Notes for December 5, 2000

1. Greetings and Felicitations!
   a. Final study guide
2. Puzzle of the day
3. Lattice Model
   a. Set of classes *SC* is a partially ordered set under relation $\leq$ with GLB ($\otimes$), LUB ($\oplus$)
   b. Note: $\leq$ is reflexive, transitive, antisymmetric
   c. Application to MLS: forms a lattice with elements being the Cartesian product of the linear lattice of levels and the subset lattics of categories
   d. Examples: $(A, C) \leq (A', C')$ iff $A \leq A'$ and $C \subseteq C'$;
      $(A, C) \oplus (A', C'') = (\max(A, A'), C \cup C')$
      $(A, C) \otimes (A', C') = (\min(A, A'), C \cap C')$
   e. Example: ({ 1, 2, 3, 4, 5 }, $\leq$); ({stages of painting by the numbers }, containment)
4. Biba: mathematical dual of BLP
   a. P may read O if $L(P) \leq L(O)$ and $C(P) \subseteq C(O)$
   b. P may write O if $L(O) \leq L(P)$ and $C(O) \subseteq C(P)$
   c. Combined with BLP: continue example
5. Clark-Wilson
   a. Theme: military model does not provide enough controls for commercial fraud, *etc*. because it does not cover the right aspects of integrity
   b. Data items: "Constrained Data Items" (CDI) to which the model applies, "Unconstrained Data Items (UDIs) to which no integrity checks are applied, "Integrity Verification Procedures" (IVP) that verify conformance to the integrity spec when IVP is run, "Transaction Procedures" (TP) takes system from one well-formed state to another
   c. Certification and enforcement rules:
      C1. All IVPs must ensure that all CDIs are in a valid state when the IVP is run
      C2. All TPs must be certified to be valid, and each TP is assocated with a set of CDIs it is authorized to manipulate
      E1. The system must maintain these lists and must ensure only those TPs manipulate those CDIs
      E2: The system must maintain a list of User IDs, TP, and CDIs that that TP can manipulate on behalf of that user, and must ensure only those executions are performed.
      C3. The list of relations in E2 must be certified to meet the separation of duty requirement.
      E3. The sysem must authenticate the identity of each user attempting to execute a TP.
      C4. All TPs must be certified to write to an append-only CDI (the log) all information necessary to reson-struct the operation.
      C5. Any TP taking a UDI as an input must be certified to perform only valid transformations, else no trans-formations, for any possible value of the UDI. The transformation should take the input from a UDI to a CDI, or the UDI is rejected (typically, for edits as the keyboard is a UDI).
      E4. Only the agent permitted to certify entities may change the list of such entities associated with a TP. An agent that can certify an entity may not have any execute rights with respect to that entity.
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