Outline for April 8, 2004

Reading: Chapter 23.3–23.4

Outline for the Day

1. Vulnerability Models
   a. PA model
   b. RISOS
   c. NRL
   d. Aslam
2. Example Flaws
   a. *fingerd* buffer overflow
   b. *xterm* race condition
3. 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.
4. PA Model (Neumann's organization)
   a. Goal: develop techniques to search for vulnerabilites that less experienced people could use
   b. Improper protection (initialization and enforcement)
      i. improper choice of initial protection domain - "incorrect initial assignment of security or integrity level at system initialization or generation; a security critical function manipulating critical data directly accessible to the user";
      ii. improper isolation of implementation detail - allowing users to bypass operating system controls and write to absolute input/output addresses; direct manipulation of a "hidden" data structure such as a directory file being written to as if it were a regular file; drawing inferences from paging activity
      iii. improper change - the "time-of-check to time-of-use" flaw; changing a parameter unexpectedly;
      iv. improper naming - allowing two different objects to have the same name, resulting in confusion over which is referenced;
      v. improper deallocation or deletion - leaving old data in memory deallocated by one process and reallocated to another process, enabling the second process to access the information used by the first; failing to end a session properly
   c. Improper validation - not checking critical conditions and parameters, so a process addresses memory not in its memory space by referencing through an out-of-bounds pointer value; allowing type clashes; overflows
   d. Improper synchronization
      i. improper indivisibility - interrupting atomic operations (*e.g.* locking); cache inconsistency

   ii. improper sequencing - allowing actions in an incorrect order (*e.g.* reading during writing)
e. Improper choice of operand or operation - using unfair scheduling algorithms that block certain processes or users from running; using the wrong function or wrong arguments.
f. Analysis procedure
   i. Collect descriptions of protection patterns
   ii. Convert to raw error patterns
   iii. Abstract into system-independent components
   iv. Determine which features in the OS code are relevant, and abstract relevant contexts of those features
   v. Compare the combinations of the relevant features in the OS with generic error patterns

5. NRL
   a. Goal: Find out how vulnerabilities enter the system, when they enter the system, and where they are
   b. Axis 1: inadvertent (RISOS classes) vs. intentional (malicious/nonmalicious)
   c. Axis 2: time of introduction (development, maintenance, operation)
   d. Axis 3: location (hardware, software: OS, support utilities, applications)
6. Aslam
   a. Goal: Treat vulnerabilities as faults
   b. Coding faults: introduced during software development
      i. Synchronization errors
      ii. Validation errors
   c. Emergent faults: introduced by incorrect initialization, use, or application
      i. Configuration errors
      ii. Environment faults
   d. Introduced decision procedure to classify vulnerabilities in exactly one category