Lecture 8 Outline

Reading: *text*, §23.2–4; [4]

Assignments due: Homework #1, due April 13, 2011 at 11:55pm
Homework #2, due April 27, 2011 at 11:55pm

1. Flaw Hypothesis Methodology
   a. System analysis
   b. Hypothesis generation
   c. Hypothesis testing
   d. Generalization
2. Hypothesis testing
   a. Look at system code, see if it would work (live experiment may be unneeded)
   b. If live experiment needed, observe usual protocols
3. Generalization
   a. See if other programs, interfaces, or subjects/objects suffer from the same problem
   b. See if this suggests a more generic type of flaw
4. Elimination
5. Where to start
   a. Unknown system
   b. Known system, no authorized access
   c. Known system, authorized access
6. Examples
   a. Burroughs system
   b. Corporate site
7. Vulnerability models
   a. PA model
   b. RISOS
   c. NRL
   d. Aslam
8. Example Flaws
   a. *fingerd* buffer overflow
   b. *xterm* race condition
9. RISOS
   a. Goal: Aid managers, others in understanding security issues in OSes, and work required to make them more secure
   b. Incomplete parameter validation—failing to check that a parameter used as an array index is in the range of the array;
   c. Inconsistent parameter validation—if a routine allowing shared access to files accepts blanks in a file name, but no other file manipulation routine (such as a routine to revoke shared access) will accept them;
   d. Implicit sharing of privileged/confidential data—sending information by modulating the load average of the system;
   e. Asynchronous validation/Inadequate serialization—checking a file for access permission and opening it non-atomically, thereby allowing another process to change the binding of the name to the data between the check and the open;
   f. Inadequate identification/authentication/authorization—running a system program identified only by name, and having a different program with the same name executed;
   g. Violable prohibition/limit—being able to manipulate data outside one's protection domain; and
   h. Exploitable logic error—preventing a program from opening a critical file, causing the program to execute an error routine that gives the user unauthorized rights.