

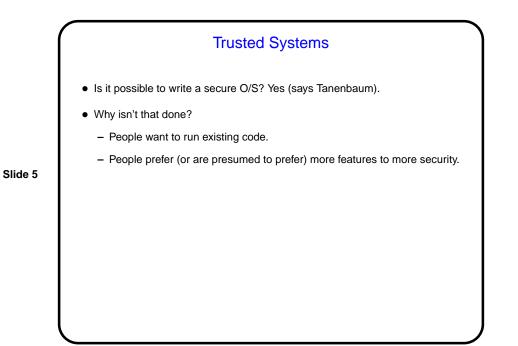


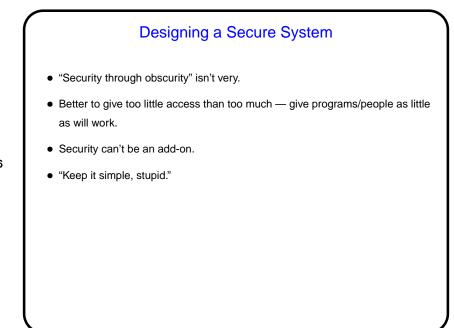


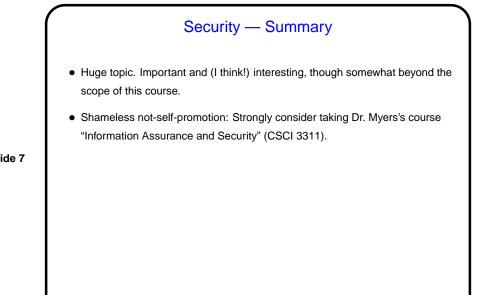
- Textbook discusses several ways "malware" (viruses, worms, etc.) can infect a system.
- Common factor (my opinion!) is allowing execution of code that does something unwanted. (Either users don't realize this is happening, or they don't realize the implications?) Social engineering is often involved. Monoculture makes the malware writer's job easier.

Slide 4

## Safe Execution of "Mobile" Code Is there a way to safely execute code from possibly untrustworthy source? Maybe — approaches include sandboxing, interpretation, code signing. Example — Java's designed-in security: At source level, very type-safe — no way to use void\* pointers to access random memory. (Contrast with C!) When classes are loaded, "verifier" checks for potential security problems (not generated by normal compilers, but could be done by hand). At runtime, security manager controls what library routines are called — e.g., applets by default can't do file operations, many kinds of network access.







Slide 8

## Minute Essay • Over the course of the semester I've told several "war stories" - tales of woe that taught me (or someone) something. Do you have a favorite war story to tell?