Notes for October 10, 2000

1. Greetings and Felicitations!
   a. Handin program will be working tonight; currently directory not set up
   b. Show how to do *gdb* when stack smashed (by putting breakpoints around function call)
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.

Puzzle of the Day

"Open source" is a movement to make source code available. The Open Source Definition gives one meaning of the term "open source." Do you think that adopting this definition, and making programs available as "open source," will improve the security of computer software and systems?

The Open Source Definition

(Version 1.7)[1]

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution
   The license may not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license may not require a royalty or other fee for such sale.
2. Source Code
   The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost -- preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.
3. Derived Works
   The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.
4. Integrity of The Author's Source Code.
   The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.
5. No Discrimination Against Persons or Groups.
   The license must not discriminate against any person or group of persons.
6. No Discrimination Against Fields of Endeavor.
   The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.
7. Distribution of License.
   The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.
8. License Must Not Be Specific to a Product.
   The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.
9. License Must Not Contaminate Other Software.
   The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

---

1. From the web page *http://www.opensource.org/osd.html*