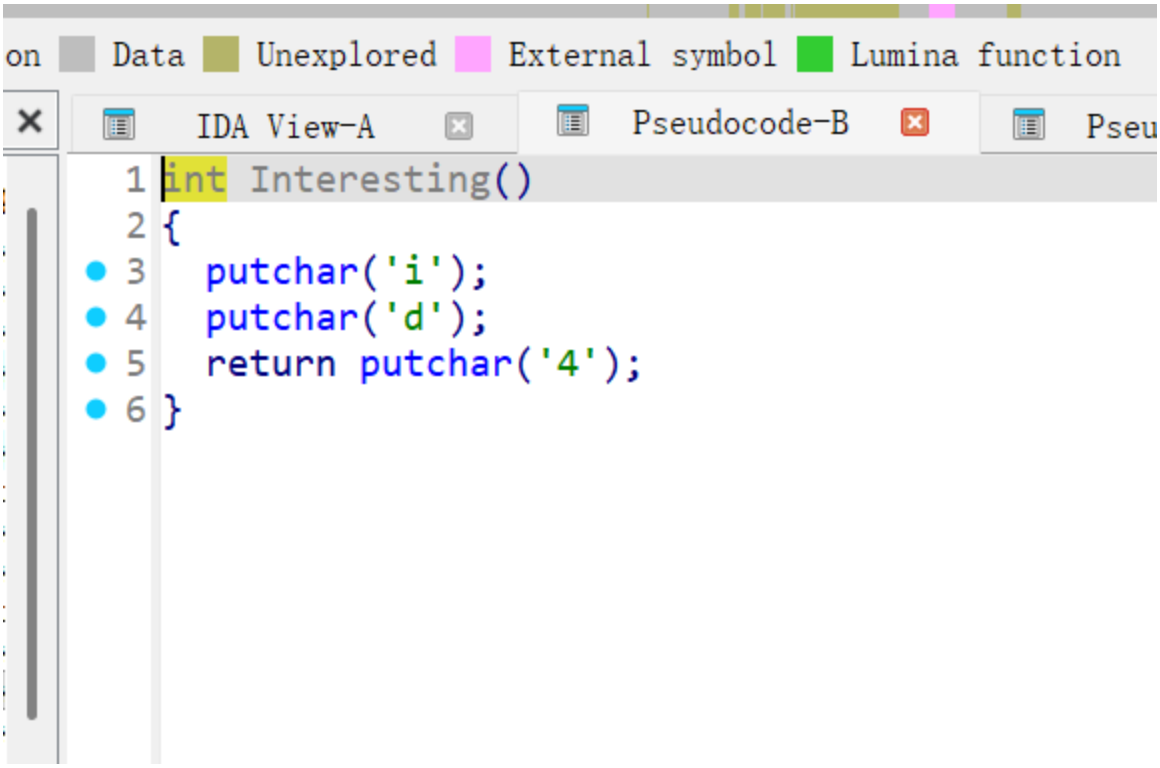
IDA View-A | Pseudocode-B | Pseudocode-A | Strings

```
1  // local variable allocation has failed, the output may be wrong!
2  int __fastcall main(int argc, const char **argv, const char **envp)
3  {
4    char Y0u_4Re__[9]; // [rsp+27h] [rbp-9h] BYREF
5
6    _main(*(_QWORD *)&argc, argv, envp);
7    strcpy(Y0u_4Re__, "Y0u_4Re_");
8    printf("This is the first part");
9    putchar(10);
10   printf("%s", Y0u_4Re__);
11   putchar(10);
12   printf("You can shift f12 look look");
13   return 0;
14 }
```

Data | Unexplored | External symbol | Lumina function

IDA View-A | Pseudocode-B | Pseudocode-A | Strings

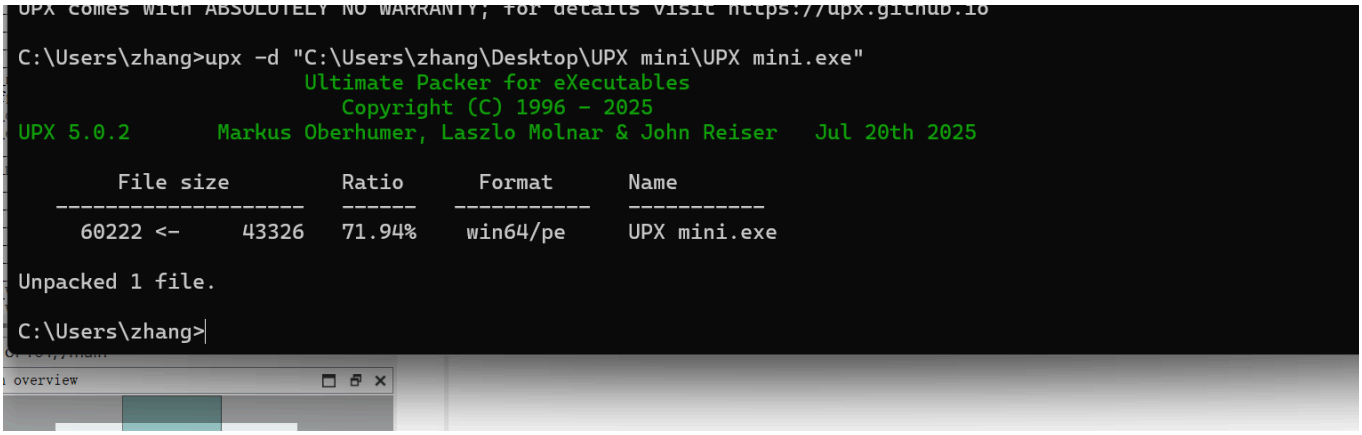
```
1  int Second()
2  {
3    printf("This is the second part");
4    putchar('9');
5    putchar('0');
6    putchar('0');
7    putchar('d');
8    putchar('_');
9    putchar('4');
10   putchar('7');
11   putchar('_');
12   printf("Only the last part remains");
13   return printf("The last part is in a named Interesting's func");
14 }
```

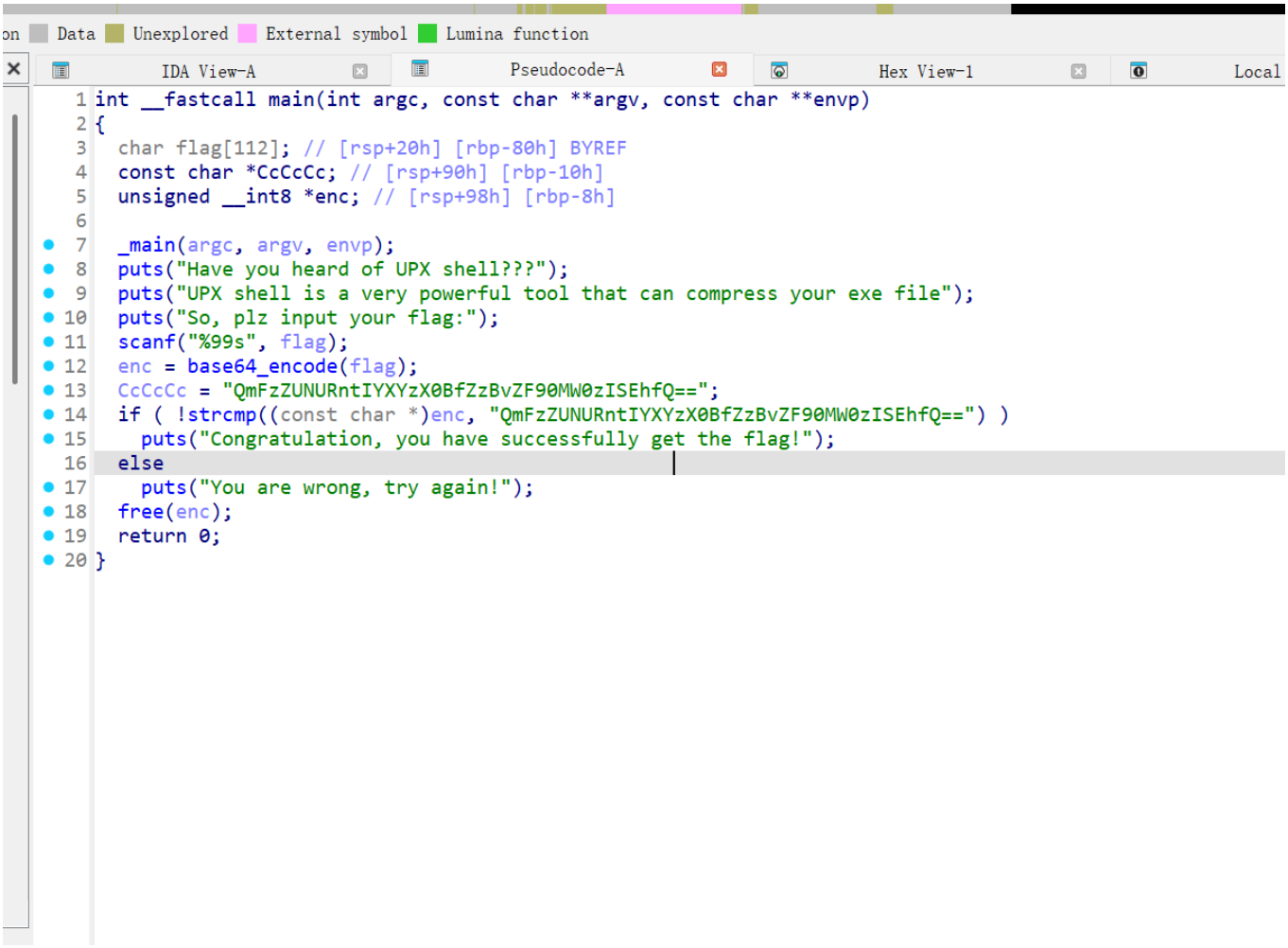


挨个在ida里找就行,拼起来即可

2,#### UPX mini

这个先用die查看,发现有壳,于是使用urx将其去壳,去壳后用ida发现main中存在base64编码,解码后得出flag(这个题学会了urx的用法和逆向题都先用die查看一下的技巧)





3,#### BasePlus

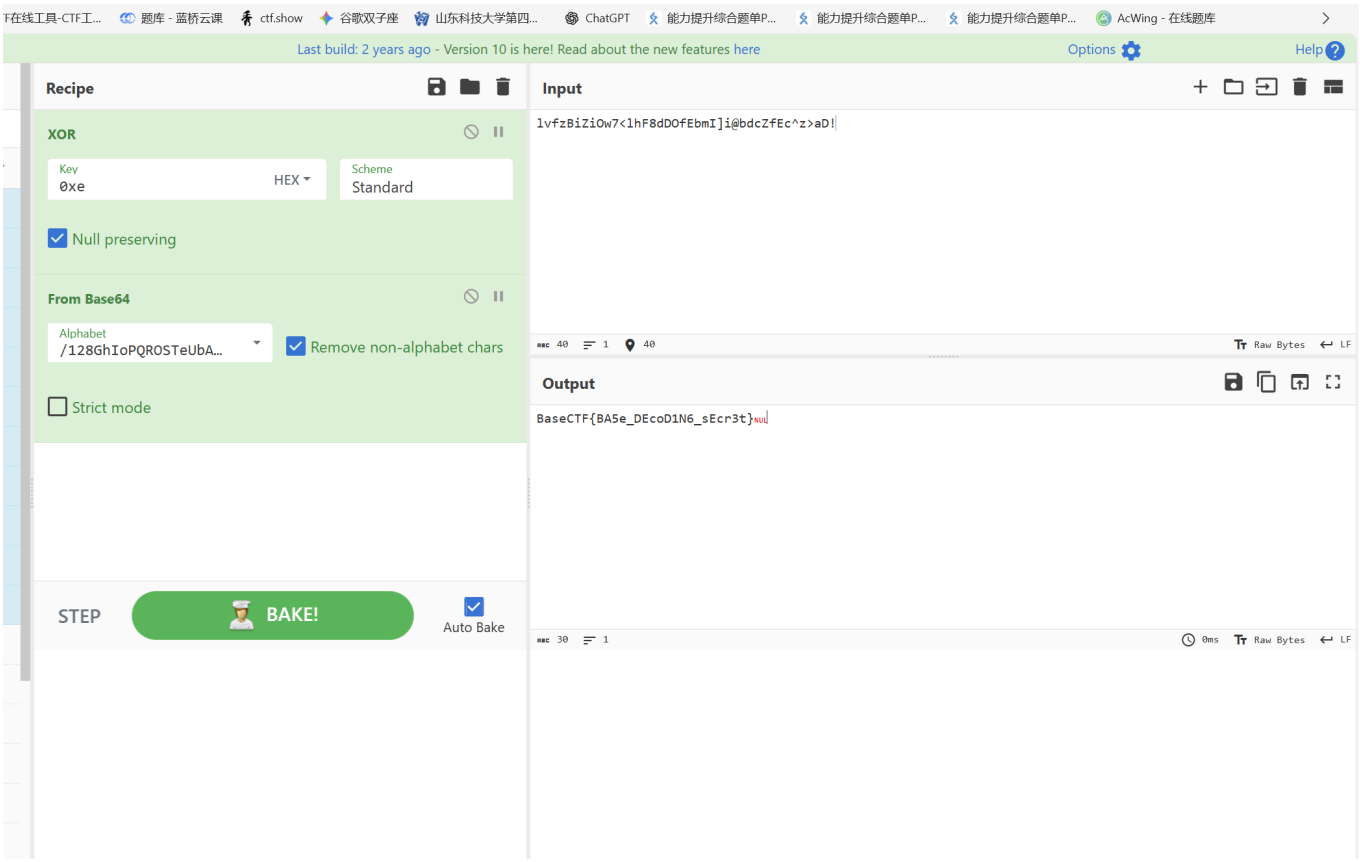
ida打开发现一串乱码类似base64,查看encode函数,发现base64解码的伪代码,但是每个字符在解码前都进行了异或处理,所以先异或每个字母再进行base64解码

IDA View-A | Pseudocode-A | Stack of main | He

```
1  int __fastcall main(int argc, const char **argv, const char **envp)
2  {
3    char Str1[2032]; // [rsp+20h] [rbp-FF8h] BYREF
4    _QWORD v5[257]; // [rsp+810h] [rbp-808h] BYREF
5
6    _main();
7    printf("Try typing something:");
8    scanf("%s", v5);
9    Encode(v5, Str1);
10   if ( !strcmp(Str1, "lvfzBiZiOw7<lhF8dDOfEbmI]i@bdcZfEc^z>aD!") )
11     printf("Wow, you actually got the flag");
12   else
13     printf("Maybe you need to try again?");
14   return 0;
15 }
```

```
38         do
39         {
40           v7 = v10;
41           *(&v14 + n2) = a1[v10 - 1];
42           v12 = (int)n2 <= 2;
43           v13 = v3 > (int)v10++;
44           ++n2;
45         }
46         while ( v13 && v12 );
47       }
48       v17[0] = Secret[(unsigned __int8)v15 >> 2];
49       v17[1] = Secret[(HIBYTE(v15) >> 4) | (16 * (_BYTE)v15) & 0x30];
50       v17[2] = Secret[(v16 >> 6) | (4 * HIBYTE(v15)) & 0x3C];
51       v17[3] = Secret[v16 & 0x3F];
52       n4_1 = n4_2;
53       do
54       {
55         *(_BYTE *)(p_Str1 + n4_1) = v4[n4_1] ^ 0xE;
56         ++n4_1;
57       }
58       while ( n4_1 != n4 );
59       LODWORD(result) = n4;
60       n4_2 += 4;
61       v4 -= 4;
62       n4 += 4;
63     }
64     while ( v3 > v7 );
65   }
66   result = (int)result;
67   *(_BYTE *)(p_Str1 + (int)result) = 0;
68   return result;
69 }
```

00000B95 Encode:26 (401595)



4,#### ez_maze

首先用die查看发现无壳,直接ida逆向分析,发现这大致是一个15*15的用wasd控制的迷宫游戏,要求最短路径才能游戏胜利

```
{
  n119 = (unsigned __int8)buf_[i];
  if ( n119 == 'd' )
  {
    if ( n209 % 15 == 14 )
      goto LABEL_20;
    ++n209;
  }
  else if ( (unsigned __int8)buf_[i] > 0x64u )
  {
    if ( n119 == 's' )
    {
      if ( n209 > 209 )
        goto LABEL_20;
      n209 += 15;
    }
    else
    {
      if ( n119 != 'w' )
      {
LABEL_21:
        j_puts(aInvalidInput);                  // "Invalid i
        return -1;
      }
      if ( n209 <= 14 )
        goto LABEL_20;
      n209 -= 15;
    }
  }
  else
  {
    if ( n119 != 'a' )
      goto LABEL_21;
    if ( !(n209 % 15) )
    {
000009F9 main:52 (4015F9)
```

shift+12后看到了矩阵,之后再用shift+f5提取这个矩阵,用脚本得出了最短到达目的的路径

```
maze = [
    8, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
```

```
    0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1,
    0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1,
    0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1,
    0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1,
    0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1,
    1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 9
]

visited = [0] * (15 * 15)  # 记录访问过的点


def BFS(maze, x, y):
    queue = [(x, y, '')]  # 设置队列，bfs用队列，dfs用栈
    while queue:
        x, y, path = queue.pop(0)
        if x < 15 and y < 15 and x >= 0 and y >= 0 and visited[x * 15 + y] !=
1 and maze[x * 15 + y] != 1:
            visited[x * 15 + y] = 1  # 证明已经访问过了
            queue.append((x + 1, y, path + 's'))  # 只能字符串相加
            queue.append((x, y - 1, path + 'a'))
            queue.append((x, y + 1, path + 'd'))
            queue.append((x - 1, y, path + 'w'))
        else:
            continue
        if maze[x * 15 + y] == 9:
            return path


flag = BFS(maze, 0, 0)
print(flag)
```

得出一串字符类似于md5的编码,尝试md5解码得出了flag