HW1Answers

Problem1:

Isolation and protection is achieved in Modern OS using hardware features as follows:

1. Instruction set: There are 2 types of instruction set- Privileged instructions and non-privileged instructions.
   a. Privileged instructions: These instructions are hidden by Operating systems. They are executed by only kernel mode. Initiate input/output, context switching etc is done by privileged instructions.
   b. Non-privileged instructions: These are the normal instructions executed in user as well as kernel mode. Example: ADD, SUB etc are the normal instructions defined in non-privileged instructions.
2. Processor modes: There are 2 modes executed by processor- kernel mode and user mode.
   a. Kernel mode: In this mode, both the privileged and non-privileged instructions are executed.
   b. User mode: Only non-privileged instructions are executed in this mode.
3. Memory Protection: Each process will have their own stack and their own resources that cannot be accessed by any other process.

   Software features:

   It behaves like an interface between user mode and kernel mode. If user mode wants to access the privileged instructions then it makes system call using software and kernel mode perform the privileged instructions for the process in user mode.

   Isolation/Protection is guaranteed using all the hardware and software features as one process will not be able to access the resources of other process and thus, any buggy or malicious apps cannot access the resources of any other application.

Problem 2:

Read() is the system call which will try to access privileged instructions. Therefore, it is about switching from user mode to kernel mode. Typically, a number is associated with each system call, and the system-call interface maintains a table indexed according to these numbers. The system call interface then invokes the intended system call in the operating system kernel and returns the status of the system call and any return values. The caller needs to know nothing about how the system call is implemented or what it does during execution.
Events when a system call is made:

1. The process will make a system call read() and it will request for privileged instructions and parameters in kernel mode.

HW1Answers

2. The processor stores the return address of the function in stack and switches from user mode to kernel mode to access the privileged instructions.
3. In kernel mode, a duplicate process stack is created which has the return address of the function. This is to maintain memory protection.
   a. While privileged instructions are being executed in kernel mode, in meantime, in user mode, another process can access the resources of the current process and change the return address of the function that is currently executing system call in kernel mode.
4. Once the privileged instructions have completed execution then the return address is retrieved from the duplicate stack present in kernel.
5. Process switches from kernel mode to user mode.

Problem 3:

Isolation/protection may also be achieved purely in software without special hardware support using Sandboxing. A sandbox controls the resources that a process may use, by executing the software in a restricted operating system environment. It is one of the examples of virtualization. In sandboxing, it protects the resources in such a way that guest programs can have limited access to the resources. For example, the guest programs run in the sense that it does not function natively on the host.

This helps in Isolation and protection purely in software.

Pros of sandboxing:

1. Gives good isolation as user cannot see anything else on the computer.
2. Malware apps cannot access the resources of other apps.

Cons:

1. Careful network firewalls required, and it causes fair amount of memory overhead.
2. Can be more complex

Pros of hardware based Isolation:

1. Very secure.
2. Better performance

Cons:

1. Not so flexible.
2. More expensive

Problem 4:

Similarities between Multics System and today's desktop Operating system like Linux and windows are:

HW1Answers

1. Both share one computer with many users at a same time.
2. Both modern systems and Multics have time-sharing.
3. Both have single hierarchal system, with subdirectories controlled sharing of files
4. Both have self –hosted development.
5. Both have paged memory that allows for dynamic storage management and has reduced overhead. Segmentation and paging is followed by both the operating systems.
6. Ability of one process to spawn other processes.
7. Isolation and protection from malware and guest programs.

Differences are:

1. Multics has dynamic linking whereas new OS like UNIX has static linking at compile time.
2. In Multics, files look likes a memory but in others, memory is totally a separate resource.
3. It is a slow operating system as compared to others due to dynamic linking.