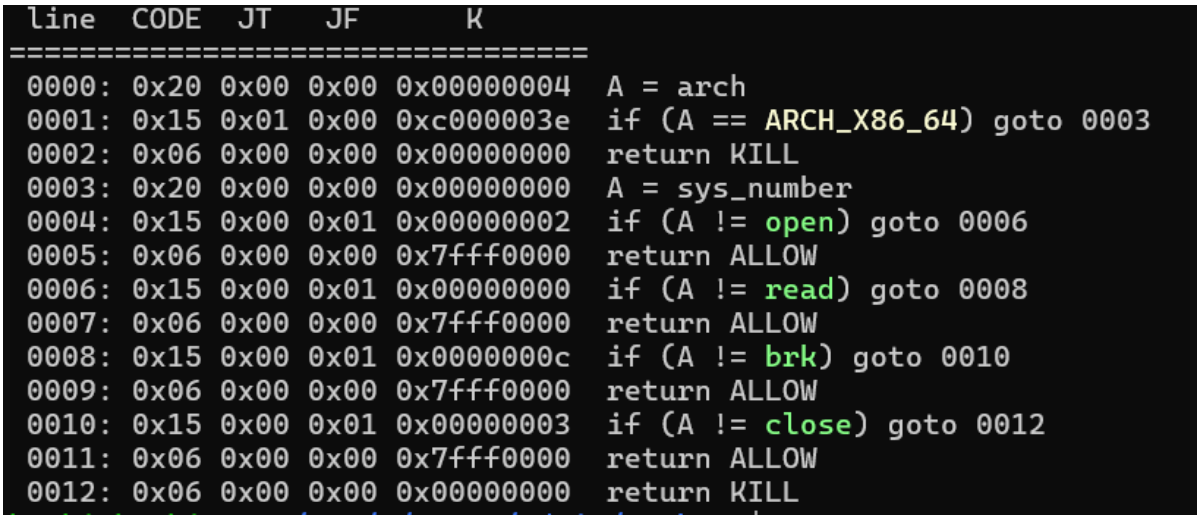
侧信道攻击的一种新方式（无法运行shellcode纯ROP实现侧信道）

例题信息

例题使用的沙箱如下



其使用白名单，几乎只保留了open，read。有or缺w，且无mprotect，并非常规的写shellcode做测信道

本题直接给了所需的一切，甚至直接将flag读入全局变量，只考察该新方式的利用，不添加任何使其复杂的元素

侧信道攻击的新方式

由于rop的自由度远小于shellcode，只能在gadget和函数层面构思

在某个清晨，我灵感迸发，想到了这个侧信道方式：

strcmp函数+ 系统调用

利用方式

通过ROP或者其他方式，在内存中写入自己猜测的flag和真实flag（在猜测过程中保证长度一致）

在ROP链中先调用strcmp函数，两个参数分别为自己猜测的flag和真实的flag。随后比较结束后调整寄存器，跳转到syscall

判断程序在ROP链中是遇到阻塞还是直接退出，若遇到阻塞代表比较通过，直接退出代表比较失败

循环此过程，直到flag完全猜测出

利用原理

strcmp函数的性质：

对两个字符串的字符作差，比较两个字符串。遇到不为0的情况，或者比较结束就会返回，返回值会储存在rax

syscall性质：

通过rax作为系统调用号，rax处于不同值时，执行syscall会运行不同的系统调用

它们的关联点在于rax寄存器，它是储存函数返回值的寄存器，也是储存系统调用号的寄存器。

而rax=0代表字符串相等，同时rax=0代表需要执行系统调用sys_read。

read系统调用时我们的连接会阻塞，而不是直接断开，这可以作为一种侧信道信息，即：

我们可以通过能否再次输入，判断是否运行sys_read，从而推断出字符串是否完全相等

此方式存在的一些问题

能够阻塞程序的系统调用号有不少，比如说有可能触发sleep导致误判为比较通过

本题采用白名单，且白名单以内，只要read存在阻塞，所以不存在干扰

如果遇到干扰，由于flag的字符集有限

```
[hex(i)[2:] for i in range(16) ]  + ["-", '{', '}']
```

我们可以缩小猜测范围，减小误判的风险