Outline for June 2, 1997

1. Greetings and Felicitations
2. VAX VMM Security Kernel Design Approach
   a. 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 tables 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
   b. Programming Language: want strong typing, but also ability to compile very large programs correctly, produce high-quality VAX code, and be supported; limited choices to 3 langs:
      (i) BLISS-32; not strongly typed
      (ii) PASCAL; high-quality code generation not yet available
      (iii) C; good code generator, but not much experience with it
      (iv) PL/I; same code generator as C, better typing support, more experience
      When PASCAL compiler became available, switched to it. Also used MACRO-32, the assembler (out of 49,000 lines, 11,500 were in assembler, 29,500 in PASCAL, and 8,000 in PL/I)
   c. Coding Strategies: avoid use of global pools (like sharing input buffers) to minimize covert channels; different sections of kernel memory separated by no-access pages, to force failures on buffer overflows; unused memory initialized to 1's, not0's, to increase chances of faulting; used special mamagement systems to enforce layering
3. Human Interfaces
   a. Secure Server: commands implemented in trusted code; SECURE commands (administrative commands) are parsed in VM.

   b. BREAK key = SAK; controls terminal connections
   c. SECURE commands: VM secure executed in context of issuing VM; User secure executed by Secure Server. Latter provides trust & accountability due to trusted path and SS displaying command from within the Secure Server
4. Network Security
   a. Quick review of ISO/OSI model
   b. Link v. end-to-end encryption
5. Network Security Threats (review)
   a. snooping
   b. modification
   c. masquerading
   d. replay
   e. delay
   f. denial of service
   g. repudiation of origin
   h. denial of receipt