Operating Systems: Homework 1

Jack Shi - A92122910

April 12, 2018

1. Missing mechanism:

Without privileged instructions Since manipulating protected control registers is a privileged instruction, without it, the user would be able to bypass the dual-mode operation and make their instructions kernel operations as they please. This can result in security issues especially if the user decide to manage I/O directly which is usually a system call through the kernel.

Without memory protection Programs would be able to access eachother's memory address space. This could allow malicious software to steal data or manipulate other programs by illegally access, change their memory data.

Without timer interrupts This would result in the inability to timeout process that take too long or stuck in an infinite loop.

2. System without privileged mode:

Possible to be secure: Programs could be forced to run in VMs such as the JVM where the enviornment is controlled and the program's access is restricted.

Not possible: In case that the program must run on bare-metal for high efficiency or performance reasons, the lack of a privileged mode could result in kernel corruption.

3. Which instructions should be privileged?

a) Set value of timer
privileged: Changing the timer could change the behavior of the interrupts which is a kernel level operation.

b) Read the clock
privileged: System time can be used for timing attacks. The system clock is also managed by the kernel.

c) Clear memory
privileged: Because of memory protection, processes should not be able to clear memory anywhere. However, within it's own allocated address space, it fine.

d) Turn off interrupts
privileged: If any process is allowed to turn off interrupts, then time-out timer interrupt would not be able to prevent malicious infinite loops.

e) Switch from user to monitor(kernel) mode
privileged: There is no point is separating modes if the user can just switch to kernel mode as they please.

4. Condition of failure:

open `pathname` refer to block device, and that block device is being used by the system. `open` fails with error `EBUSY`.

read `file descriptor` does not refer to a opened file. `read` fails with error `EBADF`.

fork `fork` fails with error `ENOMEM` when there is not sufficient memory for another process.

exec If the header of the file for `exec` is not recognized, `exec` will result in error `ENOEXEC`.

unlink If the `pathname` points to somewhere outside of the alloted address sapce, `unlink` result in error `EFAULT`.

5. Challenges when copying parameters from user to kernel modes:
Challenge 1: The program in user mode does not have the privilege to build a function stack in kernel mode and copy over the parameters.
Challenge 2: If the parameters contains pointerrs, dereferencing will require violating the user/kernel mode boundaries as well.
Solution: Temporary grant the caller kernel privileges.

6. Use hardware interval timer to keep track of the time of day:
Solution: Since time of day is usually descretized to seconds, the OS can set 1 second long interval timers that update the system time on 1Hz interrupt.

7. Divise subsitute for traps using interrupts and/or exceptions:
Solution: It is possible if the user get kernel privilege and implement it's own version of the interrupt handler.

8. Consider C program:

- **a**. 8 forks total are created.
- **b**. The statement is executed twice.