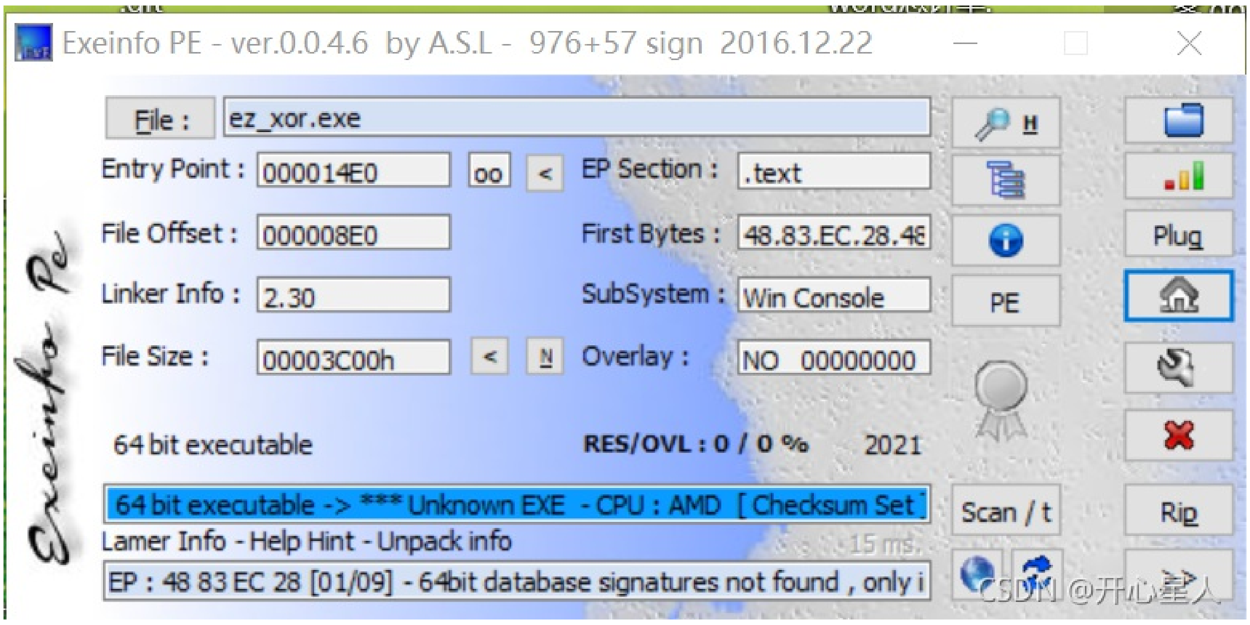
reverse ez_xor writeup

原创

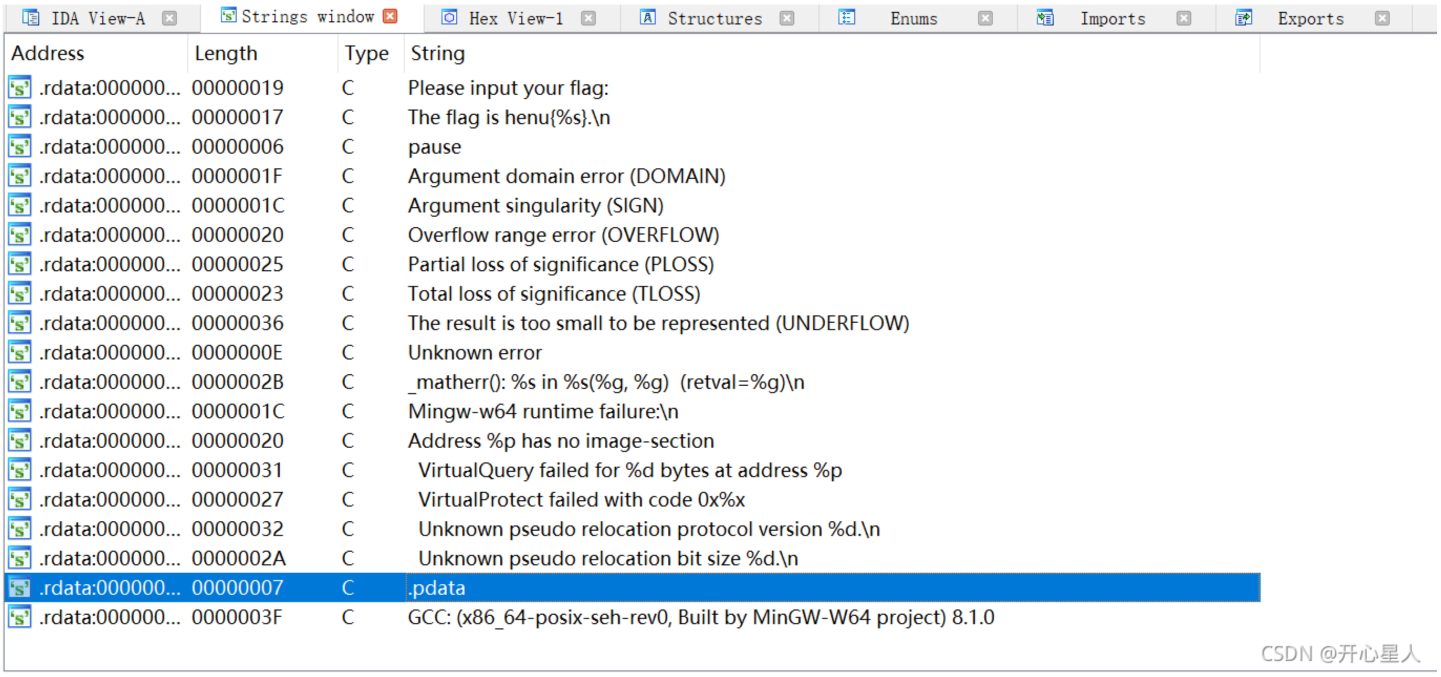
开心星人 于 2021-10-26 22:54:51 发布 50 收藏
文章标签： 经验分享

拿到ez_xor.exe附件直接丢进PE


Exeinfo PE - ver.0.0.4.6 by A.S.L - 976+57 sign 2016.12.22

File : ez_xor.exe
Entry Point : 000014E0   EP Section : .text
File Offset : 000008E0   First Bytes : 48.83.EC.28.48
Linker Info : 2.30   SubSystem : Win Console
File Size : 00003C00h   Overlay : NO 00000000
64 bit executable   RES/OVL : 0 / 0 %   2021
64 bit executable -> *** Unknown EXE - CPU : AMD [Checksum Set]
Lamer Info - Help Hint - Unpack info   15 ms.
EP : 48 83 EC 28 [01/09] - 64bit database signatures not found , only i

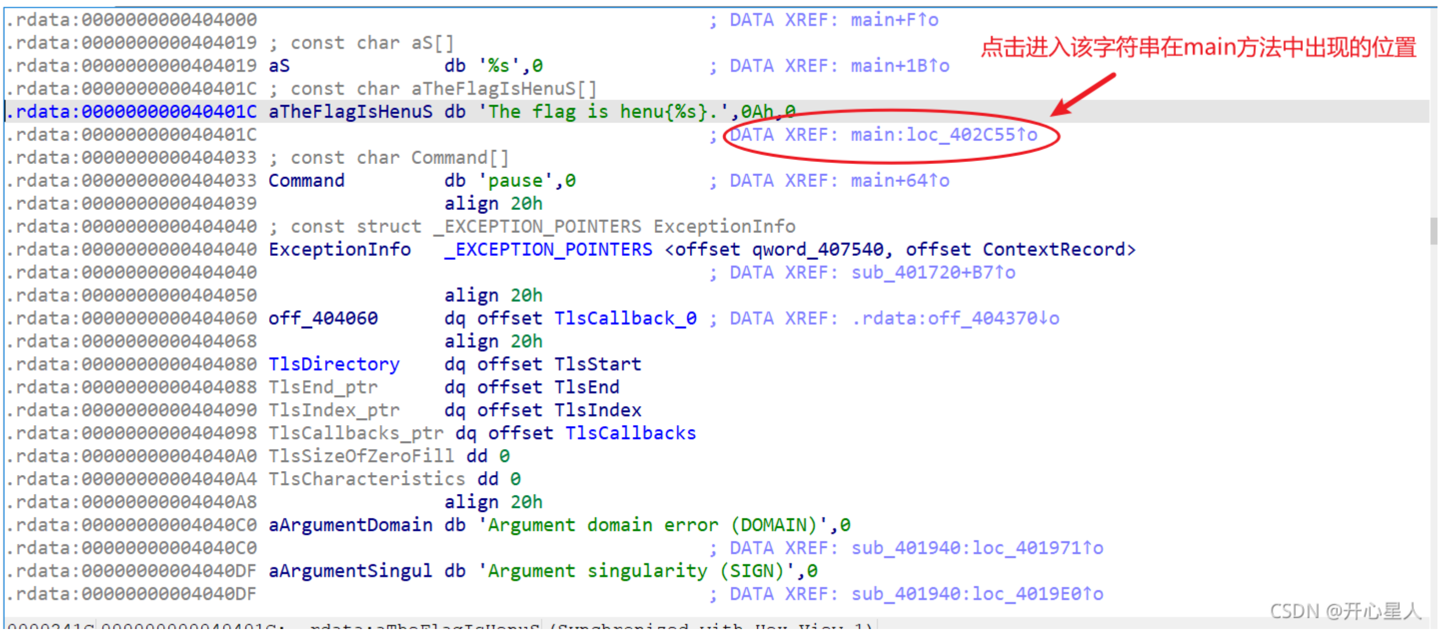
可以看到是64位exe文件，丢进ida64
Shift+F12查看字符串（如果是笔记本电脑的话，F12自带热键，先按Fn，即Fn+Shift+F12）



Address	Length	Type	String
.rdata:000000...	00000019	C	Please input your flag:
.rdata:000000...	00000017	C	The flag is henu{%s}.\n
.rdata:000000...	00000006	C	pause
.rdata:000000...	0000001F	C	Argument domain error (DOMAIN)
.rdata:000000...	0000001C	C	Argument singularity (SIGN)
.rdata:000000...	00000020	C	Overflow range error (OVERFLOW)
.rdata:000000...	00000025	C	Partial loss of significance (PLOSS)
.rdata:000000...	00000023	C	Total loss of significance (TLOSS)
.rdata:000000...	00000036	C	The result is too small to be represented (UNDERFLOW)
.rdata:000000...	0000000E	C	Unknown error
.rdata:000000...	0000002B	C	_matherr(): %s in %s(%g, %g) (retval=%g)\n
.rdata:000000...	0000001C	C	Mingw-w64 runtime failure:\n
.rdata:000000...	00000020	C	Address %p has no image-section
.rdata:000000...	00000031	C	VirtualQuery failed for %d bytes at address %p
.rdata:000000...	00000027	C	VirtualProtect failed with code 0x%x
.rdata:000000...	00000032	C	Unknown pseudo relocation protocol version %d.\n
.rdata:000000...	0000002A	C	Unknown pseudo relocation bit size %d.\n
.rdata:000000...	00000007	C	.pdata
.rdata:000000...	0000003F	C	GCC: (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0

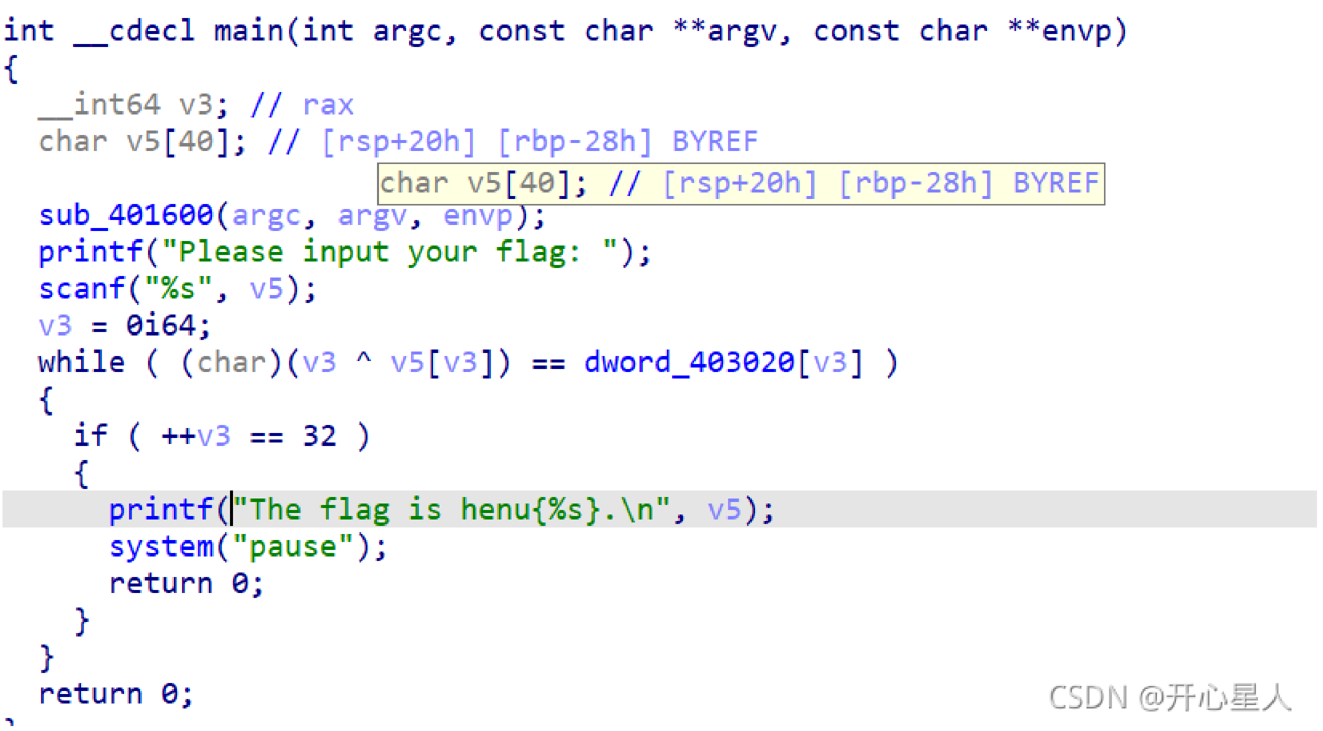
一般在这里找有没有和flag相关的字符串，可以看到这里是有的，双击进入

[illegible]



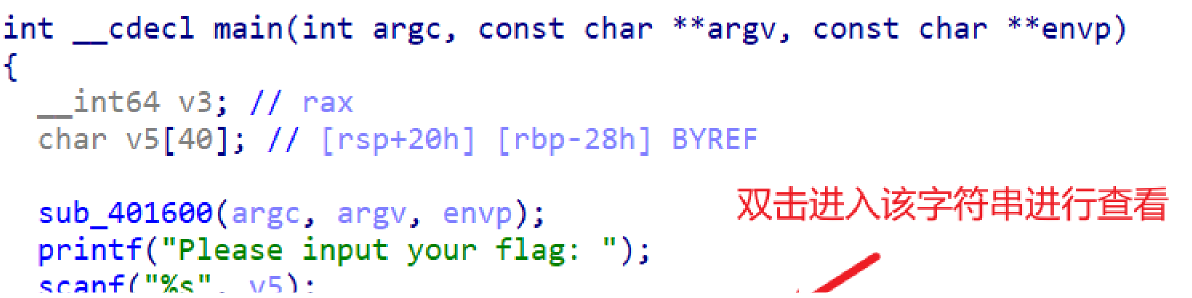
```
.rdata:0000000000404000                                         ; DATA XREF: main+F↑o
.rdata:0000000000404019 ; const char aS[]
.rdata:0000000000404019 aS              db '%s',0               ; DATA XREF: main+1B↑o
.rdata:000000000040401C ; const char aTheFlagIsHenuS[]
.rdata:000000000040401C aTheFlagIsHenuS db 'The flag is henu{%s}.',0Ah,0
.rdata:000000000040401C                                         ; DATA XREF: main:loc_402C55↑o
.rdata:0000000000404033 ; const char Command[]
.rdata:0000000000404033 Command         db 'pause',0            ; DATA XREF: main+64↑o
.rdata:0000000000404039                 align 20h
.rdata:0000000000404040 ; const struct _EXCEPTION_POINTERS ExceptionInfo
.rdata:0000000000404040 ExceptionInfo   _EXCEPTION_POINTERS <offset qword_407540, offset ContextRecord>
.rdata:0000000000404040                                         ; DATA XREF: sub_401720+B7↑o
.rdata:0000000000404050                 align 20h
.rdata:0000000000404060 off_404060      dq offset TlsCallback_0 ; DATA XREF: .rdata:off_404370↓o
.rdata:0000000000404068                 align 20h
.rdata:0000000000404080 TlsDirectory    dq offset TlsStart
.rdata:0000000000404088 TlsEnd_ptr      dq offset TlsEnd
.rdata:0000000000404090 TlsIndex_ptr    dq offset TlsIndex
.rdata:0000000000404098 TlsCallbacks_ptr dq offset TlsCallbacks
.rdata:00000000004040A0 TlsSizeOfZeroFill dd 0
.rdata:00000000004040A4 TlsCharacteristics dd 0
.rdata:00000000004040A8                 align 20h
.rdata:00000000004040C0 aArgumentDomain db 'Argument domain error (DOMAIN)',0
.rdata:00000000004040C0                                         ; DATA XREF: sub_401940:loc_401971↑o
.rdata:00000000004040DF aArgumentSingul db 'Argument singularity (SIGN)',0
.rdata:00000000004040DF                                         ; DATA XREF: sub_401940:loc_4019E0↑o
```

找到该字符串，点击上图所示，进入main方法
会进入流程图界面，按空格进入文本界面
可以看到汇编代码了，按F5（同理如果是笔记本记得按Fn+F5）反汇编，转换成C语言



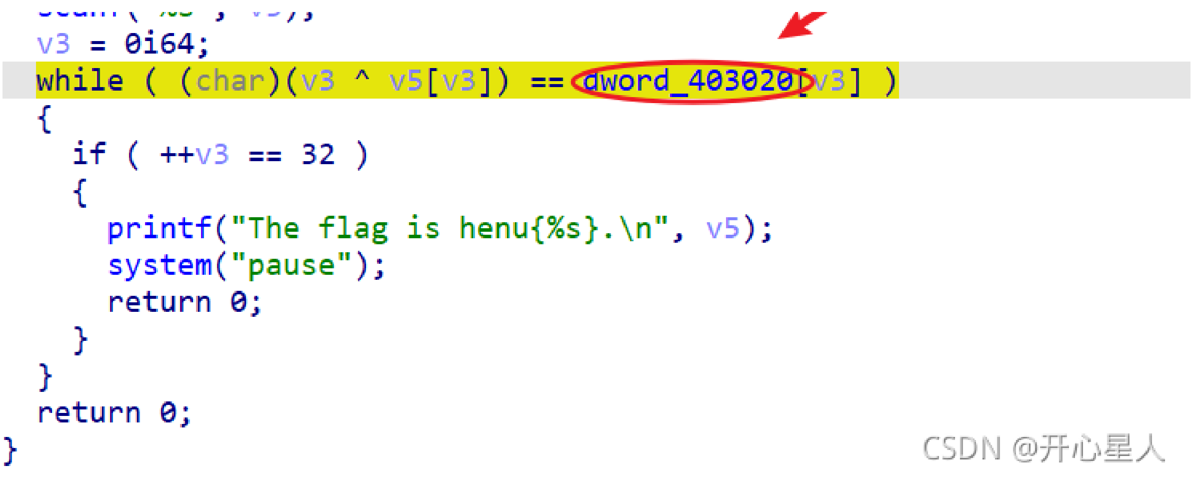
```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v3; // rax
  char v5[40]; // [rsp+20h] [rbp-28h] BYREF
                       char v5[40]; // [rsp+20h] [rbp-28h] BYREF
  sub_401600(argc, argv, envp);
  printf("Please input your flag: ");
  scanf("%s", v5);
  v3 = 0i64;
  while ( (char)(v3 ^ v5[v3]) == dword_403020[v3] )
  {
    if ( ++v3 == 32 )
    {
      printf("The flag is henu{%s}.\n", v5);
      system("pause");
      return 0;
    }
  }
  return 0;
```

现在就可以分析代码了，这里的C语言可能数据类型之类的会和我们平时的有点不一样
比如说这里的v3=0i64，0i64表示int64_t类型的0，其实就基本上可以理解为0
这里代码可以看到关键异或代码while ((char)(v3 ^ v5[v3]) == dword_403020[v3])

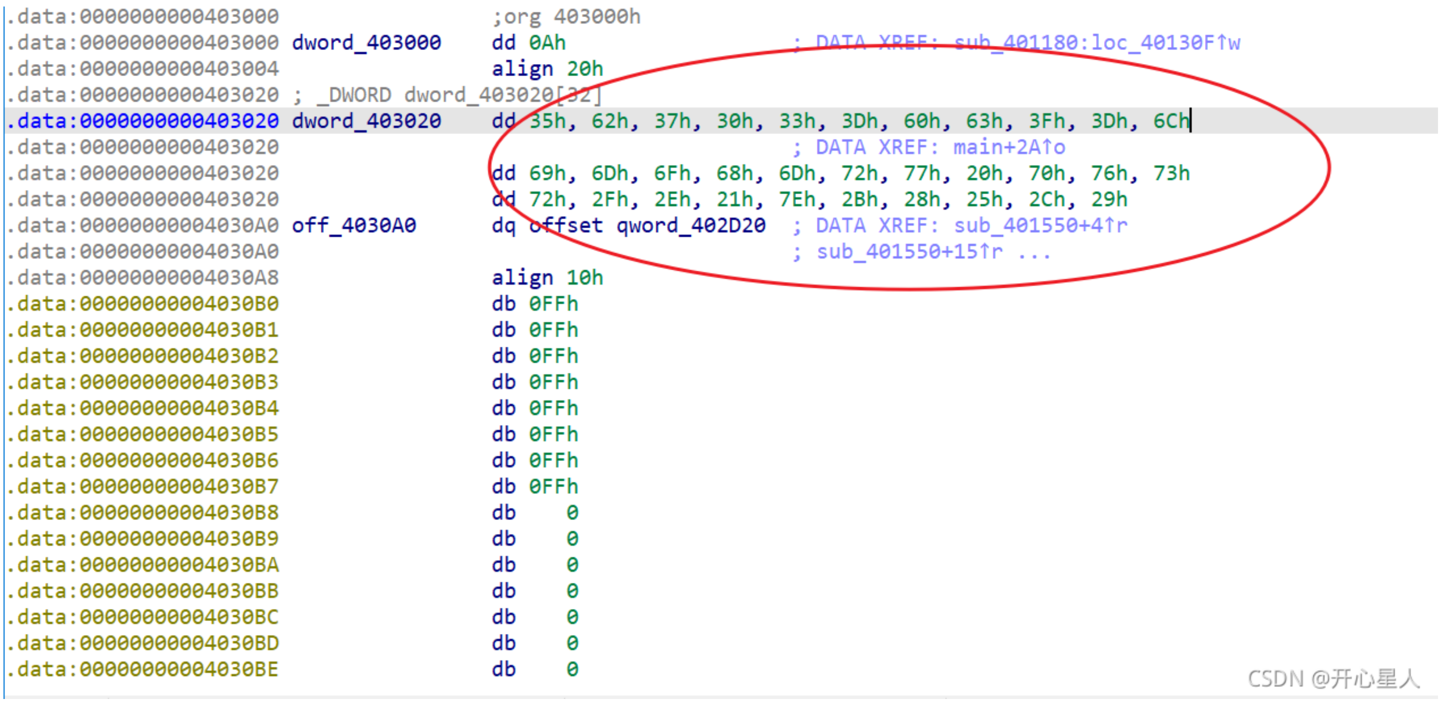


```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v3; // rax
  char v5[40]; // [rsp+20h] [rbp-28h] BYREF

  sub_401600(argc, argv, envp);
  printf("Please input your flag: ");
```



```
  v3 = 0i64;
  while ( (char)(v3 ^ v5[v3]) == dword_403020[v3] )
  {
    if ( ++v3 == 32 )
    {
      printf("The flag is henu{%s}.\n", v5);
      system("pause");
      return 0;
    }
  }
  return 0;
}
```



```
.data:0000000000403000                 ;org 403000h
.data:0000000000403000 dword_403000    dd 0Ah                  ; DATA XREF: sub_401180:loc_40130F↑w
.data:0000000000403004                 align 20h
.data:0000000000403020 ; _DWORD dword_403020[32]
.data:0000000000403020 dword_403020    dd 35h, 62h, 37h, 30h, 33h, 3Dh, 60h, 63h, 3Fh, 3Dh, 6Ch
.data:0000000000403020                                         ; DATA XREF: main+2A↑o
.data:0000000000403020                 dd 69h, 6Dh, 6Fh, 68h, 6Dh, 72h, 77h, 20h, 70h, 76h, 73h
.data:0000000000403020                 dd 72h, 2Fh, 2Eh, 21h, 7Eh, 2Bh, 28h, 25h, 2Ch, 29h
.data:00000000004030A0 off_4030A0      dq offset qword_402D20  ; DATA XREF: sub_401550+4↑r
.data:00000000004030A0                                         ; sub_401550+15↑r ...
.data:00000000004030A8                 align 10h
.data:00000000004030B0                 db 0FFh
.data:00000000004030B1                 db 0FFh
.data:00000000004030B2                 db 0FFh
.data:00000000004030B3                 db 0FFh
.data:00000000004030B4                 db 0FFh
.data:00000000004030B5                 db 0FFh
.data:00000000004030B6                 db 0FFh
.data:00000000004030B7                 db 0FFh
.data:00000000004030B8                 db    0
.data:00000000004030B9                 db    0
.data:00000000004030BA                 db    0
.data:00000000004030BB                 db    0
.data:00000000004030BC                 db    0
.data:00000000004030BD                 db    0
.data:00000000004030BE                 db    0
```

可以看到该字符串每个字符对应的ASCII码（这里按R键即可看到对应的字符）
现在已知dword_403020和v3（v3就是0~31），逐个进行异或即可得到flag

写一个Python脚本

```python
s=[0x35, 0x62, 0x37, 0x30, 0x33, 0x3D, 0x60, 0x63, 0x3F, 0x3D, 0x6C,0x69, 0x6D, 0x6F, 0x68, 0x6D, 0x72, 0x77, 0x
20, 0x70, 0x76, 0x73,0x72, 0x2F, 0x2E, 0x21, 0x7E, 0x2B, 0x28, 0x25, 0x2C, 0x29]
flag=[0 for i in range(32)] #从给出的代码很容易看到flag是32位的
for i in range(32):
    flag[i]=i^s[i]
print(flag)
```

即可得出flag