Assignment – Theory1_CS19BTECH11026

Name: Naitik Malav
Roll No: CS19TECH11026

1) How does the distinction between kernel mode and user mode function as a rudimentary form of protection (security)?

What is *Kernel Mode*? A supervisor mode commonly known as kernel mode, which has the full and you can say direct access to the system hardware. This means any code execution running at this mode has elevated permissions and access to the core system files. It can enable or disable the hardware and it has access to memory and input/output as well. Also, it can remap memory for them as well as for the other processors.

What is *User Mode*? The user mode basically helps the OS in running user applications. In this type of mode applications, executing codes do not directly access the underlying hardware as well as memory.

How it acts as a rudimentary form of protection (security)?

- By the definition of user mode, code running in the user mode must delegate to system APIs to access system memory and hardware, so it can not directly overwrite the memory or access hardware.
- By default, the manufacturers lock the boot loader which restricts the user to use only user mode and hence it protects their data and the hardware from any malicious code.
- There are some privileged instructions which can be executed only in kernel mode, so basically it restricts user to user mode only.
- This dual mode restricts user or malicious codes to remap system memory directly.
- Suppose a system gets crashes in user mode, then it can be recoverable most of the time, as dual mode prevents to overwrite data in memory.

2) Which of the following instructions should be privileged?
a. Set value of timer
b. Read the clock
c. Clear memory
d. Issue a trap instruction

There are two privileged instructions among the given instructions, which are *set value of timer*, and *clear memory*.

a) If any attempt is made to modify the value of timer in user mode then the hardware does not execute the instruction and treats it as illegal or fatal error and traps it to the OS, and that's why it is called as a privileged instruction.

b) Read the clock is not a privileged instruction because it is executed in user mode and not in a kernel mode.

c) Clear memory is also privileged instruction because it can wipe out all the data of OS, so the user level processes do not allowed to do this.

d) Issue a trap instruction is not privileged because most of the user processes need trap instruction to the run the processes. Issue a trap instruction are usually performed in a user mode.

3) Some early computers protected the operating system by placing it in a memory partition that could not be modified by either the user job or the operating system itself. Describe two difficulties that you think could arise with such a scheme.

The two most popular difficulties with such a scheme are as follows:

a) **Hard to Fix Bugs in OS:** As the OS is placed in memory partition which could not be modified by either the user or OS itself, it simply implies that OS can not be altered. Now, if the OS has some bugs, then it could not be fixed by simply overwriting the memory. In this time, almost there's an update in a week for OS's to remove bugs and for proper functionality, so it could be very difficult, or you can say practically impossible to fix the bug if we are going to follow the old scheme.

b) **User Protection can be compromised**: As we know that many of us stored useful passwords, data in the OS, and also many useful information and passwords are also generated by the OS, so these important data and access control information would have to be passed through or stored in unprotected memory slots and would be accessible to unauthorized users.