writeup--echo server

原创

charlie_heng 于 2017-05-10 09:30:21 发布 252 收藏

分类专栏： 二进制-逆向工程

本文链接：https://blog.csdn.net/charlie_heng/article/details/71514304

最近在做逆向的题，在xctf实训平台上找了一下，挑了个比较简单的来做

题目描述：

Input the secret, output the flag.

下载地址

下载之后发现是elf，先拖ida里面看一下，发现很多个函数都识别不出来，都是出现call xxxxx+1的情况，这个很明显就是插入了E8或者其他东西造成ida识别错误，

首先看一下main函数

```
int __cdecl main()
{
  setbuf(stdin, 0);
  setbuf(stdout, 0);
  dword_804A088 = 1;
  puts("               ***************\n               Echo Server 0.3
  ((void (*)(void))((char *)&loc_80487C1 + 3))();
  return 0;
}
```

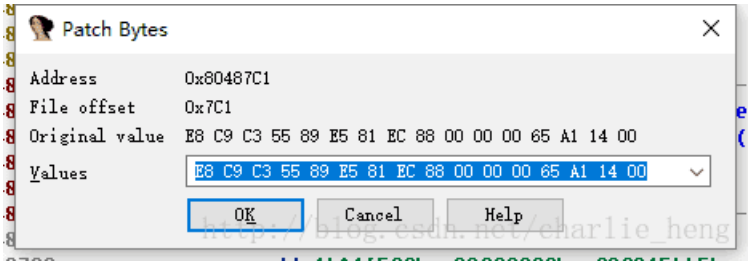

有一个loc_80487C1+3,双击跳转到对应代码段看一下

```
080487B8                call    _putchar
080487BD                jz      short near ptr loc_80487C1+1
080487BF                jnz     short near ptr loc_80487C1+1
080487C1
080487C1 loc_80487C1:                           ; CODE XREF: .text:080487BD↑j
080487C1                                        ; .text:080487BF↑j ...
080487C1                call    near ptr 915A4B8Fh
080487C6                in      eax, 81h        ; DMA page register 74LS612:
080487C6                                        ; Channel 2 (diskette DMA)  (address bits 16-23)
080487C8                in      al, dx
080487C9                mov     [eax], al
080487C9 ; ---------------------------------------------------------------------------
080487CB                db 0
080487CC                dd 14A16500h, 89000000h, 0C031F445h, 82444C7h, 14h, 42444C7h
080487CC                dd 0
080487E8                dd 8990458Dh, 0CDE82404h, 0EBFFFFFDh, 0C748C0FFh, 14082444h
080487E8                dd 8D000000h, 44899045h, 4C70424h, 24h, 0FDCEE800h, 0C033FFFFh
080487E8                dd 46E90774h, 41674031h, 2444C700h, 508h, 2444C700h, 4881704h
080487E8                dd 90458D08h, 0E8240489h, 0FFFFFE08h, 6775C085h, 0E02404C7h
080487E8                dd 0E8080489h, 0FFFFFDE8h, 4A088A1h, 74C08508h, 0EBB86615h
080487E8                dd 74C03105h, 4C7E8F9h, 124h, 0FDEAE800h, 458DFFFFh, 24048990h
080487E8                dd 0FFFD8FE8h, 2444C7FFh, 8, 24448900h, 90458D04h, 8901C083h
080487E8                dd 25E82404h, 89FFFFFDh, 44C78C45h, 100424h, 458B0000h
080487E8                dd 2404898Ch, 0FFFEBCE8h, 8D1DEBFFh, 4899045h, 0FD52E824h
080487E8                dd 0E883FFFFh, 24448901h, 90458D04h, 0E8240489h, 0FFFFFE9Dh
080487E8                dd 4A080A1h, 24048908h, 0FFFD23E8h, 0F4458BFFh, 14053365h
080487E8                dd 74000000h, 0FD42E805h, 0C3C9FFFFh
080488E0
080488E0 ; =============== S U B R O U T I N E =======================================
080488E0
080488E0 ; Attributes: bp-based frame
080488E0
080488E0 ; int __cdecl main(int, char **, char **)
080488E0 main           proc near               ; DATA XREF: start+17↑o
080488E0                push    ebp
```


发现全都是数据，这个时候在call near ptr 915A4B8Fh处按下D，将代码转化为数据

```
7BF ; ---------------------------------------------------------------------------
7C1                 db 0E8h
7C2 unk_80487C2     db 0C9h ;               ; CODE XREF: .text:080487BD↑j
7C2                                         ; .text:080487BF↑j
7C3                 db 0C3h ;
7C4                 db  55h ; U             ; CODE XREF: main+49↓p
7C5                 db  89h ;
7C6 ; ---------------------------------------
```


很明显这里是插入了一个E8让ida误以为这是一个函数调用
在080487C2处按下C，将数据转化为代码，再把下面的数据全部转换为代码。
在E8处点击一下，然后选择选择栏的Edit，再选择Patch program 的change byte
然后出现下面的对话框



将E8改成90，这里的90是nop的意思，就是什么都不干
然后在刚刚的Patch program那里选择Apply patchs to input file
再次打开ida，然后就会看到原来是数据的地方现在变成了代码

```
87C4 loc_80487C4:                               ; CODE XREF: main+49↓p
87C4                push    ebp
87C5                mov     ebp, esp
87C7                sub     esp, 88h
87CD                mov     eax, large gs:14h
87D3                mov     [ebp-0Ch], eax
87D6                xor     eax, eax
87D8                mov     dword ptr [esp+8], 14h
87E0                mov     dword ptr [esp+4], 0
87E8                lea     eax, [ebp-70h]
87EB                mov     [esp], eax
87EE                call    _memset
87F3
```


但是下面还是有类似的状况，然后全部按照上面的方法来改
但是这个时候发现有一个地方比较奇葩

```
3
3 loc_80487F3:                                  ; CODE XREF: .text:loc_80487F3↑j
3                   jmp     short near ptr loc_80487F3+1
3 - ---------------------------------------------------------------------------
```


自己跳转到自己+1的地方。。。这跟nop有什么区别。。。
果断把它变成数据，然后在0EB的下一个按C

```
E                   call    _memset
E ; ---------------------------------------------------------------------------
3                   db 0EBh
4 ; ---------------------------------------------------------------------------
4                   inc     eax
6                   dec     eax
7                   mov     dword ptr [esp+8], 14h
F                   lea     eax, [ebp-70h]
2                   mov     [esp+4], eax
6                   mov     dword ptr [esp], 0
D                   call    _read
2                   xor     eax, eax
4                   jz      short loc_804881D
```


嗯，这样好看多了
但是多出一个0EB怎么办？果断nop掉他，和上面改E8一样改就行
接着看下面，发现

```
:08048812                   xor     eax, eax
:08048814                   jz      short loc_804881D
:08048816                   jmp     near ptr 6F44B961h
:0804881B ; ---------------------------------------------------------------------------
```


xor eax,eax，这样eax无论是什么值都会变成0
然后再jz跳转，其实就相当于无条件跳转，下面的jmp之类的都是无效的代码。
但是把下面的代码改成数据之后发现一个熟悉的字符串

```
8812                xor     eax, eax
8814                jz      short loc_804881D
8814 ; ---------------------------------------------------------------------------
8816                db 0E9h
8817                db  46h ; F
8818                db  31h ; 1
8819                db  40h ; @
881A                db  67h ; g
881B                db 41h
881C                db 0
881D ; ---------------------------------------------------------------------------
```


再按一下A，把数据改成字符串

```
t:08048814                 jz      short loc_804881D
t:08048814 ; ---------------------------------------------------------------------------
t:08048816                 db 0E9h
t:08048817 aF1@ga          db 'F1@gA',0
t:0804881D ; ---------------------------------------------------------------------------
t:0804881D
t:0804881D loc_804881D:                            ; CODE XREF: .text:08048814↑j
t:0804881D                 mov     dword ptr [esp+8], 5
t:08048825                 mov     dword ptr [esp+4], 8048817h
t:0804882D                 lea     eax, [ebp-70h]
```

果然是flag。。但是其实在我第一次尝试的时候我是直接把这里给nop掉，然后后面反汇编成c语言的时候发现要输入一个字符串和一个已有的字符串比较，但是我给nop掉了。。。所以那个地址就很迷，然后后面再试的时候就发现这里是在代码段存了一个字符串。。。但是很明显在代码段存的字符串是不能用的，那么我们去数据段找一下吧

```
rodata:080489D5 ; char a02x[]
rodata:080489D5 a02x            db '%02X',0             ; DATA XREF: sub_804875D+3C↑o
rodata:080489DA aF1@g_          db 'F1@g_',0
rodata:080489E0 aYouAreVeryClos db 'You are very close! Now patch me~',0
rodata:080489E0                                         ; DATA XREF: .text:0804883C↑o
```

看来是提前给好了一个flag，但是看了一下，两个字符串是不同的，很明显是要上面那一个，那么改一下，选择patch program那里的change byte

把

46 31 40 67 5F 00 59 6F 75 20 61 72 65 20 76 65

改成

46 31 40 67 41 00 59 6F 75 20 61 72 65 20 76 65

然后把下面mov dword ptr[esp+4] , 8048817h的8048817改成数据段的flag的地址

点击这一行代码，选择change byte

把

C7 44 24 04 17 88 04 08 8D 45 90 89 04 24 E8 08

改成

C7 44 24 04 DA 89 04 08 8D 45 90 89 04 24 E8 08

再把上面的那个E9和F1@gA给nop掉

```
t:0804880D                 call    _read
t:08048812                 xor     eax, eax
t:08048814                 jz      short loc_804881D
t:08048816                 nop
t:08048817                 nop
t:08048818                 nop
t:08048819                 nop
t:0804881A                 nop
t:0804881B                 nop
t:0804881C                 nop
t:0804881D
t:0804881D loc_804881D:                            ; CODE XREF: .text:08048814↑j
t:0804881D                 mov     dword ptr [esp+8], 5
t:08048825                 mov     dword ptr [esp+4], offset aF1@g_ ; "F1@gA"
t:0804882D                 lea     eax, [ebp-70h]
t:08048830                 mov     [esp], eax
t:08048833                 call    _strncmp
t:08048838                 test    eax, eax
t:0804883A                 jnz     short loc_80488A3
t:0804883C                 mov     dword ptr [esp], offset aYouAreVeryClos ; "You are very close! Now patch
```

嗯，好看多了
下面还有一段是乱码的

```
048851
048851 loc_8048851:                            ; CODE XREF: .text:08048857↓j
048851                  mov     ax, 5EBh
048855                  xor     eax, eax
048857                  jz      short near ptr loc_8048851+1
048859                  call    near ptr 9288D25h
048859 ; ---------------------------------------------------------------------------
04885E                  db    0
04885F                  db    0
048860                  db    0
048861                  db 0E8h ;
048862                  db 0EAh ;
048863                  db 0FDh ;
048864                  db 0FFh
048865                  db 0FFh
048866 ; ---------------------------------------------------------------------------
048866
```


但是看了半天，发现并没有其他代码是跳转到这里的，那就直接nop掉吧。。。
最后变成这样

```
ext:0804883A                  jnz     short loc_80488A3
ext:0804883C                  mov     dword ptr [esp], offset aYouAreVeryClos ; "You are very close! Now patch me~"
ext:08048843                  call    _puts
ext:08048848                  mov     eax, ds:dword_804A088
ext:0804884D                  test    eax, eax
ext:0804884F                  jz      short loc_8048866
ext:08048851                  nop
ext:08048852                  nop
ext:08048853                  nop
ext:08048854                  nop
ext:08048855                  nop
ext:08048856                  nop
ext:08048857                  nop
ext:08048858                  nop
ext:08048859                  nop
ext:0804885A                  nop
ext:0804885B                  nop
ext:0804885C                  nop
ext:0804885D                  nop
ext:0804885E                  nop
ext:0804885F                  nop
ext:08048860                  nop
ext:08048861                  nop
ext:08048862                  nop
ext:08048863                  nop
ext:08048864                  nop
ext:08048865                  nop
ext:08048866
ext:08048866 loc_8048866:                              ; CODE XREF: .text:0804884F↑j
ext:08048866                  lea     eax, [ebp-70h]
ext:08048869                  mov     [esp], eax
ext:0804886C                  call    _strlen
ext:08048871                  mov     dword ptr [esp+8], 0
```


然后选择patch program那里的Apply patchs to input file
接下来就很简单了，基本就是常规的分析，输入F1@gA就可以得到flag了
F8C60EB40BF66919A77C4BD88D45DEF4