Outline for May 11, 2000

1. Greetings and felicitations!
   a. Worm in restricted area
2. Confinement problem
   a. Legitimate channels
   b. Storage channels
   c. Covert channels
3. Covert channels
   a. What are they; note probabilistic distribution
   b. Storage vs. timing; give examples
   c. Resource matrix
   d. Formal methods
4. Information flow
   a. Deals with right to disseminate information.
   b. Assume lattice-structured information flow policy (à la BLP); represent as (SC, ≤)
   c. Explicit vs. implicit information flows
5. Program statements; define when “secure”
   a. assignment
   b. compound
   c. alternation
   d. iteration
   e. function call
   f. goto; control flow graph and immediate forward dominator (first block that lies on all paths from the block under consideration and the exit)
   g. composition of above; show compile/parse tree
6. Give examples (copy)
7. Execution-Based with Fixed Classes
   a. verify flows at times of explicit assignment to object
   b. cannot report attempted security violations
8. Execution-Based with Variable Classes
   a. Change variable’s class to allow flow
   b. Fails for implicit
9. Compiler-Based Mechanisms
   a. assures secure execution of each statement
   b. may reject secure mechanisms (not precise)
   c. procedures
   d. arrays
   e. gotos (blocks)
   f. errors

Examples of Compiler-Based Information Flow Enforcement Mechanisms

Here are some examples.

copy2

```
procedure copy2(     x: integer class {x};
                var  y: integer class {x});
     "copy x to y"
     var z: integer class {x});
     begin
          z := 1;                                  Low ≤ z
          y := -1;                                 Low ≤ y
          while z = 1 do                           z ≤ y ⊗ z
               begin
                    y := y + 1;                    y ≤ y
                    if y = 0                       y ≤ z
                              then z := x          x ≤ z
                              else z := 0          Low ≤ z
               end
     end
end copy2.
```

copy2 with goto

```
procedure copy2(     x: integer class {x};
                var  y: integer class {x});
     "copy x to y"
     var z: integer class {x});
     begin
1:   z := 1;                                      b1
     y := -1;
2:   if z ≠ 1 then goto 6;                        b2
3:   y := y + 1;                                  b3
     if y ≠ 0 then goto 5;
4:   z := x;                                      b4
     goto 2;
5:   z := 0;                                      b5
     goto 2;
6:   end
end copy2.
```

IFD(b_1) = b_2
IFD(b_2) = b_6
IFD(b_3) = IFD(b_4) = IFD(b_5) = b_6