Administrivia

- Reminder: Homework 5 due Monday.

- Homework 6 on Web, due Wednesday after holiday.

- Class next Wednesday: How many will be here?

Slide 1

Minute Essay From Last Lecture

- (Question was why so many different file systems.)

  Different hardware/software configurations.

  Different users/applications have different needs.

  Different filesystems have different strengths/weaknesses.

  Different ideas about how to store files.

  New hardware.

  New ideas for software.

  Someone gets bored / wants to be different / is stubborn!

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).

Slide 5

- And many more ...

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.

Slide 6

- "Keep it simple, stupid."

Attacks From Outside

Slide 7

- 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.

Safe Execution of "Mobile" Code

Slide 8

- 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.

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

Some Places to Learn More

- Special-topics course next semester (CSCI 3394, Information Security).

- `comp.risks` newsgroup / mailing list:
  `http://catless.ncl.ac.uk/Risks`.

Slide 10

Slide 11

Minute Essay

- How to spend the remaining class periods? Ideas I've had — say which sound interesting, or propose alternatives:
    - What happens during startup / boot.
    - Device-driver demo (if feasible).
    - Linux internals.
    - More about security (maybe "stack smashing for fun and profit" paper).
    - Virtualization.