Outline for January 17, 2003

Reading: Text, §23.3–23.4, Robust Programming handout

Discussion Problem

The UNIX system reserves network ports numbered 1023 and below for *root*-owned processes only. User processes must use ports with higher numbers. So, if the source port from a remote host has a source port of 536, it must have originated with a process that was at one time *root*. This is a UNIX standard, **not** an Internet one.

What problems can this scheme cause in a heterogeneous network?

Outline for the Day

1. Vulnerability Models
    a. PA model
    b. RISOS
    c. NRL
    d. Aslam
2. PA Model (Neumann's organization)
    e. 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
3. 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)
4. 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
5. 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)
6. Robust Programming
    a. Principles
    b. Creating, reading tickets
    c. Creating, deleting queues
    d. Adding, removing items