ECS 289M Lecture 15

May 3, 2006

Policies Changing Over Time

- Problem: previous analysis assumes static system
  - In real life, ACM changes as system commands issued
- Example: $w \in C^*$ leads to current state
  - *cando*(w, s, z) holds if s can execute z in current state
  - Condition noninterference on *cando*
  - If $\neg$*cando*(w, Lara, "write f"), Lara can't interfere with any other user by writing file f

Generalize Noninterference

- $G \subseteq S$ group of subjects, $A \subseteq Z$ set of commands, p predicate over elements of C^*
- $c_s = (c_1, \ldots, c_n) \in C^*$
- $\pi''(\nu) = \nu$
- $\pi''((c_1, \ldots, c_n)) = (c_1', \ldots, c_n')$
  - $c_i' = \nu$ if $p(c_1', \ldots, c_{i-1}')$ and $c_i = (s, z)$ with $s \in G$ and $z \in A$
  - $c_i' = c_i$ otherwise

Intuition

- $\pi''(c_s) = c_s$
- But if p holds, and element of c_s involves both command in A and subject in G, replace corresponding element of c_s with empty command ν
  - Just like deleting entries from c_s as $\pi_{A,G}$ does earlier

Noninterference

- G, $G' \subseteq S$ groups of subjects, $A \subseteq Z$ set of commands, p predicate over C^*
- Users in G executing commands in A are noninterfering with users in G' under condition p iff, for all $c_s \in C^*$, all $s \in G'$, $proj(s, c_s, \sigma_i) = proj(s, p''(c_s), \sigma_i)$
  - Written A,G :| G' **if** p

Example

- From earlier one, simple security policy based on noninterference:

$\forall(s \in S)\ \forall(z \in Z)$

[$\{z\}, \{s\}$:| S **if** $\neg cando(w, s, z)$]

- If subject can't execute command (the $\neg$*cando* part), subject can't use that command to interfere with another subject

Another Example

- Consider system in which rights can be passed
  - $pass(s, z)$ gives s right to execute z
  - $w_n = v_1, \ldots, v_n$ sequence of $v_i \in C^*$
  - $prev(w_n) = w_{n-1}$; $\text{last}(wn) = v_n$

Policy

- No subject s can use z to interfere if, in previous state, s did not have right to z, and no subject gave it to s

$$
\{ z \}, \{ s \} :| S \textbf{ if}
$$

$$
[\, \neg cando(prev(w), s, z) \wedge
$$

$$
[\, cando(prev(w), s', pass(s, z)) \Rightarrow
$$

$$
\neg last(w) = (s', pass(s, z)) \,]\,]
$$

Effect

- Suppose $s_1 \in S$ can execute *pass*(s_2, z)
- For all $w \in C^*$, *cando*(w, s_1, *pass*(s_2, z)) true
- Initially, *cando*(ν, s_2, z) false
- Let $z' \in Z$ be such that (s_3, z') noninterfering with (s_2, z)
  - So for each w_n with $v_n = (s_3, z')$,

$$cando(w_n, s_2, z) = cando(w_{n-1}, s_2, z)$$

Effect

- Then policy says for all $s \in S$

$$proj(s, ((s_2, z), (s_1, pass(s_2, z)), (s_3, z'), (s_2, z)), \sigma_i) =$$
$$proj(s, ((s_1, pass(s_2, z)), (s_3, z'), (s_2, z)), \sigma_i)$$

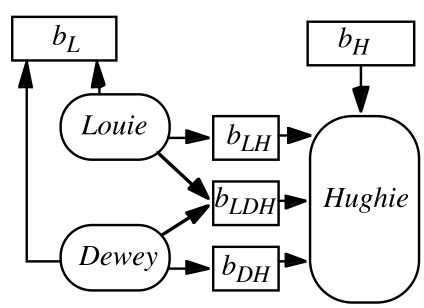
- So s_2's first execution of z does not affect any subject's observation of system

Policy Composition I

- Assumed: Output function of input
  - Means deterministic (else not function)
  - Means uninterruptability (differences in timings can cause differences in states, hence in outputs)
- This result for deterministic, noninterference-secure systems

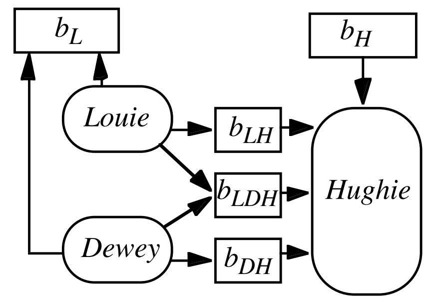
Compose Systems

- Louie, Dewey LOW
- Hughie HIGH
- b_L output buffer
  - Anyone can read it
- b_H input buffer
  - From HIGH source
- Hughie reads from:
  - b_{LH} (Louie writes)
  - b_{LDH} (Louie, Dewey write)
  - b_{DH} (Dewey writes)



Systems Secure

- All noninterference-secure
  - Hughie has no output
    - So inputs don't interfere with it
  - Louie, Dewey have no input
    - So (nonexistent) inputs don't interfere with outputs



Security of Composition

- Buffers finite, sends/receives blocking: composition *not* secure!
  - Example: assume b_{DH}, b_{LH} have capacity 1
- Algorithm:
  1. Louie (Dewey) sends message to b_{LH} (b_{DH})
     - Fills buffer
  2. Louie (Dewey) sends second message to b_{LH} (b_{DH})
  3. Louie (Dewey) sends a 0 (1) to b_L
  4. Louie (Dewey) sends message to b_{LDH}
     - Signals Hughie that Louie (Dewey) completed a cycle

Hughie

- Reads bit from b_H
  - If 0, receive message from b_{LH}
  - If 1, receive message from b_{DH}
- Receive on b_{LDH}
  - To wait for buffer to be filled

Example

- Hughie reads 0 from b_H
  - Reads message from b_{LH}
- Now Louie's second message goes into b_{LH}
  - Louie completes setp 2 and writes 0 into b_L
- Dewey blocked at step 1
  - Dewey cannot write to b_L
- Symmetric argument shows that Hughie reading 1 produces a 1 in b_L
- So, input from b_H copied to output b_L

Nondeducibility

- Noninterference: do state transitions caused by high level commands interfere with sequences of state transitions caused by low level commands?
- Really case about inputs and outputs:
  – Can low level subject deduce *anything* about high level outputs from a set of low level outputs?

Example: 2-Bit System

- *High* operations change only *High* bit
  – Similar for *Low*
- s0 = (0, 0)
- Commands (Heidi, xor1), (Lara, xor0), (Lara, xor1), (Lara, xor0), (Heidi, xor1), (Lara, xor0)
  – Both bits output after each command
- Output is: 00101011110101

Security

- Not noninterference-secure w.r.t. Lara
  - Lara sees output as 0001111
  - Delete *High* and she sees 00111
- But Lara still cannot deduce the commands deleted
  - Don't affect values; only lengths
- So it is deducibly secure
  - Lara can't deduce the commands Heidi gave

Event System

- 4-tuple (E, I, O, T)
  - E set of events
  - $I \subseteq E$ set of input events
  - $O \subseteq E$ set of output events
  - T set of all finite sequences of events legal within system
- E partitioned into H, L
  - H set of *High* events
  - L set of *Low* events

More Events …

- $H \cap I$ set of *High* inputs
- $H \cap O$ set of *High* outputs
- $L \cap I$ set of *Low* inputs
- $L \cap O$ set of *Low* outputs
- T_{Low} set of all possible sequences of *Low* events that are legal within system
- $\pi_L: T \rightarrow T_{Low}$ projection function deleting all *High* inputs from trace
  - *Low* observer should not be able to deduce anything about *High* inputs from trace $t_{Low} \in T_{low}$

Deducibly Secure

- System deducibly secure if, for every trace $t_{Low} \in T_{Low}$, the corresponding set of high level traces contains every possible trace $t \in T$ for which $\pi_L(t) = t_{Low}$
  - Given any t_{Low}, the trace $t \in T$ producing that t_{Low} is equally likely to be *any* trace with $\pi_L(t) = t_{Low}$

Example

- Back to our 2-bit machine
  - Let xor0, xor1 apply to both bits
  - Both bits output after each command
- Initial state: (0, 1)
- Inputs: $1_H 0_L 1_L 0_H 1_L 0_L$
- Outputs: 10 10 01 01 10 10
- Lara (at *Low*) sees: 001100
  - Does not know initial state, so does not know first input; but can deduce fourth input is 0
- Not deducibly secure

Example

- Now *xor0*, *xor1* apply only to state bit with same level as user
- Inputs: $1_H 0_L 1_L 0_H 1_L 0_L$
- Outputs: 1011111011
- Lara sees: 01101
- She cannot deduce *anything* about input
  - Could be $0_H 0_L 1_L 0_H 1_L 0_L$ or $0_L 1_H 1_L 0_H 1_L 0_L$ for example
- Deducibly secure

Security of Composition

- In general: deducibly secure systems not composable
- *Strong noninterference*: deducible security + requirement that no *High* output occurs unless caused by a *High* input
  - Systems meeting this property *are* composable

Example

- 2-bit machine done earlier does not exhibit strong noninterference
  - Because it puts out *High* bit even when there is no *High* input
- Modify machine to output only state bit at level of latest input
  - *Now* it exhibits strong noninterference

Problem

- Too restrictive; it bans some systems that are *obviously* secure
- Example: System *upgrade* reads *Low* inputs, outputs those bits at *High*
  - Clearly deducibly secure: low level user sees no outputs
  - Clearly does not exhibit strong noninterference, as no high level inputs!

Remove Determinism

- Previous assumption
  - Input, output synchronous
  - Output depends only on commands triggered by input
    - Sometimes absorbed into commands …
  - Input processed one datum at a time
- Not realistic
  - In real systems, lots of asynchronous events

Generalized Noninterference

- Nondeterministic systems meeting noninterference property meet *generalized noninterference-secure property*
  - More robust than nondeducible security because minor changes in assumptions affect whether system is nondeducibly secure

Example

- System with *High* Holly, *Low* lucy, text file at *High*
  - File fixed size, symbol b marks empty space
  - Holly can edit file, Lucy can run this program:

```
while true do begin
    n := read_integer_from_user;
    if n > file_length or char_in_file[n] = b then
          print random_character;
    else
          print char_in_file[n];
end;
```

Security of System

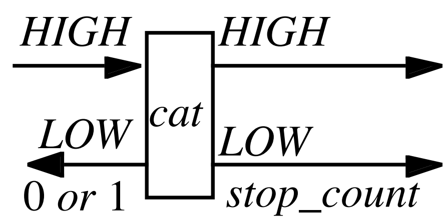
- Not noninterference-secure
  - High level inputs—Holly's changes—affect low level outputs
- *May* be deducibly secure
  - Can Lucy deduce contents of file from program?
  - If output meaningful ("This is right") or close ("Thes is riqht"), yes
  - Otherwise, no
- So deducibly secure depends on which inferences are allowed

Composition of Systems

- Does composing systems meeting generalized noninterference-secure property give you a system that also meets this property?
- Define two systems (*cat*, *dog*)
- Compose them

First System: *cat*

- Inputs, outputs can go left or right
- After some number of inputs, *cat* sends two outputs
  - First *stop_count*
  - Second parity of *High* inputs, outputs

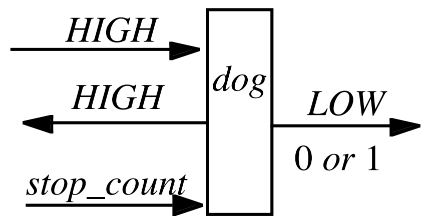


Noninterference-Secure?

- If even number of *High* inputs, output could be:
  - 0 (even number of outputs)
  - 1 (odd number of outputs)
- If odd number of *High* inputs, output could be:
  - 0 (odd number of outputs)
  - 1 (even number of outputs)
- High level inputs do not affect output
  - So noninterference-secure

Second System: *dog*

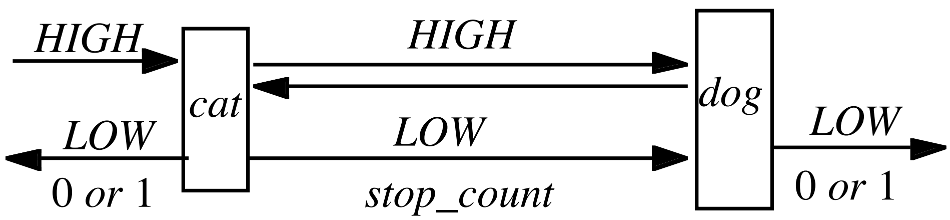
- High outputs to left
- Low outputs of 0 or 1 to right
- *stop_count* input from the left
  - When it arrives, *dog* emits 0 or 1



Noninterference-Secure?

- When *stop_count* arrives:
  - May or may not be inputs for which there are no corresponding outputs
  - Parity of *High* inputs, outputs can be odd or even
  - Hence *dog* emits 0 or 1
- High level inputs do not affect low level outputs
  - So noninterference-secure

Compose Them



- Once sent, message arrives
  - But *stop_count* may arrive before all inputs have generated corresponding outputs
  - If so, even number of *High* inputs and outputs on *cat*, but odd number on *dog*
- Four cases arise

The Cases

- *cat*, odd number of inputs, outputs; *dog*, even number of inputs, odd number of outputs
  - Input message from *cat* not arrived at *dog*, contradicting assumption
- *cat*, even number of inputs, outputs; *dog*, odd number of inputs, even number of outputs
  - Input message from *dog* not arrived at *cat*, contradicting assumption

The Cases

- cat, odd number of inputs, outputs; dog, odd number of inputs, even number of outputs
  - dog sent even number of outputs to cat, so cat has had at least one input from left
- cat, even number of inputs, outputs; dog, even number of inputs, odd number of outputs
  - dog sent odd number of outputs to cat, so cat has had at least one input from left

The Conclusion

- Composite system *catdog* emits 0 to left, 1 to right (or 1 to left, 0 to right)
  - Must have received at least one input from left
- Composite system *catdog* emits 0 to left, 0 to right (or 1 to left, 1 to right)
  - Could not have received any from left
- So, *High* inputs affect *Low* outputs
  - Not noninterference-secure

Feedback-Free Systems

- System has n distinct components
- Components c_i, c_j connected if any output of c_i is input to c_j
- System is *feedback-free* if for all c_i connected to c_j, c_j not connected to any c_i
  – Intuition: once information flows from one component to another, no information flows back from the second to the first

Feedback-Free Security

- *Theorem*: A feedback-free system composed of noninterference-secure systems is itself noninterference-secure

Some Feedback

- *Lemma*: A noninterference-secure system can feed a high level output *o* to a high level input *i* if the arrival of *o* at the input of the next component is delayed until *after* the next low level input or output
- *Theorem*: A system with feedback as described in the above lemma and composed of noninterference-secure systems is itself noninterference-secure