CSE 303: Quiz #1

Due Friday, September 16th, 2022 at 11:59 PM

The quiz has THREE questions. Please submit your answer on CourseSite as a pdf whose name is precisely your user Id and the "pdf" extension (e.g., abc123.pdf) before the deadline.

**Question 1: "**Storing data permanently in a secondary storage device appears to be a *persistence* problem. However, a deeper view reveals that it is indeed related to all the five OS themes we discussed in lectures". Briefly explain with clear examples to what extent you agree or disagree with this statement.

I agree with this statement. The idea of data being secure as well as persistent is hugely important and consumers rely on these principles even if they are not aware of them. Below are three of the OS themes that I chose to highlight.

**Persistence:** The data that is stored in secondary storage needs to be persistent since when the computer shuts down, the data is expected to still be accessible. From intel.com: files must meet several requirements, as they represent the data that applications use from one run to the next; They must survive after process exit, read by different programs, be able to be backed up and restored, and accessed by several applications simultaneously. The example is clear within our personal computers. We save a request.h file after we complete a function and when we work on the same file the next time, we expect our work to have been saved.

**Concurrency:** From the intel.com article, "[files must be able to be] read by different programs." This feature is largely found within databases, where multiple users can read, write, store, and get files simultaneously. In our first program, the server acts as the database, where the file is not saved to disk after every request, but when the client sends the SAVE request. There are two types of data concurrency within databases: Simultaneous Access to Data and Coexistent Query Workload, from indicative.com. The two types deal with the same issue of concurrency, where SAD is the multiple-user access and CQW is how many users are progressing at the same time.

**Virtualization:** Data visualization allows a single large volume of data to be mapped to a virtual address so multiple computers seem like they have their own independent data. This feature is mostly found within server management within a technique called RAID, or Redundant Array of Independent Disks (Redundant is within the name because of the idea of persistence and Safety). From hpe.com, blocks of data are presented to remote servers as virtual disks, but they look like and act like physical disks to the server. Data virtualization has a host of benefits, including lower costs from purchasing fewer appliances and licenses, saving time through quick upgrades, and reduced risk when one drive fails, its data is mirrored in the virtual array.

**Question 2:** Separating mechanism and policy is widely used in OS designs. Briefly discuss two examples of this separation in two different themes of the five themes we cover in this course (clearly state which theme is related to each example you give).

Please use the remainder of this page to provide your answer to the question. Give a detailed explanation, but keep your entire response to a single side of an 8.5x11 page.

One example of separating mechanism and policy is within the theme of security by protecting the kernel and OS, i.e. who is allowed access and execution (policy) and allowing for dual-mode execution (a mechanism). The policy of only allowing low-level execution of code or low-level changes to the OS by an authorized source is separate from the execution by assigning a user mode and a kernel mode. Because of this one-bit difference, the software will not execute the command. However, a user process can be completed through a system call through interrupts. From geeksforgeeks.org, “when the user application requests for a service from the operating system or an interrupt occurs or system call, then there will be a transition from user to kernel mode to fulfill the requests.

Another example of separating mechanism and policy is within the theme of concurrency through scheduling, i.e. policies like FIFO, shortest, priority level, and the mechanism of timer interrupts. From informit.com, the kernel could have a scheduler with k priority levels, assign each entry with a level and execute from highest priority to lowest. “The system may have different classes of users, each with a different priority” that can increase after I/O or decrease after the array. The separation is between the type of execution and the storing mechanism.

**Question 3:** In P1 all messages are encrypted except req_key and its response, which is claimed to be harmless since the public key sent in the response cannot be used to decrypt further messages. To what extent do you agree with that claim? Are there any other possible attacks a malicious third party can make if it is able to intercept this unencrypted req_key message and/or its response? If yes, give an example of such attacks and briefly explain one way to protect against them. If not, briefly explain why.

Please use the remainder of this page to provide your answer to the question. Give a detailed explanation, but keep your entire response to a single side of an 8.5x11 page.

I agree with this claim because the process of RSA does not rely on the req_key for it to be secure. RSA works on public/private keys, where the private keys are used when decrypting a message and the public key is used for encrypting a message. If Bob wants to send a message to Alice, Bob uses the publicly available public key for encryption, sends that encrypted message to Alice, and since only Alice has the key to decrypt, her private key, Alice is able to decrypt the message (this process works the same way from Alice to Bob). Of course, a malicious third party could intercept the message even before encrypting through an app that records the keyboard, but this type of attack is not a vulnerability of RSA and public/private keys.