XCTF的re1

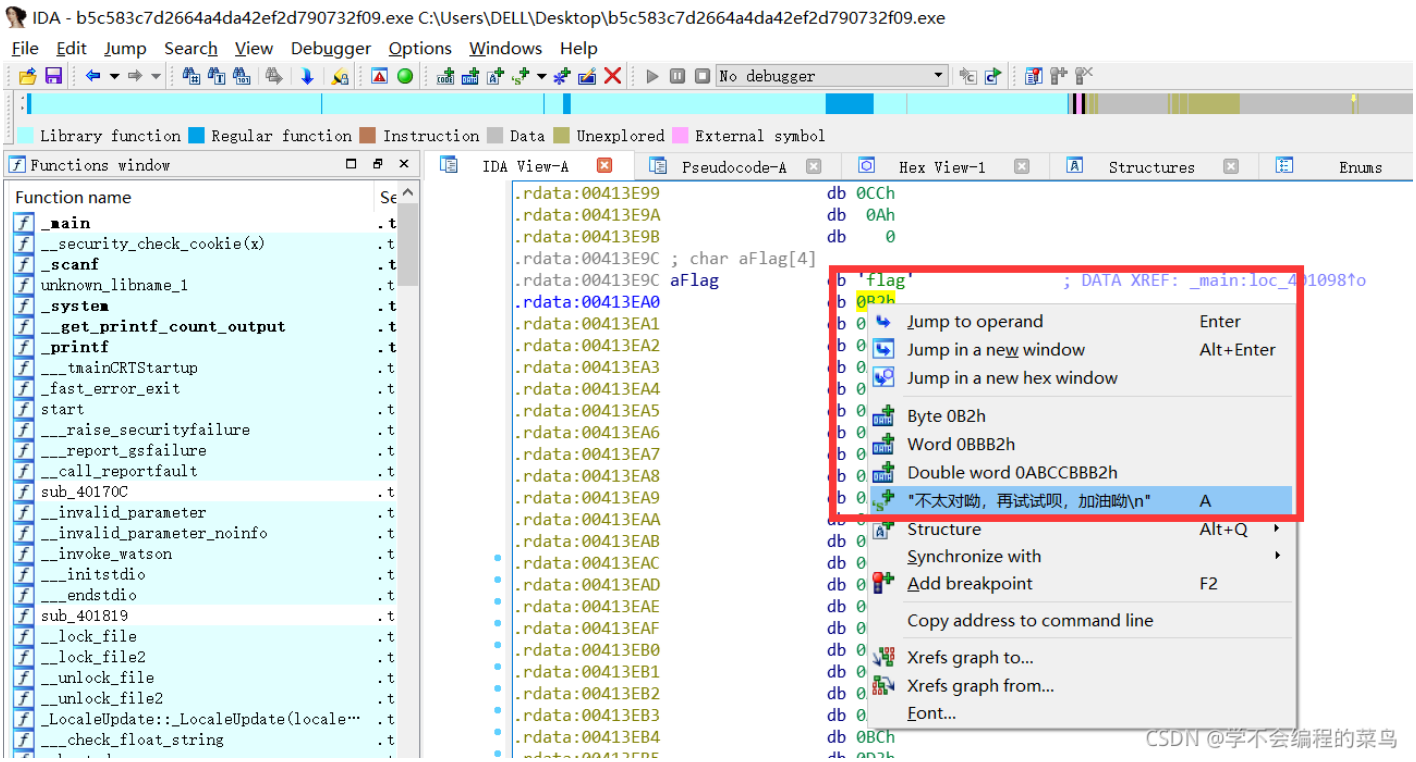
原创

why you learn hard? 于 2021-10-27 22:47:11 发布 44 收藏
分类专栏： re 文章标签： ctf

知识点1、解锁IDA新功能

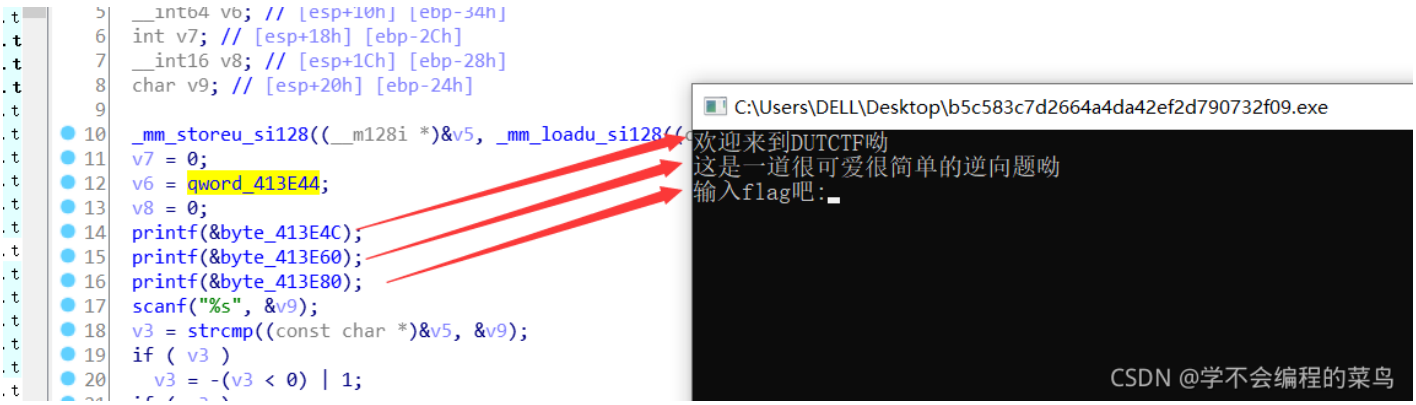
> 右键十六进制，直接就可查看连续内存地址存储的内容。就比如下面这一连串十六进制数字，看着好像flag，谁知道并不是。

```
.rdata:00413E9C aFlag           db 'flag'
.rdata:00413EA0                 db 0B2h
.rdata:00413EA1                 db 0BBh
.rdata:00413EA2                 db 0CCh
.rdata:00413EA3                 db 0ABh
.rdata:00413EA4                 db 0B6h
.rdata:00413EA5                 db 0D4h
.rdata:00413EA6                 db 0DFh
.rdata:00413EA7                 db 0CFh
.rdata:00413EA8                 db 0A3h
.rdata:00413EA9                 db 0ACh
.rdata:00413EAA                 db 0D4h
.rdata:00413EAB                 db 0D9h
.rdata:00413EAC                 db 0CAh
.rdata:00413EAD                 db 0D4h
.rdata:00413EAE                 db 0CAh
.rdata:00413EAF                 db 0D4h
.rdata:00413EB0                 db 0DFh
.rdata:00413EB1                 db 0C2h
.rdata:00413EB2                 db 0A3h
.rdata:00413EB3                 db 0ACh
.rdata:00413EB4                 db 0BCh
.rdata:00413EB5                 db 0D3h
.rdata:00413EB6                 db 0D3h
.rdata:00413EB7                 db 0CDh
.rdata:00413EB8                 db 0DFh
.rdata:00413EB9                 db 0CFh
.rdata:00413EBA                 db  0Ah
.rdata:00413EBB                 db     0
```



二、结合程序运行结果逆向。

不要以为人家给的那个文件 除了反编译之外啥用没有，就拿这个程序来讲，我们运行一下程序，再结合发编译源码，就知道那三块内存区域存储的东西对我们找flag来说没什么用。



然后这有个if判断语句，if后面的那个printf打印的是废话，那else后面应该就是flag内容，这是我猜测的。（但是并不是，他只是成功了之后提示flag get的字符串）。

```
scanf("%s", &v9);
v3 = strcmp((const char *)&v5, &v9);
if ( v3 )
  v3 = -(v3 < 0) | 1;
if ( v3 )
  printf(aFlag);
else
  printf((const char *)&unk_413E90);
```

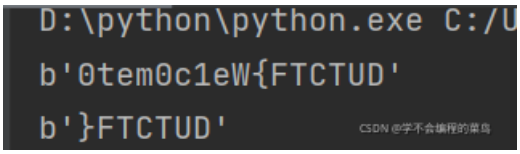
但是这有两块存储区域存储的东西很可疑，很值得怀疑，所以我们打开看看，发现是一堆十六进制串。

```
13E2C a1Qnan            db '1#QNAN',0           ; DATA XREF: _$110_OUTPUT:
13E33                   align 4
13E34 xmmword_413E34    xmmword 3074656D30633165577B465443545544h
13E34                                           ; DATA XREF: _main+10↑r
13E44 qword_413E44      dq 7D465443545544h      ; DATA XREF: _main+27↑r
```



```
    pw_str = '3074656D30633165577B465443545544'
    print(bytes.fromhex(pw_str))
    pw_str='7D465443545544'
    print(bytes.fromhex(pw_str))
```

出现下面这俩串字符，看着像反过来的。



三、知识点：小端序

> **上面那个字符串看着像反向的，因为我们知道他这个是小端序的存储方式，所以我们得从后面往前读。 我们将所有字符串分成两个一组，然后从后往前读取每一组。（切记不是一个字符一个字符读）**

写出exp

```
def Demo_05():
    pw_str = '3074656D30633165577B465443545544'
    for i in range(len(pw_str) - 2, -1, -2):
        print(chr(int(pw_str[i] + pw_str[i + 1], 16)), end='')
    pw_str = '7D465443545544'
    for i in range(len(pw_str) - 2, -1, -2):
        print(chr(int(pw_str[i] + pw_str[i + 1], 16)), end='')
```