fd	
bk	
data	
fd	
bk	
data	
fd	
bk	
data	

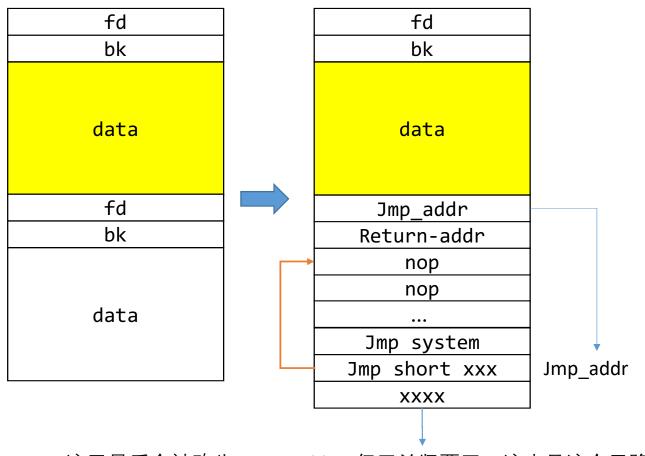
$$A = B->bk$$
 $C = B->fd$
 $Addr_1 = B->bk$
 $Addr_2 = B->fd$
 $A->fd = C$
 $Addr_1->fd = Addr_2$
 $Addr_2->bk = Addr_1$

前提: A的data可以泄露,所以B的fd, bk都可以篡改

关键点不在于发现,而是如何寻找合适的Addr_1 && Addr_2

思路一	
(addr_1, return_addr)(addr_2, system_addr)	
Return_addr System_addr	这个思路常常不行的原因 system函数被这个retrun_addr污染
System_addr Return addr	

思路二



执行过程

- Return_addr 后程序开始在堆上执行,开始执行位置是Jmp_addr
- 首先执行Jmp short xxx, 跳转到nop
- 然后一系列nop,最后跳转到system

这个思路不行的原因 有NX,不可在堆栈上执行指令

这里最后会被改为return-addr,但无关紧要了,这也是这个思路的精髓,利用 jmp直接不去执行污染后的东西

思路三

开始填充

```
<+0>
      : lea
                ecx,[esp+0x4]
                                                        fd
              esp,0xfffffff0
<+4> : and
                                                        bk
                DWORD PTR [ecx-0x4]
<+7> :
         push
                                                                  THIS ADDR
                                                       data
<+195> : call 0x8048504 <unlink>
                                                        fd
<+200> :
         add
                esp,0x10
                                                        bk
<+203> :
                eax,0x0
         mov
                ecx, DWORD PTR [ebp-0x4]
<+208> : mov
<+211> : leave
                                                       data
                esp,[ecx-0x4]
<+212> : lea
<+215> : ret
```

```
<+0>
       : lea
               ecx,[esp+0x4]
                                                         fd
<+4> : and esp,0xfffffff0
                                                         bk
<+7> : push
               DWORD PTR [ecx-0x4]
                                                     System addr
                                                                   THIS ADDR
                                                         XXX
                                                         XXX
<+195> : call 0x8048504 <unlink>
                                                         XXX
. . .
                                                         XXX
                                                   THIS ADDR + 0x4
<+200> :
         add
                esp,0x10
                                                       Ebp - 4
<+203> :
                eax,0x0
         mov
<+208> : mov
                 ecx, DWORD PTR [ebp-0x4]
<+211> : leave
                                                        data
                 esp,[ecx-0x4]
<+212> : lea
<+215> : ret
```

```
<+0>
       : lea
               ecx,[esp+0x4]
                                                         fd
<+4> : and esp,0xfffffff0
                                                         bk
<+7> : push
               DWORD PTR [ecx-0x4]
                                                     System addr
                                                                   THIS ADDR
                                                         XXX
                                                       Ebp - 4
<+195> : call 0x8048504 <unlink>
                                                         XXX
. . .
                                                         XXX
                                                   THIS ADDR + 0x4
<+200> :
         add
                esp,0x10
                                                       Ebp - 4
<+203> :
                eax,0x0
         mov
                 ecx, DWORD PTR [ebp-0x4]
<+208> : mov
<+211> : leave
                                                        data
<+212> : lea
                 esp,[ecx-0x4]
<+215> : ret
```

THIS ADDR + 0x4

Ebp - 4

fd
bk
System_addr
Ebp - 4 - 4
XXX
XXX
XXX
Ebp - 4 - 4
THIS_ADDR + 4
data

THIS_ADDR

两种选择

fd	
bk	
System_addr	THIS_ADDR
XXX	
Ebp - 4	
XXX	
XXX	
THIS ADDR + 0x4	
THIS_ADDR I OXT	
Ebp - 4	

THIS_ADDR + 4 Ebp-4

THIS_ADDR + 0x4 Ebp - 4

```
ecx,[esp+0x4]
<+0>
       : lea
<+4>
       : and
                 esp,0xffffff0
                DWORD PTR [ecx-0x4]
<+7> : push
<+195> : call
               0x8048504 <unlink>
                                            [ebp - 0x4] = THIS\_ADDR + 0x4
. . .
<+200> :
         add
                esp,0x10
<+203> :
                eax,0x0
         mov
                                            ecx = THIS\_ADDR + 0x4
                 ecx, DWORD PTR [ebp-0x4]
<+208> : mov
<+211> : leave
<+212> : lea
                 esp,[ecx-0x4]
                                            esp = [THIS_ADDR] = system
         ret i
<+215> :
                                          eip = esp = system
```