Isolation Mech
- The processes in the system have to go via OS (system calls) to communicate with each other.

```c
int sys_write(char *buf)
{
    char *tmp,
    int len;
    mark_ro(buf); //solution
    len = strlen(buf);  // buf is in the user space.  // ISSUE in multithread
    tmp = malloc(len + 1);
    strcpy(tmp,buf);
}
```

- There can be issue when multiple threads are running. They can change the buf, causing overflow.
- We can have locks, but t2(bad guy) will not use them!

Solution:
- make the buffer read only before extracting the length. (doubt! what if context switch happens before this statement.)
- another solution can be to modify the TLB so that the ‘buf’ becomes part of kernel address space. This is when we pass messages between trust domains.
- one more solution is to copy the buffer from user space to kernel space

To be taken care while coding
  1. Time-of-Check-To-Time-of-Use (TOCTTOU bugs)
  2. Pass by Value between trust domains.

Virtual Memory Layout
wftpd 2.6.0
void loguser(char *user)
{
  char buf[512];
  sprintf(buf, user);
}

Stack
if the user = “aaa......a” - 513 chars, the sprint shall overright the buf thus making way to the return address in stack frame of loguser

the bad guy can make the user = “mov ro........etc"

**Defense**

NX bit - mark pages of stack as non-executable

**Counter Attack**

Arc-Injection a.k.a “return-to-libc”

```c
system(char *cmd)
```
user = “sh.........&system.......&buf”