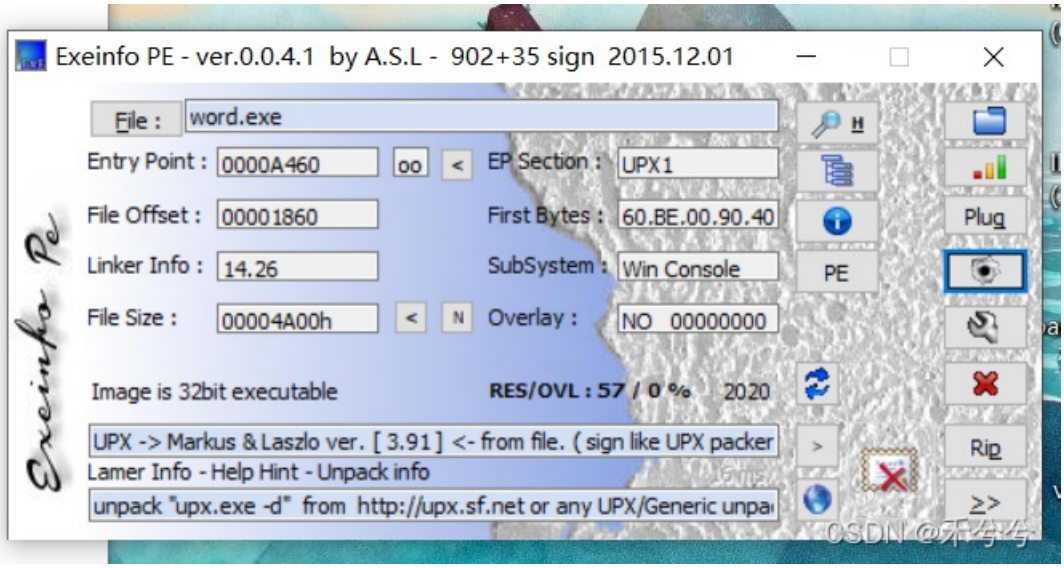
Tworld bugku writeup

原创

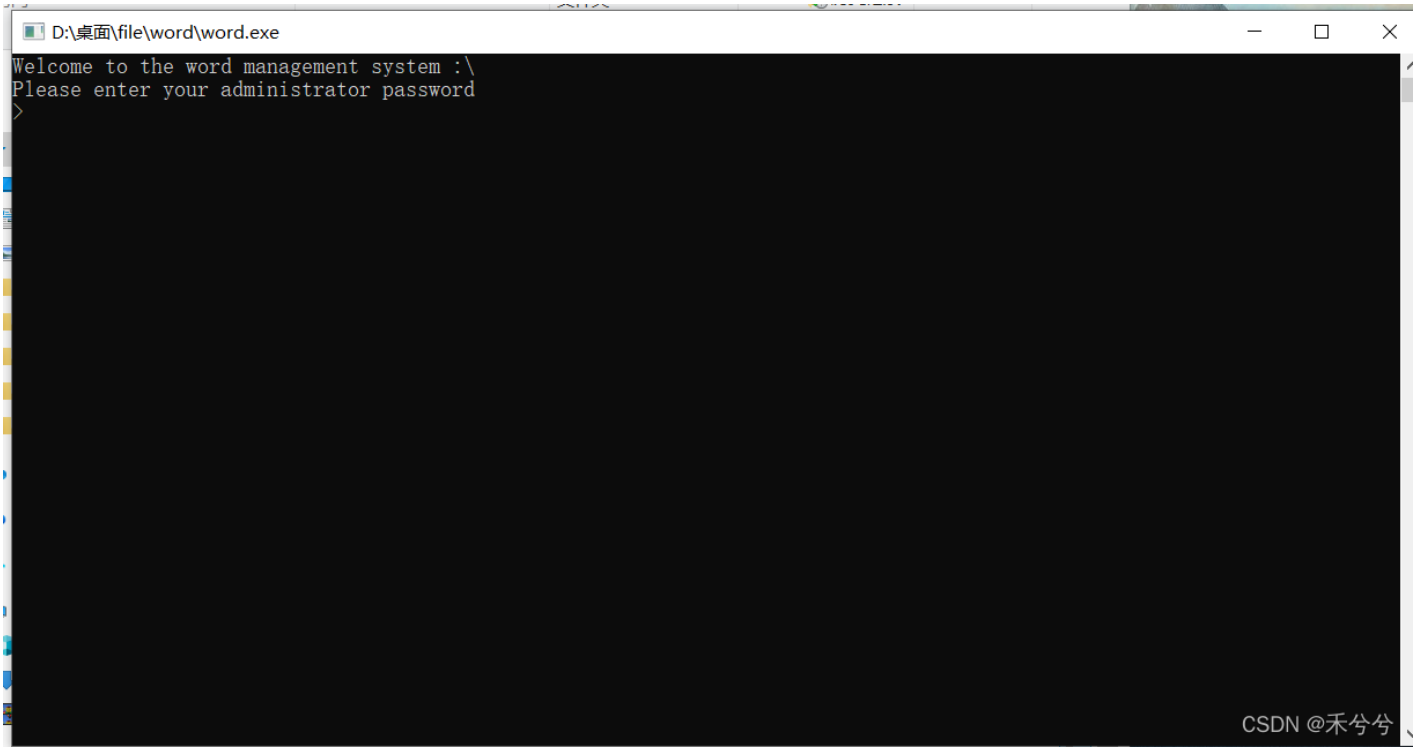
禾兮兮 于 2021-12-26 20:02:17 发布 179 收藏
文章标签： 其他 经验分享

ExeinfoP查壳，发现有upx壳，利用upx脱壳机脱壳



打开不能正常运行，查看了大佬的writeup Tworld-Bugku-Reverse - Moominn - 博客园 (cnblogs.com)

使用 StudyPE+ 固定 PE 基址,后可以正常运行



根据语句找到相应的函数

```
void __noreturn sub_4012A0()
{
  signed int v0; // eax
  HRSRC v1; // eax
  HRSRC v2; // esi
  DWORD v3; // ebx
  HGLOBAL v4; // eax
  const void *v5; // edi
  HANDLE v6; // esi
  signed int v7; // ecx
  DWORD NumberOfBytesWritten; // [esp+Ch] [ebp-70h]
  int v9; // [esp+10h] [ebp-6Ch]
  char v10[100]; // [esp+14h] [ebp-68h]

  sub_401020((int)"Welcome to the word management system :\\\n");
  sub_401020((int)"Please enter your administrator password\n>");
  sub_401050("%s", v10);
  v0 = 0;
  do
  {
    if ( (dword_4032E8[v0] ^ v10[v0]) != dword_4032D0[v0] )
    {
      sub_401020((int)"You're not an administrator :(\n");
      exit(0);
    }
    ++v0;
  }
  while ( v0 < 6 );
  sub_401020((int)"Hello administrator :)\n");
  while ( 1 )
  {
    sub_401020((int)"====================\n");
    sub_401020((int)"1.flag.doc\n");
    sub_401020((int)"2.Walk the maze\n");
    sub_401020((int)"3.exit\n");
```

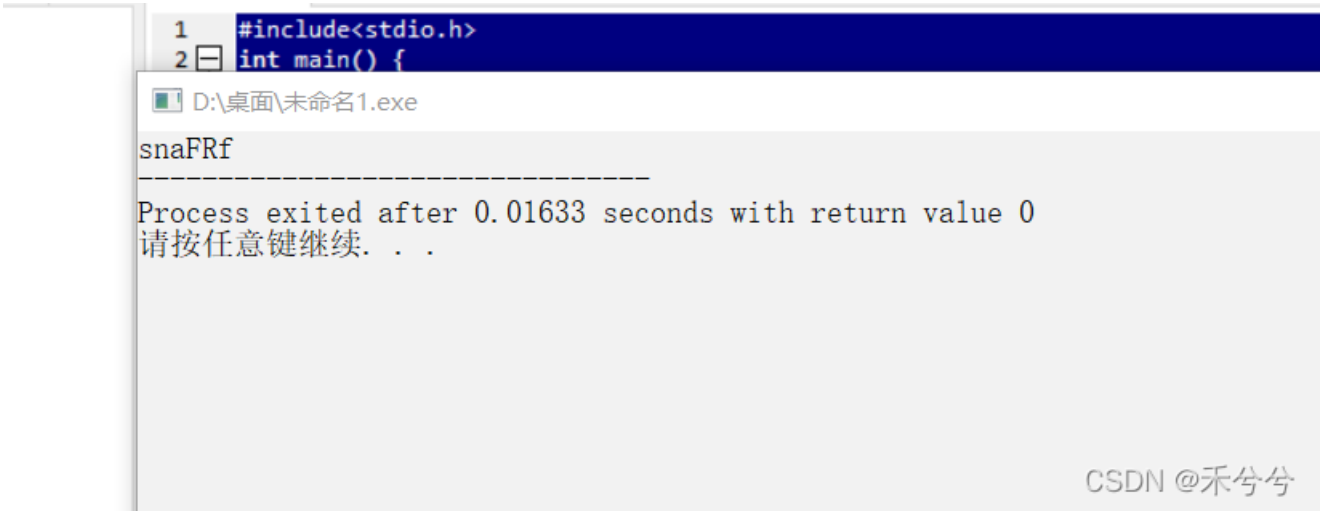
```
    sub_401020((int)"====================\n");
    sub_401050("%d", &v9);
    if ( v9 != 1 )
    {
      if ( v9 != 2 )
        exit(0);
      sub_401120();
    }
    v1 = FindResourceA(0, (LPCSTR)0x67, "doc");
    v2 = v1;
    if ( !v1 )
    {
      v7 = -1;
      goto LABEL_17;
    }
    v3 = SizeofResource(0, v1);
    v4 = LoadResource(0, v2);
    if ( !v4 )
    {
      v7 = -2;
      goto LABEL_17;
    }
    v5 = LockResource(v4);
    DeleteFileA("flag.doc");
    if ( PathFileExistsA("flag.doc") )
    {
      v7 = -3;
      goto LABEL_17;
    }
    v6 = CreateFileA("flag.doc", 0x10000000u, 0, 0, 1u, 0, 0);
    if ( v6 == (HANDLE)-1 )
      break;
    if ( !WriteFile(v6, v5, v3, &NumberOfBytesWritten, 0) )
    {
      v7 = -5;
LABEL_17:
      sub_401080(v7);
      exit(0);
    }
    CloseHandle(v6);
  }
  v7 = -4;
  goto LABEL_17;
}
```

找到admin的加密，进行破解

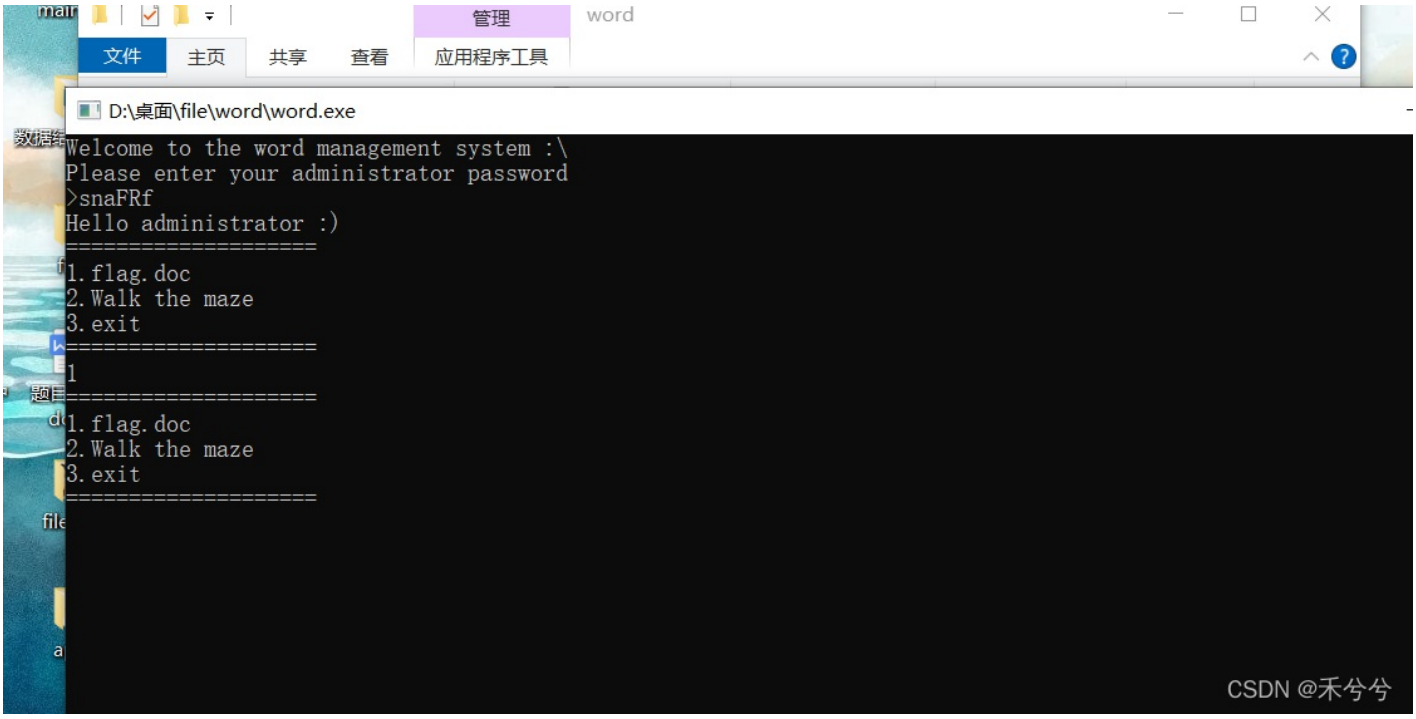
```
v0 = 0;
do
{
  if ( (dword_4032E8[v0] ^ v10[v0]) != dword_4032D0[v0] )
  {
    sub_401020((int)"You're not an administrator :(\n");
    exit(0);
  }
  ++v0;
}
while ( v0 < 6 );
sub_401020((int)"Hello administrator :)\n");
while ( 1 )
```

```
#include<stdio.h>
int main() {

  int a[] = {0xD0,0xCF,0x11,0xE0,0xA1,0xB1};
  int b[] = {0xA3,0xA1,0x70,0xA6,0xF3,0xD7};
  for (int i = 0; i < 6; i++)
      printf("%c",a[i]^b[i]);
}
```

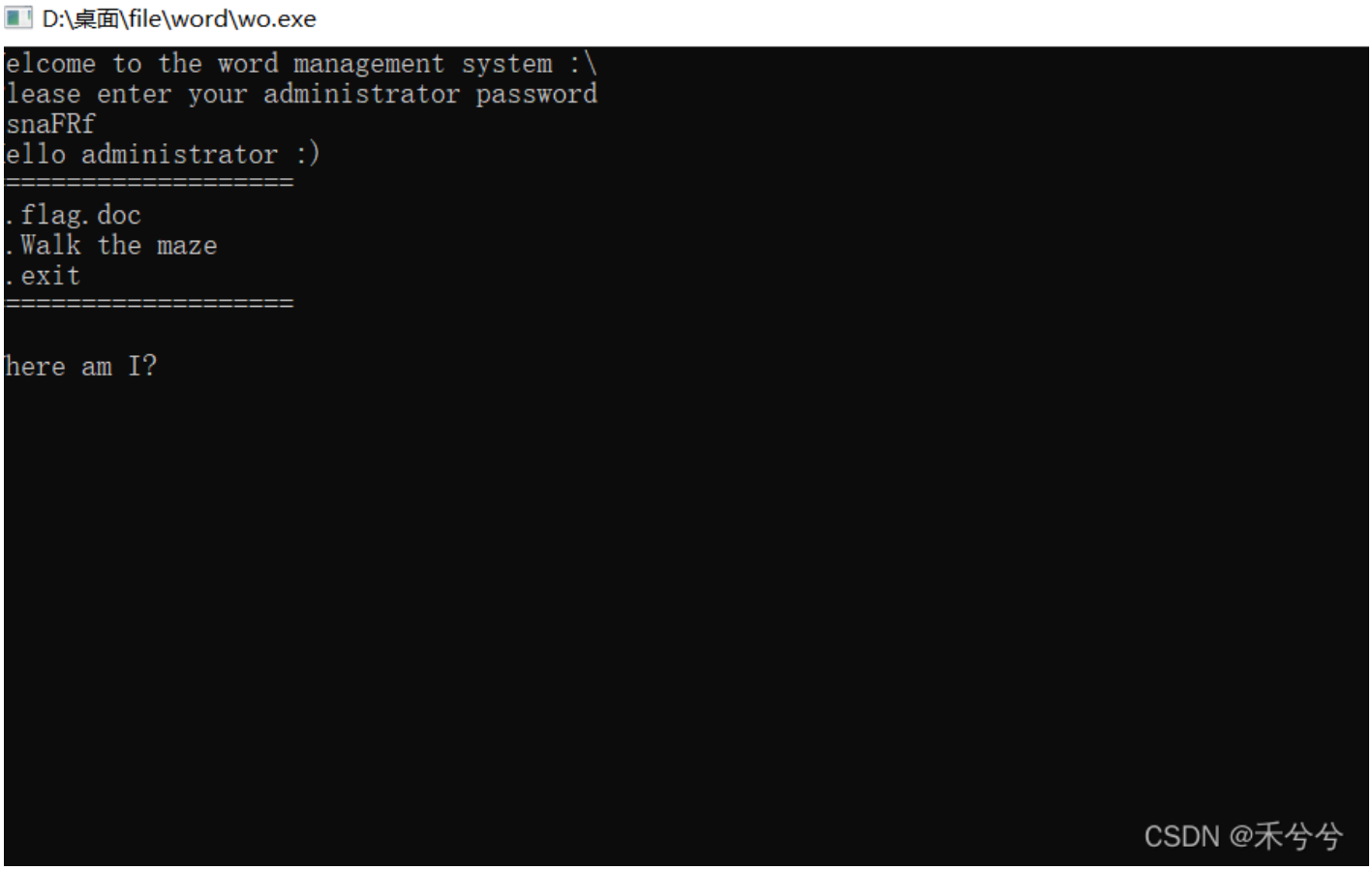


得到snaFRf



得的flag，但是发现flag文件是加密的

来到了第二项Walk the maze 走迷宫，输入得到关键字



找到函数

```
void __noreturn sub_401120()
{
  char v0; // cl
  signed int v1; // ebx
  signed int v2; // edx
  char *v3; // edi
  signed int v4; // esi
  int v5; // eax
  char v6; // al
  unsigned int v7; // eax
  int v8; // ecx
  __int128 v9; // [esp+Ch] [ebp-10Ch]
  __int128 v10; // [esp+1Ch] [ebp-FCh]
  __int128 v11; // [esp+2Ch] [ebp-ECh]
  __int128 v12; // [esp+3Ch] [ebp-DCh]
  char v13; // [esp+4Ch] [ebp-CCh]
  char v14; // [esp+4Dh] [ebp-CBh]
  char v15; // [esp+4Eh] [ebp-CAh]
  char v16[97]; // [esp+4Fh] [ebp-C9h]
  char v17; // [esp+B0h] [ebp-68h]
  char v18; // [esp+B1h] [ebp-67h]
  char v19; // [esp+B2h] [ebp-66h]
  char v20[97]; // [esp+B3h] [ebp-65h]

  sub_401020((int)"Where am I?\n");
  sub_401050("%s", &v17);
  v0 = v17;
  v1 = 0;
  v2 = 1;
  if ( v17 )
```

```
  {
    v3 = &v17;
    v4 = 12;
    do
    {
      switch ( v0 )
      {
        case 'w':
          v4 -= 12;
          break;
        case 'a':
          --v2;
          break;
        case 's':
          v4 += 12;
          break;
        case 'd':
          ++v2;
          break;
        default:
          goto LABEL_15;
      }
      v5 = dword_403300[v4 + v2];
      if ( v5 != '.' )
      {
        if ( v5 != 'd' )
          break;
        v1 = 1;
      }
      v6 = (v3++)[1];
      v0 = v6;
    }
    while ( v6 );
LABEL_15:
    if ( v1 )
    {
      v7 = 0;
      v9 = xmmword_403480;
      v10 = xmmword_403470;
      v11 = xmmword_403450;
      v12 = xmmword_403460;
      do
      {
        *(&v13 + v7) = *(&v17 + v7) ^ *((_BYTE *)&v9 + 4 * v7);
        *(&v14 + v7) = *(&v18 + v7) ^ *((_BYTE *)&v9 + 4 * v7 + 4);
        *(&v15 + v7) = *(&v19 + v7) ^ *((_BYTE *)&v9 + 4 * v7 + 8);
        v8 = *((unsigned __int8 *)&v9 + 4 * v7 + 12);
        LOBYTE(v8) = v20[v7] ^ v8;
        v16[v7] = v8;
        v7 += 4;
      }
      while ( v7 < 0x10 );
      if ( v7 < 0x64 )
      {
        *(&v13 + v7) = 0;
        sub_401020((int)&unk_4031C0, &v13);
        exit(0);
      }
      sub_401587(v8, v2);
    }
```

```
  }
  sub_401020((int)"I'm lost.");
  exit(0);
}
```

找到迷宫图：

```
11 00 00 00 E0 00 00 00  ................
A3 00 00 00 A1 00 00 00  ................
F3 00 00 00 D7 00 00 00  p...............
31 00 00 00 31 00 00 00  1...1...1...1...
31 00 00 00 31 00 00 00  1...1...1...1...
31 00 00 00 31 00 00 00  1...1...1...1...
31 00 00 00 2E 00 00 00  1...s...1.......
2E 00 00 00 2E 00 00 00  ................
31 00 00 00 31 00 00 00  ........1...1...
31 00 00 00 2E 00 00 00  1.......1.......
31 00 00 00 31 00 00 00  1...1...1...1...
31 00 00 00 31 00 00 00  1.......1...1...
31 00 00 00 2E 00 00 00  1.......1.......
31 00 00 00 31 00 00 00  1...1...1...1...
31 00 00 00 31 00 00 00  1.......1...1...
2E 00 00 00 2E 00 00 00  1...............
31 00 00 00 31 00 00 00  1...1...1...1...
31 00 00 00 31 00 00 00  1.......1...1...
31 00 00 00 31 00 00 00  1...1...1...1...
2E 00 00 00 2E 00 00 00  1...d...........
31 00 00 00 31 00 00 00  ........1...1...
31 00 00 00 31 00 00 00  1...1...1...1...
31 00 00 00 31 00 00 00  1...1...1...1...
31 00 00 00 31 00 00 00  1...1...1...1...
```


经过整理得到（把.换成了0）

111111111111

1s1000000011

101011111011

101011111011

100011111011

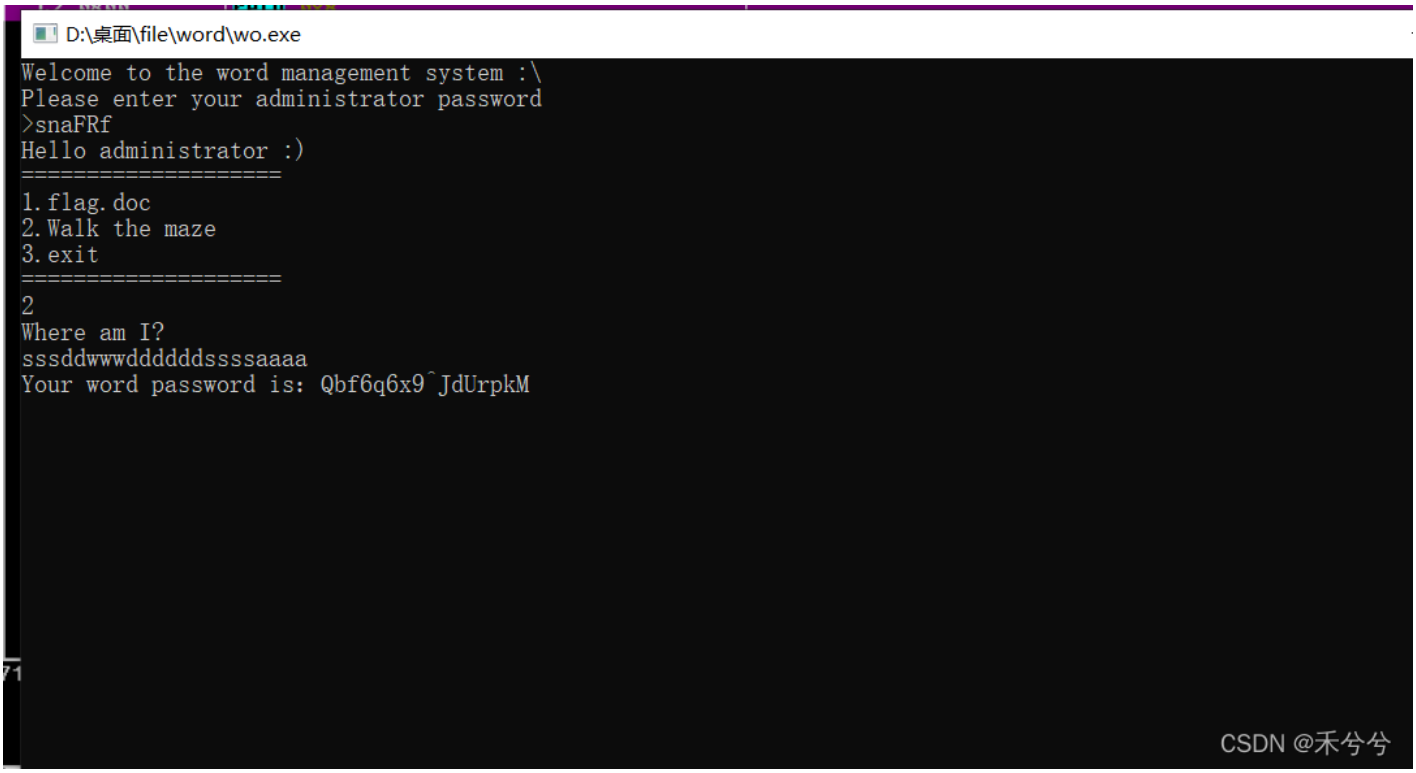
11111d000011

111111111111

sssddwwwddddddsssaaaa

走法为wsda，必须走.，到d结束，根据v4和v2的值可以知道我们是从s开始的

在原程序中输入闪退，可以在OD上输入，得到密码



Flag{Oh_my_God_this_one_is_so_trong}

唯一的不知道的是固定 PE 基址这个操作，看了挺多文章的，在这篇文章找到了我想要的答案

(38条消息) PE文件结构（五）基址重定位_billvsme的专栏-CSDN博客

按我的理解就是程序装载的地址与其他程序装载的地址相同或者起冲突。理论知识不够

然后我问了师兄，师兄说可能是upx脱壳过程中，代码从压缩到展开导致展开后的代码与原来的源码没有对齐，所以PE头没有回到原来的地方，我觉得这个比较有信服力。