CSE221: Homework 1

October 1, 2007

Due: Thursday, October 11th, 2007 at the start of class (11:00am)

Answer the following questions. For questions asking for short answers, support your answers with material from the references papers, and/or with your own critical arguments, as appropriate. I am interested in your justifications as much as the answer itself. There may not necessarily be a “right” answer, although some answers may be easier to justify. Finally, do not use shorthand; write your answers using complete sentences.

1 Layering and concurrency

1.1

In the THE operating system paper, Dijkstra introduced to his system’s architecture both a set of synchronization primitives and a concept of layering. He was motivated in part by fear. Quoting from the paper,

"As captain of the crew I had had extensive experience (dating back to 1958) in making basic software dealing with real-time interrupts and I knew by bitter experience that as a result of the irreproducibility of the interrupt moments a program error could present itself misleadingly like an occasional machine malfunctioning. As a result I was terribly afraid."

What is one more modern term for the kinds of errors which he describes? Give one way in which his design decisions might have mitigated these kinds of errors. Is there any way in which his design decisions could have potentially made the effects of these errors worse? Explain.

1.2

The Nucleus system, like the THE system, is hierarchical, although it uses different synchronization primitives. What benefit do the authors claim their primitives gain them over Dijkstra’s primitives? Briefly discuss the implications on Dijkstra’s claims in his THE paper.

1.3

Later systems, particularly HYDRA, reject the notion of hierarchical layering as a system architecture. Why do they do so? Who is right? Make sure to
2 Rights Amplification

2.1
A rights amplification mechanism is one which enables a more privileged protection domain to perform an operation on behalf of a less privileged protection domain. Ideally, we would like to do this in a controlled fashion such that it preserves protection in the system. For each of the following operating systems, state (a) the protection domains that they support, (b) the mechanism for crossing protection domains, (c) how rights are represented, (d) how rights are amplified crossing domains, and (e) how the OS determines whether to allow the domain crossing.

1. UNIX
2. HYDRA
3. Multics

2.2
Briefly discuss how the functionality of these systems would change if rights amplification were not allowed.

3 Airline Security

3.1
Capabilities and ACLs are two mechanisms for implementing protection policies, not only in computer systems, but also in systems which exist primarily in a social context. For example, to board an airplane in the US, a potential passenger must be in possession of a boarding pass, which an airline issues to its customers on entry to an airport. Then, to enter the protected boarding and departure area of the airport, a passenger must present her boarding pass as well as her state-issued photo identification card to a government employee. Finally, to board her plane, a passenger must present her boarding pass to an airline employee who checks it against the passenger manifest for the flight before letting her board. Additionally, boarding passes are dispensed by airline employees within the secure area as passenger manifests are updated to reflect the changing needs of passengers.

Is such a boarding pass an ACL or a capability? both? neither? Make sure to justify your answer.

3.2
Consider the airline security problem to be, loosely, the problem of controlling who may board commercial passenger airplanes at major airports in the US.
One of the principles the HYDRA authors expound upon is the separation of mechanism and policy. In particular, they say,

"In our view, protection is a mechanism; security is a policy. A system utilizing such a mechanism may be more or less secure depending upon policies governing the use of the mechanism”

What does this mean in terms of HYDRA? How can it be applied to a discussion of airline security?

3.3

The authors of the Multics paper similarly detail five design principles which they consider to be paramount in the design of secure systems. Pick two of these design principles and briefly relate them to the airline security problem.

4 The role of hardware

At least two of the systems we have read about and discussed use specialized hardware to facilitate their implementation. Choose one such instance, describe the hardware that was used, and what advantage it gave the system implementors and designers. Do we still use hardware of this form today?

5 Debugging

Operating systems go to great lengths to provide isolation and protection among processes executing on the system. Process debugging, however, presents a necessary, interesting feature that is made more difficult by process isolation and protection, and requires support from the operating system to function correctly. To the extent possible and where appropriate, when answering the following questions, support your answers with approaches for debugging support found in the papers you have read to this point.

1. Why must a traditional operating system like UNIX explicitly provide support for process debugging?

2. List two distinct operations that a debugger must perform that require support from such an operating system. Are your operations sufficient for process debugging?

3. We have read about at least two other operating systems with architectures which deviate from that of UNIX. Choose one and explain either why it does not need explicit operating system support for process debugging, or if it does, what about its design requires explicit support.

4. Because processes are protected and isolated from each other, operating systems must also provide support for communication and coordination among processes. Why can’t debuggers just use the support that operating systems already provide for process communication and coordination?
5. When working on an operating system, developers also need to use a debugger on the operating system itself. Why is debugging the kernel of an operating system more challenging than debugging a user-level process? What is one option for where to run a kernel debugger?

6. How have the system designers which we’ve studied dealt with the problem of debugging their system implementations? Give one example. What system design principles were used to facilitate this approach?