ECS 235B Module 40

Nondeducibility
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*
• $\sigma_0 = (0, 0)$
• Sequence of 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 outputs 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 \textit{High} events
  • \(L\) set of \textit{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 all traces $t_{\text{Low}} \in T_{\text{Low}}$, 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: 2-Bit Machine

• Let xor0, xor1 apply to both bits, and both bits output after each command
• Initial state: (0, 1)
• Inputs: 1_H0_L1_L0_H1_L0_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: 2-Bit Machine

- 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 ✧ 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] = ✧ 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 \((\text{cat}, \text{dog})\)
• 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

```
<table>
<thead>
<tr>
<th>HIGH</th>
<th>HIGH</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOW</td>
<td>LOW</td>
</tr>
<tr>
<td>0 or 1</td>
<td><code>stop_count</code></td>
</tr>
</tbody>
</table>
```
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 (i.e., no HIGH inputs)
- So, *High* inputs affect *Low* outputs
  - Not noninterference-secure