Notes for November 21, 2000

1. Greetings and Felicitations!
   a. Turn back midterm
   b. Go over project
2. Puzzle of the day
3. Identity
   a. Principal and identity
   b. Users, groups, roles
   c. Identity on the web
   d. Host identity: static and dynamic identifiers
   e. State and cookies
   f. Anonymous remailers: type 1 and type 2 (mixmaster)
4. Privilege in Languages
   a. Nesting program units
   b. Temporary upgrading of privileges
5. Access Control Lists
   a. UNIX method
   b. ACLs: describe, revocation issue
6. MULTICS ring mechanism
   a. MULTICS rings: used for both data and procedures; rights are REWA
   b. (b_1, b_2) access bracket - can access freely; (b_3, b_4) call bracket - can call segment through gate; so if *a*'s access bracket is (32,35) and its call bracket is (36,39), then *assuming permission mode (REWA) allows access*, a procedure in:
      rings 0-31: can access *a*, but ring-crossing fault occurs
      rings 32-35: can access *a*, no ring-crossing fault
      rings 36-39: can access *a*, provided a valid gate is used as an entry point
      rings 40-63: cannot access *a*
   c. If the procedure is accessing a data segment *d*, no call bracket allowed; given the above, *assuming permission mode (REWA) allows access*, a procedure in:
      rings 0-32: can access *d*
      rings 33-35: can access *d*, but cannot write to it (W or A)
      rings 36-63: cannot access *d*
7. Capabilities
   a. Capability-based addressing: show picture of accessing object
   b. Show process limiting access by not inheriting all parent's capabilities
   c. Revocation: use of a global descriptor table
8. Lock and Key
   a. Associate with each object a lock; associate with each process that has access to object a key (it's a cross between ACLs and C-Lists)
   b. Example: use crypto (Gifford). *X* object enciphered with key *K*. Associate an opener *R* with *X*. Then:
      OR-Access: *K* can be recovered with any D_i in a list of *n* deciphering transformations, so
      $R = (E_1(K), E_2(K), ..., E_n(K))$ and any process with access to any of the D_i's can access the file
      AND-Access: need all *n* deciphering functions to get *K*: $R = E_1(E_2(...E_n(K)...))$

Puzzle of the Day

Your UNIX system has been attacked. The *uucp* entry in your /etc/passwd file has a UID of 0. You have run *ps* to see if any unusual processes were executing. None were. You ran *ls* to find any unusual files or directories. None were reported. You ran *du* to determine if the size of any file system was unusually large (indicating hidden files). Nope.

You suspect that someone has, somehow, hidden files (or directories) and an executing process. You decide to start at the /dev directory, to see if they created any new device files. Again, an *ls* lists only those files you expect to see. But you are still suspicious, and want to confirm the results.

1. What would you do?
2. You still suspect that the attacker left an illicit process executing. But *ps* showed nothing. How would you confirm or refute your suspicion?