Outline for May 18, 2000

1. Greetings and felicitations!
   a. A bit about the penetration stuff for homework 3
2. Verifiably Secure Systems
   a. Review notion of reference monitor
   b. Review notion of "trusted path"
   c. Isolate all control functions into a small nucleus called a "security kernel"
   d. Review Levels of Abstraction
3. UCLA Secure UNIX
   a. Each user process in separate domain, with 2 part
   b. Application program runs in user mode
   c. UNIX interface and Kernel Interface SubSystem run in supervisor mode
   d. Protection domain represented by a C-List
   e. Policy manager establishes policies for kernel objects, shared files
   f. Dialoguer establishes trusted path between user, kernel
4. Verification
   a. Top level specification
   b. Abstract level specification
   c. Low level specification
   d. Code satisfying specifications: formulate specs in terms of abstract machines with states and transitions such that protected objects may be modified or read *only* by explicit request; and all accesses must be authorized
   e. Verify code implementations satisfies low level specs, all levels consistent
5. KSOS
   a. Kernel is an operating system, not a security kernel
   b. Enforces access control policy, including multi-level security
   c. Handles files, type extensions à la DOS and TOPS-20
   d. UNIX emulator, "trusted" non-lernel system software run in supervisor mode
6. PSOS
   a. Capabilities at lowest level used for addressing
   b. 15 layers; all below 8 invisible at user interface, except level 4 (basic operations)
   c. PSOS Hierarchy
      16. Command interpreter
      15. User environments and name space
      14. User input/output
      13. Procedure records
      12. User processes and visible input/output
      11. Creation, deletion of user objects
      10. Directories
      9. Abstract data types
      8. Virtual memory (segmentation)
      7. Paging
      6. System processes and system input/output
      5. Primitive input/output
      4. Basic arithmetic and logical operations
      3. Clocks
      2. Interrupts
      1. Real memory
      0. Capabilities

7. Verification
   a. Hierarchical Decomposition Methodology breaks system into hierarchy or abstract machines
   b. Specify each module in SPECIAL
   c. Functions: primitive V-functions give value of state variable
      derived V-functions give values computed from state values
      O-functions cause state transitions
      OV-functions do both
8. Methodology
   a. Interface definition; decomposed into set of modules each of which manages some system object (collection of V, O functions); general security requirements formulated (Detection Principle, Alteration Principle)
   b. Hierarchical Decomposition: modules arranged in linear hierarchy; consistency of structure, function names verified
   c. Module Specification: develop formal specs for each module; verify internal consistency, global assertions including representation of general security requirements (which are the PSOS principles expressed in terms of read/write capabilities)
   d. Mapping functions; define these to describe the state space at one level in terms of lower level and verify consistency of mapping functions with respect to specifications, modular decomposition
   e. Implementation: implement, verify modules as you go
9. VAX VMM Security Kernel
   a. VM monitor for the VAX; can run VMS or Ultrix, but is itself a security kernel
   b. Design: present VAX architecture
   c. Compress rings: Real: user, supervisor, executive, kernel; VM user, supervisor, VM executive, VM kernel, forbidden
   d. Subjects: users, VMs; servers run in VM kernel, and only run kernel software; can't run user code
   e. Objects: flat file system for kernel, each VM has its own file system
   f. Access classes: security, integrity levels form access class; A = B iff security, integrity levels and classes the same; A > B iff A > B for *both* integrity and security (> dominates)
   g. Layered design:
      16. Users
      15. VMOS (virtual machine OS)
      14. Secure Server layer (trusted path for security kernel)
      13. Virtual VAX layer (emulates sensitive instructions, interrupts, exceptions, etc.)
      12. Kernel interface layer (virtual controllers for the virtual I/O devices)
      11. Virtual printers layer (implements virtual printers for each VM)
      10. Virtual terminals layer
      9. Volumes layer (VAX Security kernel file columes; registries of all subkects, objects)
      8. Files-11 Files layer implements subset of a file system used in the VMS operating system; all files must be preallocated and contiguous)
      7. Auditing layer
      6. High-level scheduler
      5. VM Virtual memory layer (shadow page tables, etc.)
      4. VM Physical layer (manages physical memory)
      3. I/O Services layer (implements real I/O)
      2. Lower Level Scheduler
      1. Hardware Interrupt Handler layer (interrupt handlers for the physical I/O controllers)