Outline for May 30, 2000

1. Greetings and felicitations!
2. Vulnerabilities Models
   a. RISOS (1975), to let managers, etc. know about integrity problems
   b. PA (1976-78), automated checking of programs
   c. NSA, contents unknown but similar to PA and RISOS
   d. Aslam, fault-based; for C programs
   e. Landwehr, classify according to attack purpose as well as type; based on RISOS
   f. Bishop, still being developed
3. RISOS (Research Into Secure Operating Systems); Abbott *et al.*
   a. Improper parameter validation
   b. Inconsistent parameter validation
   c. Implicit sharing of privileged data
   d. Asynchronous validation/incorrect serialization (*eg*., TOCTTOU)
   e. Inadequate identification/authorization/authentication
   f. Violable prohibition/limit
   g. Exploitable logic error
4. PA (Protection Analysis); Bisbey *et al*.
   a. Improper protection domain; 5 subclasses
      - Improper initial protection domain
      - Improper isolation of implementation details
      - Improper change, (TOCTTOU flaws)
      - Improper naming
      - Improper deletion/deallocation
   b. Improper validation
   c. Improper synchronization; 2 subclasses
      - Improper divisibility
      - Improper sequencing
   d. Improper choice of operand and operation
5. Note: PA classes map into RISOS classes and vice versa
6. Flaw Hypothesis Methodology
   a. Information gathering -- emphasize use of sources such as manuals, protocol specs, design documentation, social engineering, source code, knowledge of other systems, *etc.*
   b. Flaw hypothesis -- old rule of “if forbidden, try it; if required, don’t do it”; knowledge of other systems’ flaws, analysis of interfaces particularly fruitful, go for assumptions and trusts
   c. Flaw testing -- see if hypothesized flaw holds; preferable *not* to try it out, but look at system closely enough to see if it will work, design attack and be able to show why it works; but sometimes actual test necessary -- do not use live production system and be sure it’s backed up!
   d. Flaw generalization -- given flaw, look at causes and try to generalize. Example: UNIX environment variables.
   e. (sometimes) Flaw elimination -- fix it; may require redesign so the penetrators may not do it
7. Example penetrations
   a. MTS
   b. Burroughs
8. Principles of Secure Design
   a. Refer to both designing secure systems and securing existing systems
   b. Speaks to limiting damage
9. Principle of Least Privilege
   a. Give process only those privileges it needs
   b. Discuss use of roles; examples of systems which violate this (vanilla UNIX) and which maintain this

(Secure Xenix)
- c. Examples in programming (making things setuid to root unnecessarily, limiting protection domain; modularity, robust programming)
- d. Example attacks (misuse of privileges, etc.)

10. Principle of Fail-Safe Defaults
    - a. Default is to deny
    - b. Example of violation: *su* program
11. Principle of Economy of Mechanism
    - a. KISS principle
    - b. Enables quick, easy verification
    - c. Example of complexity: *sendmail*
12. Principle of Complete Mediation
    - a. All accesses must be checked
    - b. Forces system-wide view of controls
    - c. Sources of requests must be identified correatly
    - d. Source of problems: caching (because it may not reflect the state of the system correctly); examples are race conditions, DNS poisoning
13. Principle of Open Design
    - a. Designs are open so everyone can examine them and know the limits of the security provided
    - b. Does ***not*** apply to cryptographic keys
    - c. Acceptance of reality: they can get this info anyway
14. Principle of Separation of Privilege
    - a. Require multiple conditions to be satisfied before granting permission/access/*etc*.
    - b. Advantage: 2 accidents/errors/*etc*. must happen together to trigger failure
15. Principle of Least Common Mechanism
    - a. Minimize sharing
    - b. New service: in kernel or as a library routine? Latter is better, as each user gets their own copy
16. Principle of Psychological Acceptability
    - a. Willingness to use the mechanisms
    - b. Understanding model
    - c. Matching user’s goal