2022-03-17

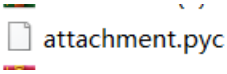
原创

无名函数 于 2022-03-17 23:00:53 发布 625 收藏

分类专栏： Buu-re 文章标签： list

本文链接：https://blog.csdn.net/m0_57291352/article/details/123536749

[GWCTF 2019]pyre



pyc，如此亲切啊

随便找一个python反编译的

```
# Embedded file name: encode.py
print 'Welcome to Re World!'
print 'Your input1 is your flag~'
l = len(input1)
for i in range(l):
    num = ((input1[i] + i) % 128 + 128) % 128
    code += num

for i in range(l - 1):
    code[i] = code[i] ^ code[i + 1]

print code
code = ['\x1f',
 '\x12',
 '\x1d',
 '(',
 '0',
 '4',
 '\x01',
 '\x06',
 '\x14',
 '4',
 ',',
 '\x1b',
 'U',
 '?',
 'o',
 '6',
 '*',
 ':',
 '\x01',
 'D',
 ';',
 '%',
 '\x13']
```

不是很长，主要是上面代码是用python2写的不是3

尝试逆一下

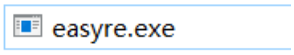
```
code = ['\x1f','\x12','\x1d','(', '0', '4', '\x01', '\x06', '\x14', '4', ',', '\x1b', 'U', '?', 'o', '6', '*', '
:', '\x01', 'D', ';', '%','\x13']
le = len(code)
for i in range(le-1):
    code[le-2-i] = chr(ord(code[le-2-i]) ^ ord(code[le-1-i]))

m = ''
for i in range(le):
    num = (ord(code[i])%128 - i ) % 128
    m += chr(num)
print(m,end='')
```

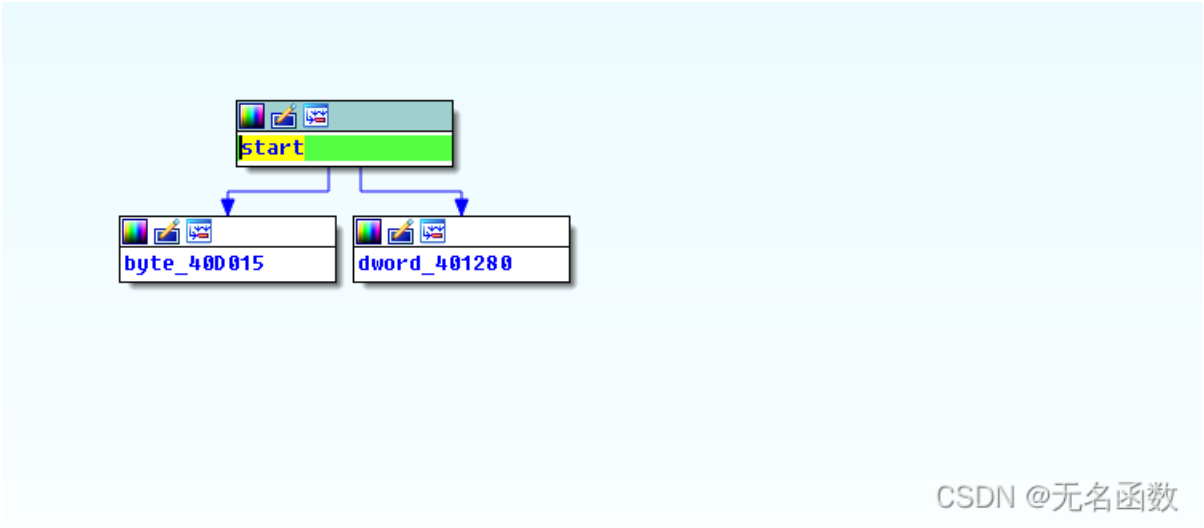
果然还是不能用str

运行得到`GWHT{Just_Re_1s_Ha66y!}`

[ACTF新生赛2020]easyre



IDA打开

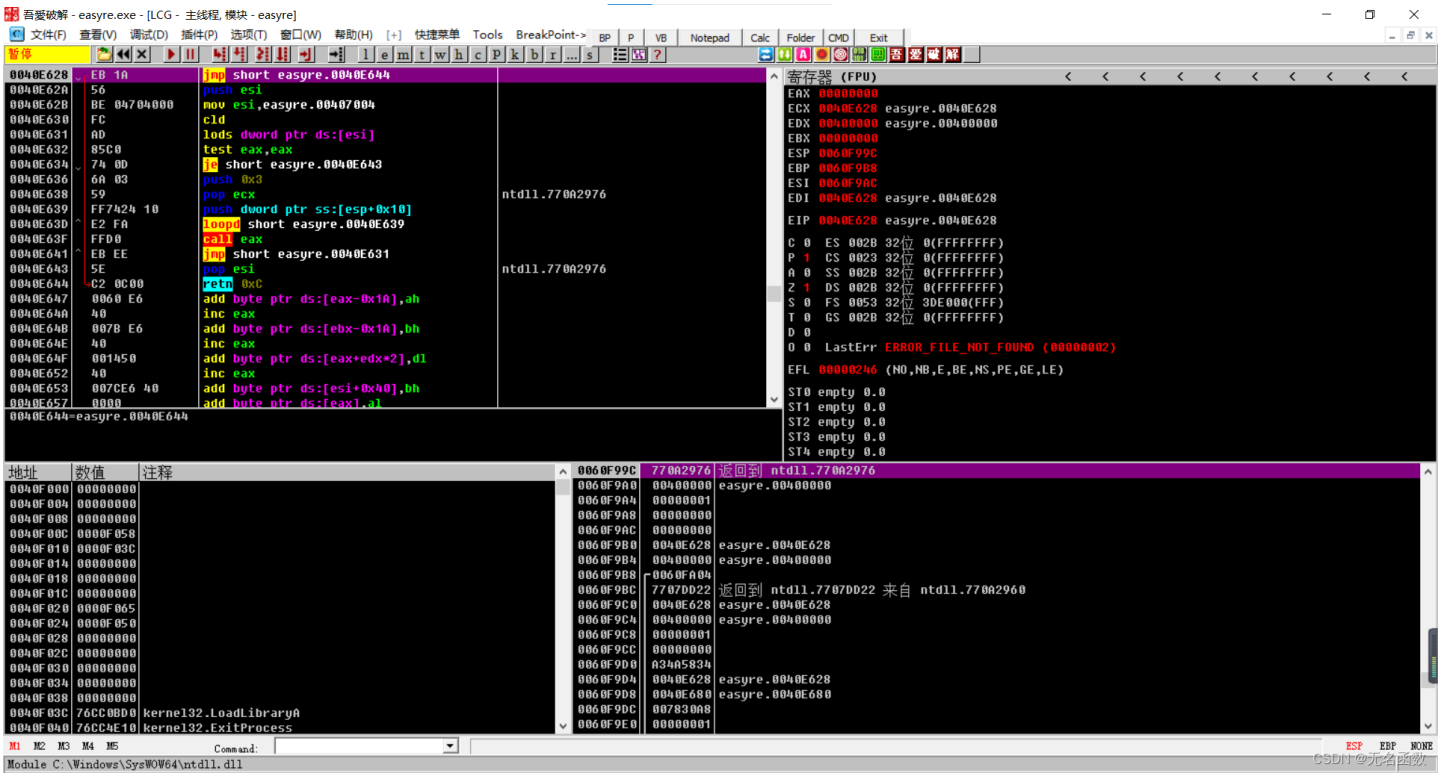


但是F5会报错，一开始还以为是64位和32位的问题，但是两个都尝试了都不能看伪代码

> UPX (the Ultimate Packer for eXecutables)是一款先进的可执行程序文件压缩器，压缩过的可执行文件体积缩小50%-70%，这样减少了磁盘占用空间、网络上传下载的时间和其它分布以及存储费用。 通过 UPX 压缩过的程序和程序库完全没有功能损失和压缩之前一样可正常地运行，对于支持的大多数格式没有运行时间或内存的不利后果。 UPX 支持许多不同的可执行文件格式 包含 Windows 95/98/ME/NT/2000/XP/CE 程序和动态链接库、DOS 程序、 Linux 可执行文件和核心。

难怪
这种情况之前遇到过，是在新年快乐里

用OD打开



脱壳之后返回IDA打开主函数

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v4; // [esp+12h] [ebp-2Eh]
  char v5; // [esp+13h] [ebp-2Dh]
  char v6; // [esp+14h] [ebp-2Ch]
  char v7; // [esp+15h] [ebp-2Bh]
  char v8; // [esp+16h] [ebp-2Ah]
  char v9; // [esp+17h] [ebp-29h]
  char v10; // [esp+18h] [ebp-28h]
  char v11; // [esp+19h] [ebp-27h]
  char v12; // [esp+1Ah] [ebp-26h]
  char v13; // [esp+1Bh] [ebp-25h]
  char v14; // [esp+1Ch] [ebp-24h]
  char v15; // [esp+1Dh] [ebp-23h]
  int v16; // [esp+1Eh] [ebp-22h]
  int v17; // [esp+22h] [ebp-1Eh]
  int v18; // [esp+26h] [ebp-1Ah]
  __int16 v19; // [esp+2Ah] [ebp-16h]
  char v20; // [esp+2Ch] [ebp-14h]
  char v21; // [esp+2Dh] [ebp-13h]
  char v22; // [esp+2Eh] [ebp-12h]
  int v23; // [esp+2Fh] [ebp-11h]
  int v24; // [esp+33h] [ebp-Dh]
  int v25; // [esp+37h] [ebp-9h]
  char v26; // [esp+3Bh] [ebp-5h]
  int i; // [esp+3Ch] [ebp-4h]

  __main();
  v4 = 42;
  v5 = 70;
  v6 = 39;
  v7 = 34;
  v8 = 78;
  v9 = 44;
  v10 = 34;
  v11 = 40;
  v12 = 73;
  v13 = 63;
  v14 = 43;
  v15 = 64;
  printf("Please input:");
  scanf("%s", &v19);
  if ( (_BYTE)v19 != 65 || HIBYTE(v19) != 67 || v20 != 84 || v21 != 70 || v22 != 123 || v26 != 125 )
    return 0;
  v16 = v23;
  v17 = v24;
  v18 = v25;
  for ( i = 0; i <= 11; ++i )
  {
    if ( *(&v4 + i) != _data_start__[*((char *)&v16 + i) - 1] )
      return 0;
  }
  printf("You are correct!");
  return 0;
}
```

逆函数还可以

```
key = '~}|{zyxwvutsrqponmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.-,+*)(\'&%$# !"'
encrypt = [42,70,39,34,78,44,34,40,73,63,43,64]
x = []
flag = ''
for i in encrypt:
  x.append(key.find(chr(i))+1)

for i in x:
  flag += chr(i)
print(flag)
```