XCTF easy_go

原创

夏了茶糜 于 2020-03-15 12:48:45 发布 451 收藏

分类专栏： CTF-REVERSE 文章标签： 安全

本文链接：https://blog.csdn.net/qin9800/article/details/104876528


```
pwn@VirtualBox:~/easyGo$ file easyGo
easyGo: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked,
 stripped
```

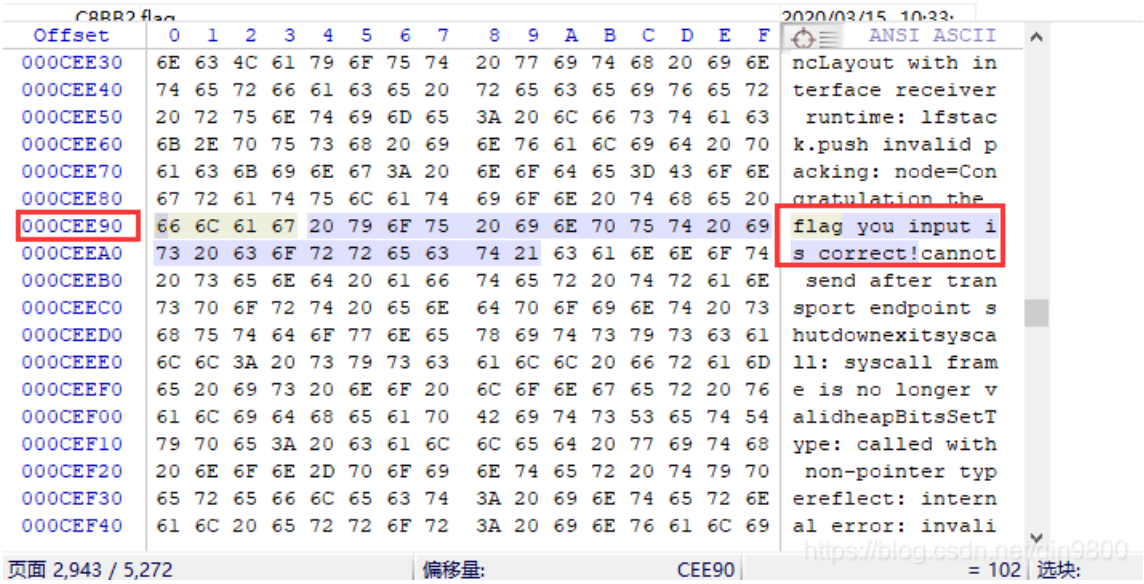
运行下看下

```
pwn@VirtualBox:~/easyGo$ ./easyGo
Please input you flag like flag{123} to judge:
flag{2222}
Try again! Come on!
pwn@VirtualBox:~/easyGo$
```

go语言程序，参考网上的wp，go语言无符号逆向，网上的wp大多数使用IDA脚本进行符号还原；我是来下，不懂为啥脚本在我的IDA7.0下都不正常。然后。。我就睡觉了

第二天起床又搜索了下，发现另一位大哥的方法很好，记录下

我在IDA中Shift+F12搜素关键字符串，没找到。大佬用WinHex搜索，学习下

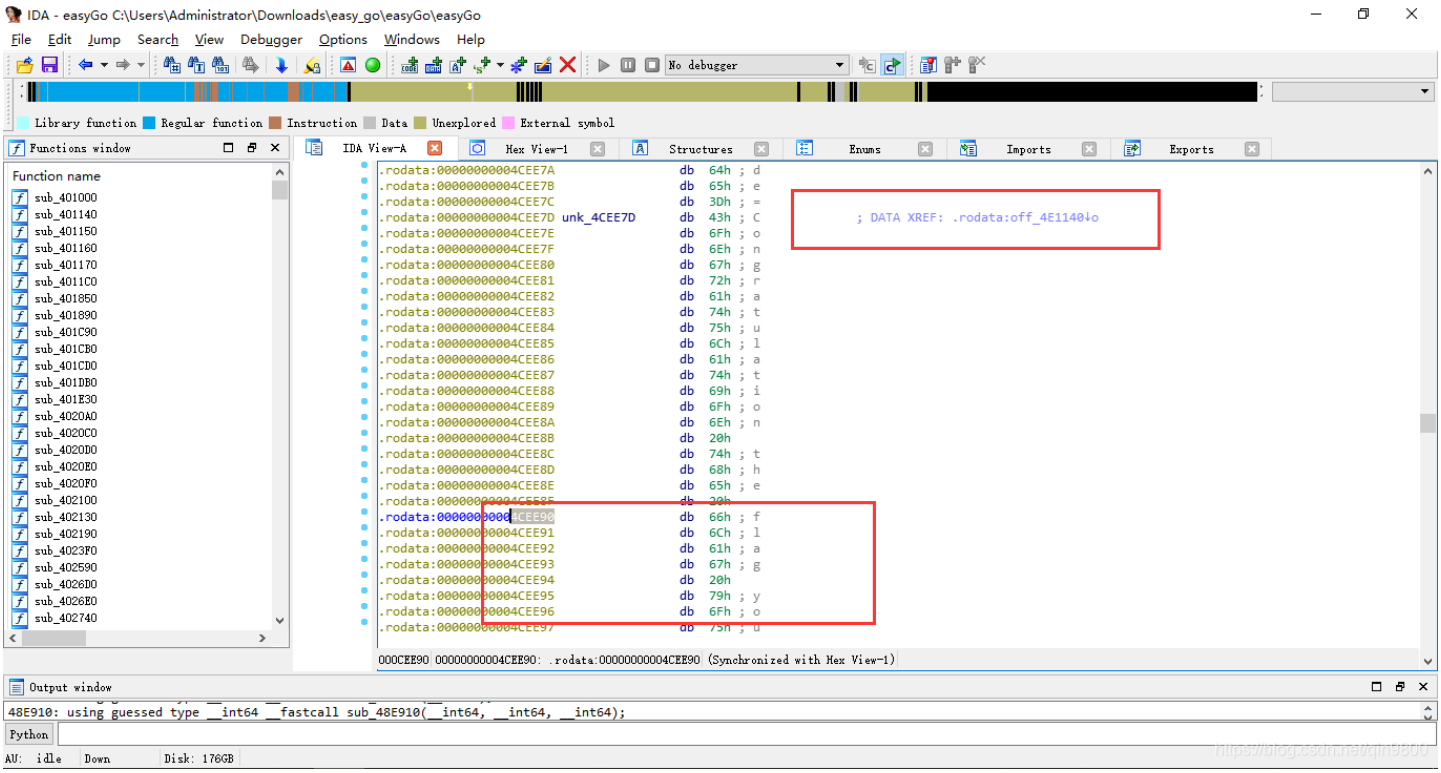


Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
000CEE30	6E	63	4C	61	79	6F	75	74	20	77	69	74	68	20	69	6E	ncLayout with in
000CEE40	74	65	72	66	61	63	65	20	72	65	63	65	69	76	65	72	terface receiver
000CEE50	20	72	75	6E	74	69	6D	65	3A	20	6C	66	73	74	61	63	runtime: lfstac
000CEE60	6B	2E	70	75	73	68	20	69	6E	76	61	6C	69	64	20	70	k.push invalid p
000CEE70	61	63	6B	69	6E	67	3A	20	6E	6F	64	65	3D	43	6F	6E	acking: node=Con
000CEE80	67	72	61	74	75	6C	61	74	69	6F	6E	20	74	68	65	20	gratulation the
000CEE90	66	6C	61	67	20	79	6F	75	20	69	6E	70	75	74	20	69	flag you input i
000CEEA0	73	20	63	6F	72	72	65	63	74	21	63	61	6E	6E	6F	74	s correct!cannot
000CEEB0	20	73	65	6E	64	20	61	66	74	65	72	20	74	72	61	6E	send after tran
000CEEC0	73	70	6F	72	74	20	65	6E	64	70	6F	69	6E	74	20	73	sport endpoint s
000CEED0	68	75	74	64	6F	77	6E	65	78	69	74	73	79	73	63	61	hutdownexitsysca
000CEEE0	6C	6C	3A	20	73	79	73	63	61	6C	6C	20	66	72	61	6D	ll: syscall fram
000CEEF0	65	20	69	73	20	6E	6F	20	6C	6F	6E	67	65	72	20	76	e is no longer v
000CEF00	61	6C	69	64	68	65	61	70	42	69	74	73	53	65	74	54	alidheapBitsSetT
000CEF10	79	70	65	3A	20	63	61	6C	6C	65	64	20	77	69	74	68	ype: called with
000CEF20	20	6E	6F	6E	2D	70	6F	69	6E	74	65	72	20	74	79	70	non-pointer typ
000CEF30	65	72	65	66	6C	65	63	74	3A	20	69	6E	74	65	72	6E	ereflect: intern
000CEF40	61	6C	20	65	72	72	6F	72	3A	20	69	6E	76	61	6C	69	al error: invali

页面 2,943 / 5,272 偏移量: CEE90 = 102 选块:

搜索到一个提示成功的字符串，我们拿它的文件偏移加一下IDA的基质，在IDA中定位到字符串

```
In [1]: hex(0x400000 + 0xcee90)
Out[1]: '0x4cee90'
```



定位到这里，发现上面的地方有个引用，跟过去看看

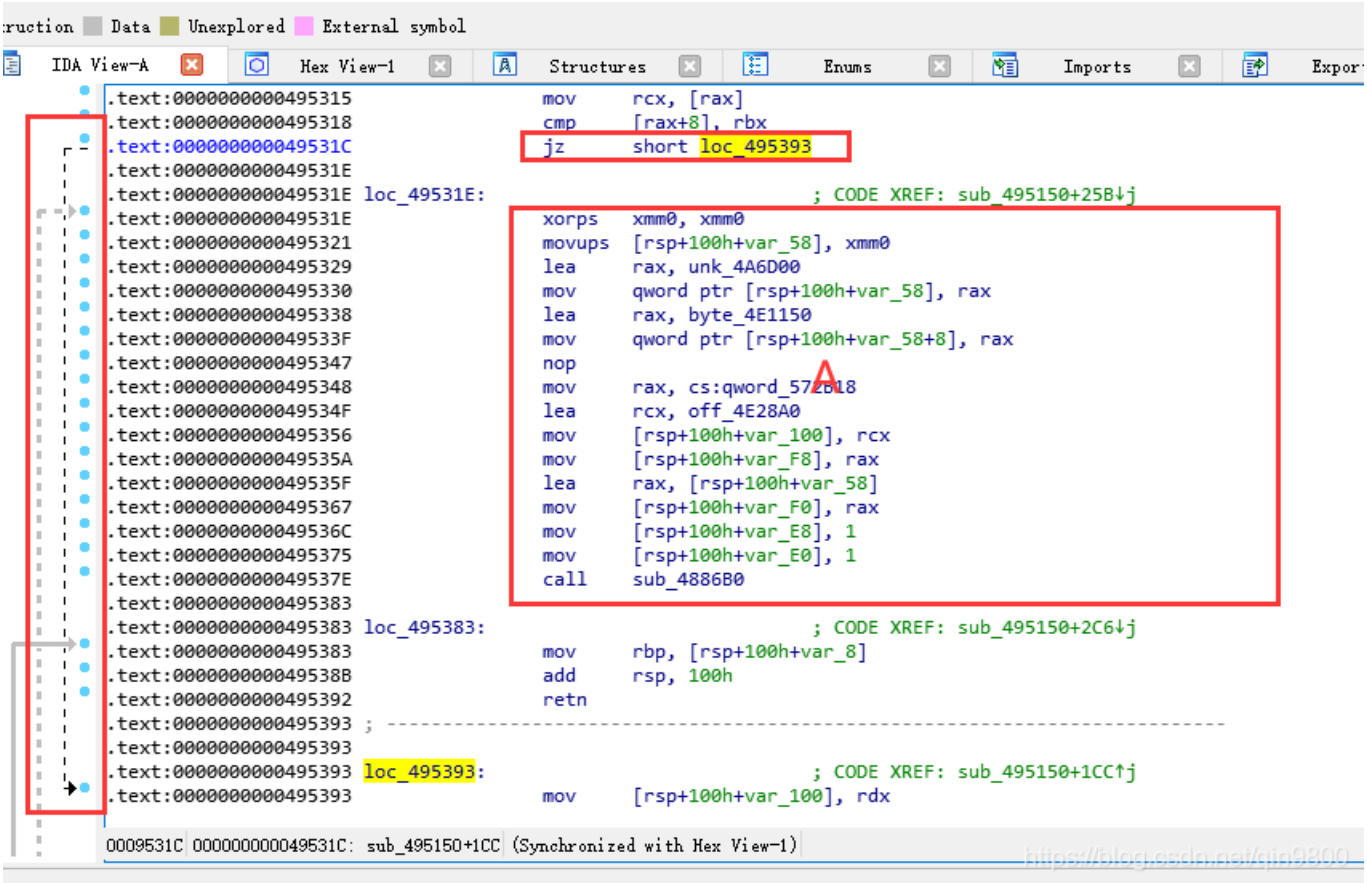
```
.rodata:00000000004E113C                db    0
.rodata:00000000004E113D                db    0
.rodata:00000000004E113E                db    0
.rodata:00000000004E113F                db    0
.rodata:00000000004E1140 off_4E1140     dq offset unk_4CEE7D    ; DATA XREF: sub_495150+27B↑o
.rodata:00000000004E1148                db  2Dh ; -
.rodata:00000000004E1149                db    0
.rodata:00000000004E114A                db    0
.rodata:00000000004E114B                db    0
.rodata:00000000004E114C                db    0
.rodata:00000000004E114D                db    0
.rodata:00000000004E114E                db    0
.rodata:00000000004E114F                db    0
.rodata:00000000004E1150 byte_4E1150    db 0BFh                 ; DATA XREF: sub_495150+1E8↑o
.rodata:00000000004E1151                db 0A6h
.rodata:00000000004E1152                db  4Ch ; L
.rodata:00000000004E1153                db    0
.rodata:00000000004E1154                db    0
.rodata:00000000004E1155                db    0
.rodata:00000000004E1156                db    0
.rodata:00000000004E1157                db    0
.rodata:00000000004E1158                db  13h
.rodata:00000000004E1159                db    0
.rodata:00000000004E115A                db    0
.rodata:00000000004E115B                db    0
.rodata:00000000004E115C                db    0
.rodata:00000000004E115D                db    0
.rodata:00000000004E115E                db    0
.rodata:00000000004E115F                db    0
.rodata:00000000004E1160 off_4E1160     dq offset unk_4CEF31    ; DATA XREF: sub_474070+3B4↑o
000E1149 00000000004E1149: .rodata:00000000004E1149 (Synchronized with Hex View-1)
```


还有引用，接着跟

```
.text:0000000000495393 ; ---------------------------------------------------------------------------
.text:0000000000495393
.text:0000000000495393 loc_495393:                             ; CODE XREF: sub_495150+1CC↑j
.text:0000000000495393                  mov     [rsp+100h+var_100], rdx
.text:0000000000495397                  mov     [rsp+100h+var_F8], rcx
.text:000000000049539C                  mov     [rsp+100h+var_F0], rbx
```

```
.text:00000000004953A1                call    sub_4023F0
.text:00000000004953A6                cmp     byte ptr [rsp+100h+var_E8], 0
.text:00000000004953AB                jz      loc_49531E
.text:00000000004953B1                xorps   xmm0, xmm0
.text:00000000004953B4                movups  [rsp+100h+var_48], xmm0
.text:00000000004953BC                lea     rax, unk_4A6D00
.text:00000000004953C3                mov     qword ptr [rsp+100h+var_48], rax
.text:00000000004953CB                lea     rax, off_4E1140
.text:00000000004953D2                mov     qword ptr [rsp+100h+var_48+8], rax
.text:00000000004953DA                nop
.text:00000000004953DB                mov     rax, cs:qword_572B18
.text:00000000004953E2                lea     rcx, off_4E28A0
.text:00000000004953E9                mov     [rsp+100h+var_100], rcx
.text:00000000004953ED                mov     [rsp+100h+var_F8], rax
.text:00000000004953F2                lea     rax, [rsp+100h+var_48]
.text:00000000004953FA                mov     [rsp+100h+var_F0], rax
.text:00000000004953FF                mov     [rsp+100h+var_E8], 1
.text:0000000000495408                mov     [rsp+100h+var_E0], 1
.text:0000000000495411                call    sub_4886B0
.text:0000000000495416                jmp     loc_495383
.text:000000000049541B ; ---------------------------------------------------------------------------
.text:000000000049541B
.text:000000000049541B loc_49541B:                            ; CODE XREF: sub_495150+1B7↑j
.text:000000000049541B                mov     [rsp+100h+var_68], rdx
000953CB 00000000004953CB: sub_495150+27B (Synchronized with Hex View-1)
```

跟到汇编代码出，猜测loc_495393这是个输出提示正确的分支，往上跟下，看是从哪跳过来的，



猜测A区域应该是提示错误的区域

0x495318处有一个CMP，猜测这里可能是关键，动调一下，在这里下个断点，看下值。

```
gdb-peda$ b *0x495318
Breakpoint 1 at 0x495318
gdb-peda$ r
Starting program: /home/pwn/easyGo/easyGo
[New LWP 9260]
[New LWP 9261]
Please input you flag like flag{123} to judge:
flag{88888}
```

这里根据我的分析，应该只是第一步判断用户输入的字符串长度是否位42，如果不是42则报错，如果是，应该是进入比较字符串的流程

这里已经不需要往下跟踪了，flag已经出来了，这次主要是学习了它这个定位关键代码的方法

flag

```
flag{92094daf-33c9-431e-a85a-8bfbd5df98ad}
```