Slide 1

Administrivia

- Homeworks and exams graded; averages and approximate letter grades mailed this morning.
- Extra-credit problems will be on Web later today. Will be due next Tuesday. Can only help your grade.

Slide 2

Security — Overview

- Goals:
  - Data confidentiality — prevent exposure of data.
  - Data integrity — prevent tampering.
  - System availability — prevent DOS.
- What can go wrong:
  - Deliberate intrusion — from casual snooping to “serious” intrusion.
  - Accidental data loss — “acts of God”, hardware or software error, human error.

Slide 3

User Authentication

- Based on "something the user knows" — e.g., passwords. Problems include where to store them, whether they can be guessed, whether they can be intercepted.
- Based on "something the user has" — e.g., key or smart card. Problems include loss/theft, forgery.
- Based on "something the user is" – biometrics. Problems include inaccuracy/spoofing.

Slide 4

Attacks From Within

- Trojan horses (and how this relates to `$PATH`).
- Login spoofing.
- Logic bombs and trap doors.
- Buffer overflows (and how this relates to, e.g, `gets`).
- And many more . . .

Slide 5

Designing a Secure System

- "Security through obscurity" isn't very.
- Better to give too little access than too much — give programs/people as little as will work.
- Security can't be an add-on.
- "Keep it simple, stupid."

Slide 6

Attacks From Outside

- Can categorize as viruses (programs that reproduce themselves when run) and worms (self-replicating) — similar ideas, though.
- Many, many ways such code can get invoked — when legit programs are run, at boot time, when file is opened by some applications ("macro viruses"), etc.
- Also many ways it can spread — once upon a time floppies were vector of choice, now networks or e-mail. Common factors:
  - Executable content from untrustworthy source.
  - Human factors.

  "Monoculture" makes it easier!
- Virus scanners can check all executables for known viruses (exact or fuzzy matches), but hard/impossible to do this perfectly.
- Better to try to avoid viruses — some nice advice on p. 633.

Slide 7

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.
  - 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

Trusted Systems

- Is it possible to write a secure O/S? Yes (says Tanenbaum).
- Why isn't that done?
  - People want to run existing code.
  - People prefer (or are presumed to prefer) more features to more security.

Slide 9

Course Recap

- Four key areas (the gospel according to Pitts):
  - Process management.
  - Memory management.
  - I/O management.
  - Filesystem management.
- Also a little about history, a little about security.

Slide 10

Recap, Continued

- Some recurring themes:
  - Interaction between h/w and s/w — some h/w features are there to support o/s features; o/s influenced by what's available in h/w.
  - Trade-offs — often the answer to "which is best?" is "it depends".
- We didn't cover the whole book, but if you look at the ACM's guidelines for an undergrad o/s course — we pretty much did what they said.

Process Management

- O/S as virtual machine — process abstraction, “concurrent” execution, IPC, distributed algorithms.
- O/S as resource manager — implementation of above, including interrupts and context switches, CPU scheduling.

Slide 11

Memory Management

- O/S as virtual machine — memory protection, virtual memory, “multiprogramming”.
- O/S as resource manager — implementation of above, including page replacement algorithms.

Slide 12

Slide 13

I/O Management

- O/S as virtual machine — layered abstractions for working with I/O devices (user-level s/w, device-independent s/w).
- O/S as resource manager — implementation of above, plus a little about lower-level interaction with devices (programmed versus interrupt-driven I/O versus DMA).

Slide 14

Filesystem Management

- O/S as virtual machine — filesystem abstractions (files, file attributes, directory structures).
- O/S as resource manager — implementation of above, disk-space management, reliability and consistency.

Slide 15

Minute Essay

- None — sign in.
- (Good luck with your finals, and have a good break!)