

Outline for January 14, 2002

1. Greetings and Felicitations!
 - a. Homework turn-in directory had a problem; if submitted before 8PM on Sunday, please *resubmit*
2. Puzzle of the day
3. Common Implementation Vulnerabilities
 - a. Unknown interaction with other system components (DNS entry with bad names, assuming finger port is finger and not chargen)
 - b. Overflow (year 2000, *lpr* overwriting flaw, *sendmail* large integer flaw, *su* buffer overflow)
 - c. Race conditions (*xterm* flaw, *ps* flaw)
 - d. Environment variables (*vi* one-upsmanship, *loadmodule*)
 - e. Not resetting privileges (Purdue Games incident)
4. Vulnerability Models
 - a. PA model
 - b. RISOS
 - c. NSA
5. PA Model (Neumann's organization)
 - a. 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
 - b. Improper validation - not checking critical conditions and parameters, leading to a process' addressing memory not in its memory space by referencing through an out-of-bounds pointer value; allowing type clashes; overflows
 - c. 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)
 - d. 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.
6. RISOS
 - a. Incomplete parameter validation - failing to check that a parameter used as an array index is in the range of the array;
 - b. 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;
 - c. Implicit sharing of privileged/confidential data - sending information by modulating the load average of the system;

- d. 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;
 - e. Inadequate identification/authentication/authorization - running a system program identified only by name, and having a different program with the same name executed;
 - f. Violable prohibition/limit - being able to manipulate data outside one's protection domain; and
 - g. 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.
7. Penetration Studies
 - a. Why? Why not analysis?
 - b. Effectiveness
 - c. Interpretation
 8. Flaw Hypothesis Methodology
 - a. System analysis
 - b. Hypothesis generation
 - c. Hypothesis testing
 - d. Generalization
 9. System Analysis
 - a. Learn everything you can about the system
 - b. Learn everything you can about operational procedures
 - c. Compare to models like PA, RISOS
 10. Hypothesis Generation
 - a. Study the system, look for inconsistencies in interfaces
 - b. Compare to previous systems
 - c. Compare to models like PA, RISOS
 11. 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
 12. 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
 13. Peeling the Onion
 - a. You know very little (not even phone numbers or IP addresses)
 - b. You know the phone number/IP address of system, but nothing else
 - c. You have an unprivileged (guest) account on the system.
 - d. You have an account with limited privileges.