Outline for May 30, 1997

1. Greetings and Felicitations
   a. UCD students: everything except final report due next Friday, and that is due 2 weeks from Friday
   b. NTU students: homeworks 1, 2, and 3 no later than June 6, please!
2. Verification technique: Hierarchical Design Methodology (HDM)
   a. Break system into hierarchy of abstract machines
   b. Specify each as (set of) modules using SPECIAL: primitive V-functions give value of state variable, and derived V-functions give value computed from value of state variable; O-functions cause state transitions; and OV-functions do both
   c. Stage #1: Interface definitions:
      • decomposed into a set of modules each of which manages some object (collection of V-, O-, OV- functions)
      • general security requirements formulated: for PSOS, Detection Principle (no unauthorized acquisition of information) and Alteration Principle (no unauthorized alteration of information)
   d. Stage #2: Hierarchical Decomposition
      • Modules arranged in linear hierarchy
      • Consistency of structure, function names verified
   e. Stage #3: Module Specification
      • develop formal specifications for each model

      verify internal consistence and global assertions (including representation of basic security requirements); for PSOS, the above 2 principles are expressed in terms of read/write capabilities
   f. Stage #4: Mapping Functions
      • define these to describe the state space at one level in terms of the state space of the level beneath it (i.e., i-th level V-functions related to (i-1)-st level V-functions by expressions
      • verify consistency of mapping functions with specifications and decompositions
   g. Stage #5: Implementation
      • implement and verify modules, from lowest to highest; verify as you go
3. Verification technique: UCLA Secure Kernel Method
   a. Top-level specifications
   b. Abstract level specifications
   c. Low level specifications
   d. Code satisfying specifications; formulate the specifications in terms of abstract machine with states and transitions satisfying:
      • protected objects may be modified or read only by explicit requests
      • all accesses must be authorized
   e. Then verify code implementation satisfies the low-level specifications, and all levels are consistent
4. VAX VMM Security Kernel Goals
   a. Meet A1 requirements

   b. Run on commercial hardware without special modifications (other than microcode changes for virtualization)
   c. Provide software compatibility for VMS and ULTRIX-32 applications
   d. Provide acceptable level of performance
5. Kernel Overview
   a. Several virtual machines on one VMM Security Kernel
   b. Architecture, microcode had to be extended to support this; runs only on some VAXen (8530, 8550, 8800, 8810)
   c. VMs assigned access class (secrecy class + integrity class) based on BLP and Biba; supports ACLs on objects too
6. Design Approach
   a. Decision to use VMM to enable ULTRIX, VMS to run with no changes, rather than redesign either or both
   b. Ring compression technique: needed five rings (VM: user, supervisor, VM executive, VM kernel, real kernel) but had four(real machine: user, supervisor, executive, kernel). Added a bit to the PSL and made executive, kernel both be in real executive ring
   c. Subjects: users, VMs. User uses trusted path to reach a server (part of the kernel itself) to connect to a VM. If authorized, server session suspended, VM takes over communication line
   d. Objects: real devices, access to which is controlled by the TCB; disk and tape volumes, under control of VM (“exchangeable volumes”) or VAX Security Kernel (“VAX security kernel volume”), which is a flat file system used by the security kernel or to create virtual disk volumes.
   e. Access Classes: equal => seclvl(A) = seclvl(B), seccat(A) = seccat(B), intlvl(A) = intlvl(B), intcat(A) = intcat(B); A dominates B iff: seclvl(A) ≥ seclvl(B), seccat(A) contains seccat(B), intlvl(A) ≤ intlvl(B), intcat(A) is contained in intcat(B); A can read B only if A dominates B, and A can write to B only if B dominates A.
   f. Privileges: user (really, operator, sys managers, and security managers) privileges (turn auditing on/off, changing passwords, assigning access classes, mount volumes, etc.) and VM (dismount volumes, activate and deactivate other VMs; change ACLs if class permits). Done through trusted path.
   g. Layers:

      low VAX hardware
      modified microcode for virtualization
      hardware interrupt handlers
      lower-level scheduler
      I/O services: device drivers controlling real I/O devices
      VM-physical space manager: manages physical memory, assigns it to VMs
      VM-virtual space manager: shadow page tabvles used by VM page managers
      higher-level scheduler
      audit trail
      files-11 Files: subset of ODS-2 file system, used by VMS; all files must be preallocated and contiguous
      volumes: registries of objects, implements volumes

virtual terminals: physical terminal lines, virtual terminals
virtual printers: VM printers, labelling of output
kernel interface: virtual I/O & security function controllers (loading virtual disks onto virtual drives)
secure server/virtual VAX: implements trusted path/emulates sensitive instructions
---------security perimeter---------
virtual machine OS: virtual machine's OS
users