Outline for March 13, 2002

Reading: §15.2, 15.4, §18

1. Greetings and Felicitations
2. Puzzle of the day
3. 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
4. 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: $R = E_1(E_2(...E_n(K)...))$
   c. Types and locks
5. Privilege in Languages
   a. Nesting program units
   b. Temporary upgrading of privileges; amplification
6. Malicious logic
   a. Quickly review Trojan horses, viruses, bacteria; include animal and Thompson's compiler trick
   b. Logic Bombs, Worms (Schoch and Hupp)
7. Ideal: program to detect malicious logic
   a. Can be shown: not possible to be precise in most general case
   b. Can detect all such programs if willing to accept false positives
   c. Can constrain case enough to locate specific malicious logic
   d. Can use: writing, structural detection (patterns in code), common code analyzers, coding style analyzers, instruction analysis (duplicating OS), dynamic analysis (run it in controlled environment and watch)
8. Best approach: data, instruction typing
   a. On creation, it's type "data"
   b. Trusted certifier must move it to type "executable"
   c. Duff's idea: executable bit is "certified as executable" and must be set by trusted user
9. Practise: Trust
   a. Untrusted software: what is it, example (USENET)
   b. Check source, programs (what to look for); C examples
   c. Limit who has access to what; least privilege
   d. Your environment (how do you know what you're executing); UNIX examples
10. Practise: detecting writing
    a. Integrity check files a la binaudit, tripwire; go through signature block
    b. LOCUS approach: encipher program, decipher as you execute.
    c. Co-processors: checksum each sequence of instructions, compute checksum as you go; on difference, complain