Lecture #16

- Composition
- Nondeducibility
- Generalized Noninterference
- Restrictiveness
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 step 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
- $\sigma_0 = (0, 0)$
- 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 \( t_{\text{Low}} \in T_{\text{Low}} \), the corresponding set of high level traces contains every possible trace \( t \in T \) for which \( \pi_L(t) = t_{\text{Low}} \)
  - Given any \( t_{\text{Low}} \), the trace \( t \in T \) producing that \( t_{\text{Low}} \) is equally likely to be any trace with \( \pi_L(t) = t_{\text{Low}} \)
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: $1_H0_L1_L0_H1_L0_L$
- Outputs: 1011111011
- Lara sees: 01101
- She cannot deduce *anything* about input
  - Could be $0_H0_L1_L0_H1_L0_L$ or $0_L1_H1_L0_H1_L0_L$ 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 $\texttt{b}$ marks empty space
  - Holly can edit file, Lucy can run this program:

```plaintext
while true do begin
  n := read_integer_from_user;
  if n > file_length or char_in_file[n] = \texttt{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 righ”), 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

```
   HIGH          HIGH
     ↓            ↓
  cat  0 or 1   cat
     ↑            ↑
   LOW          LOW
```

`stop_count` 0 or 1
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 \textit{stop\_count} arrives:
  – May or may not be inputs for which there are no corresponding outputs
  – Parity of \textit{High} inputs, outputs can be odd or even
  – Hence \textit{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 \textit{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.
Why Didn’t They Work?

• For compositions to work, machine must act same way regardless of what precedes low level input (high, low, nothing)

• *dog* does not meet this criterion
  – If first input is *stop_count*, *dog* emits 0
  – If high level input precedes *stop_count*, *dog* emits 0 or 1
State Machine Model

- 2-bit machine, levels High, Low, meeting 4 properties:

1. For every input $i_k$, state $\sigma_j$, there is an element $c_m \in C^*$ such that $T^*(c_m, \sigma_j) = \sigma_n$, where $\sigma_n \neq \sigma_j$

   - $T^*$ is total function, inputs and commands always move system to a different state
Property 2

- There is an equivalence relation \( \equiv \) such that:
  - If system in state \( \sigma_i \) and high level sequence of inputs causes transition from \( \sigma_i \) to \( \sigma_j \), then \( \sigma_i \equiv \sigma_j \)
  - If \( \sigma_i \equiv \sigma_j \) and low level sequence of inputs \( i_1, \ldots, i_n \) causes system in state \( \sigma_i \) to transition to \( \sigma_i' \), then there is a state \( \sigma_j' \) such that \( \sigma_i' \equiv \sigma_j' \) and the inputs \( i_1, \ldots, i_n \) cause system in state \( \sigma_j \) to transition to \( \sigma_j' \)

- \( \equiv \) holds if low level projections of both states are same
Property 3

• Let $\sigma_i \equiv \sigma_j$. If high level sequence of outputs $o_1, \ldots, o_n$ indicate system in state $\sigma_i$ transitioned to state $\sigma_i'$, then for some state $\sigma_j'$ with $\sigma_j' \equiv \sigma_i'$, high level sequence of outputs $o_1', \ldots, o_m'$ indicates system in $\sigma_j$ transitioned to $\sigma_j'$
  
  – High level outputs do not indicate changes in low level projection of states
Property 4

- Let $\sigma_i \equiv \sigma_j$, let $c, d$ be high level output sequences, $e$ a low level output. If $ced$ indicates system in state $\sigma_i$ transitions to $\sigma_i'$, then there are high level output sequences $c'$ and $d'$ and state $\sigma_j'$ such that $c'ed'$ indicates system in state $\sigma_j$ transitions to state $\sigma_j'$
  - Intermingled low level, high level outputs cause changes in low level state reflecting low level outputs only
Restrictiveness

• System is restrictive if it meets the preceding 4 properties
Composition

- Intuition: by 3 and 4, high level output followed by low level output has same effect as low level input, so composition of restrictive systems should be restrictive
Composite System

- System $M_1$’s outputs are $M_2$’s inputs
- $\mu_{1i}, \mu_{2i}$ states of $M_1, M_2$
- States of composite system pairs of $M_1, M_2$ states ($\mu_{1i}, \mu_{2i}$)
- $e$ event causing transition
- $e$ causes transition from state ($\mu_{1a}, \mu_{2a}$) to state ($\mu_{1b}, \mu_{2b}$) if any of 3 conditions hold
Conditions

1. \( M_1 \) in state \( \mu_{1a} \) and \( e \) occurs, \( M_1 \) transitions to \( \mu_{1b} \); \( e \) not an event for \( M_2 \); and \( \mu_{2a} = \mu_{2b} \)

2. \( M_2 \) in state \( \mu_{2a} \) and \( e \) occurs, \( M_2 \) transitions to \( \mu_{2b} \); \( e \) not an event for \( M_1 \); and \( \mu_{1a} = \mu_{1b} \)

3. \( M_1 \) in state \( \mu_{1a} \) and \( e \) occurs, \( M_1 \) transitions to \( \mu_{1b} \); \( M_2 \) in state \( \mu_{2a} \) and \( e \) occurs, \( M_2 \) transitions to \( \mu_{2b} \); \( e \) is input to one machine, and output from other
Intuition

• Event causing transition in composite system causes transition in at least 1 of the components

• If transition occurs in exactly one component, event must not cause transition in other component when not connected to the composite system
Equivalence for Composite

- Equivalence relation for composite system:
  \[(\sigma_a, \sigma_b) \equiv_C (\sigma_c, \sigma_d) \text{ iff } \sigma_a \equiv \sigma_c \text{ and } \sigma_b \equiv \sigma_d\]
- Corresponds to equivalence relation in property 2 for component system