Notes for December 3, 1999

1. Greetings and Felicitations!
2. Puzzle of the Day
3. 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)
4. 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
5. 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
6. Practise: detecting writing
   a. Integrity check files *à 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
7. Network security
   a. Main point: just like a system
8. Review of ISO model
9. Authentication protocols
   a. Kerberos
10. PKI
    a. Certificate-based key management
    b. X.509 model, other models
11. PEM, PGP
    a. Goals: confidentiality, authentication, integrity, non-repudiation (maybe)
    b. Design goals: drop in (not change), works with any RFC 821-conformant MTA and any UA, and exchange messages without prior interaction
    c. Use of Data Exchange Key, Interchange Key
    d. Review of how to do confidentiality, authentication, integrity with public key IKs
    e. Details: canonicalization, security services, printable encoding (PEM)
    f. PGP v. PEM