Dirty COW

All questions are 2 points unless otherwise noted.

8.1. The `fork()` system call creates a new process from a parent process. The new process, i.e., the child process, will have a copy of the parent process's memory. Typically, the memory copy is not performed when the child process is created. Instead, it is delayed. Please explain when the memory copy will occur, and why this method is preferable to copying the moment a child is created.

8.2. When a process maps a file into memory using the `MAP_PRIVATE` mode, the memory mapping is depicted in Figure 1. (1) Please describe what is going to happen when this process writes data to address `0x5100`. (2) The Dirty COW race condition occurs inside the `write()` system call. Please explain exactly where the problem is. (3) How can this race condition vulnerability be exploited? (3 points)

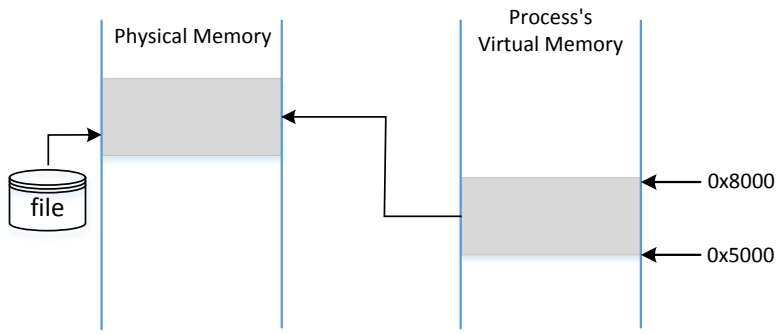

Figure 1: Figure for Problem 8.2.

8.3. The permission of the file `/home/seed/zzz` is readable and writable to the user `seed`. Does the following code (executed by `seed`) modify the content of `/home/seed/zzz`?

```
int f=open("/home/seed/zzz", O_RDWR);
fstat(f, &st);
// Map the entire file to memory
map=mmap(NULL, st.st_size, PROT_READ|PROT_WRITE,
                           MAP_PRIVATE, f, 0);
memcpy(map, "new content", strlen("new content"));
```

8.4. Consider a modified version of the Dirty COW attack, in which we run two processes, instead of two threads. Can we still launch the attack? If so, explain why the attack still succeeds. Else, explain why the attack cannot be run.

8.5. Why cannot we implement copy-on-write in `memcpy()`, so we can use it to write to a private copy of the mapped memory? (3 points)