Lecture 13, April 27

ECS 235B, Foundations of Computer and Information Security

Spring Quarter 2026

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: 00 10 10 11 11 01 01
  - Short for (0, 0), (1, 0), (1, 0), (1, 1), (1, 1), (0, 1), (0, 1)

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 *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
- π_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 $t_{Low} \in T_{Low}$, 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: 2-Bit Machine

- Let *xor0*, *xor1* apply to both bits, and 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 (as she did not give it and the third and fourth outputs are the same, so it's an xor with input of 0)
- Not deducibly secure

Example: 2-Bit Machine

- 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: 10 11 11 10 11
- Lara sees: 01101
- She cannot deduce *anything* about input
  - Equally likely to 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 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
- But that is 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* Lara, text file at *High*
  - File fixed size, symbol ✧ marks empty space
  - Holly can edit file, Lara can run this program:

```
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

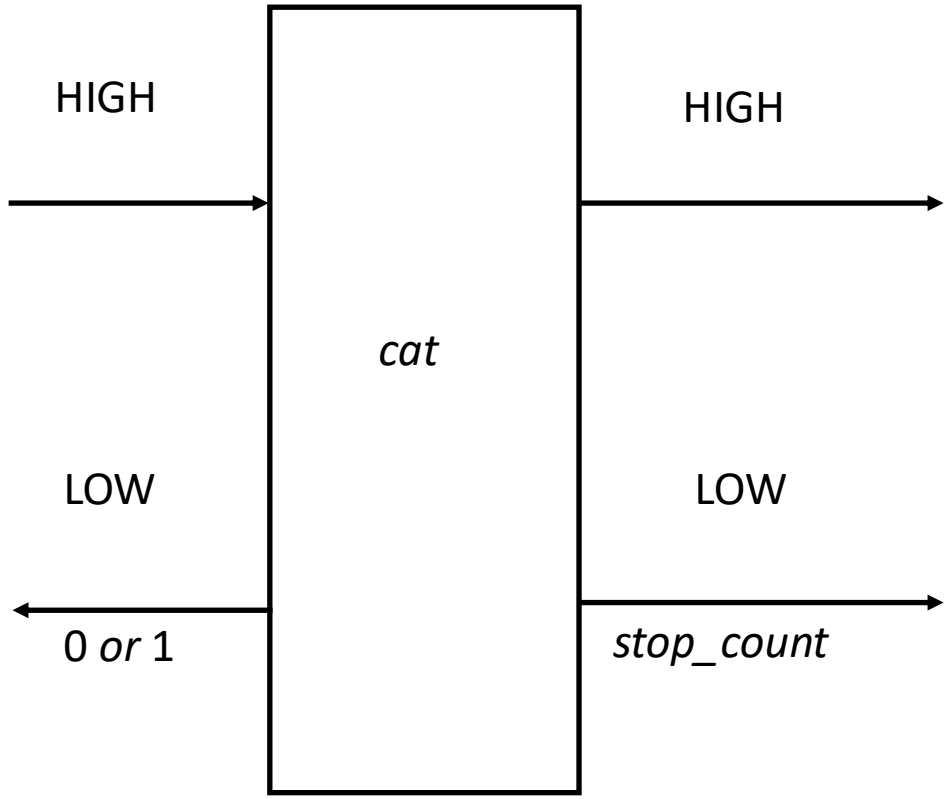
- Not noninterference-secure
  - High level inputs—Holly's changes—affect low level outputs
- *May* be deducibly secure
  - Can Lara deduce contents of file from program?
  - If output meaningful ("This is right") or close ("Thes is riqht"), yes
  - Otherwise, no
- So deducibly secure depends on which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

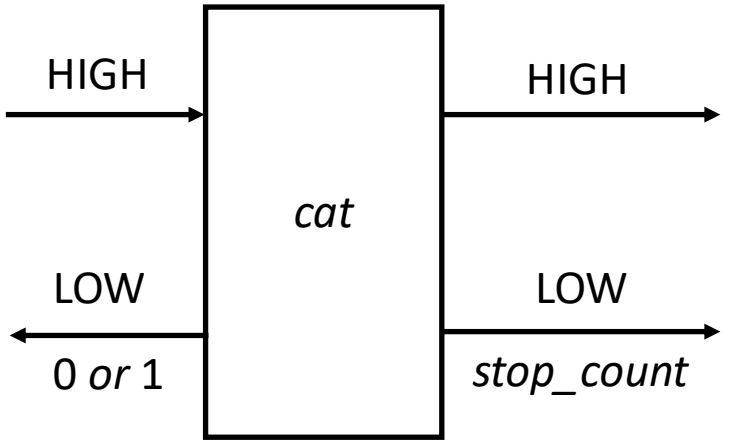
First System: *cat*

- 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



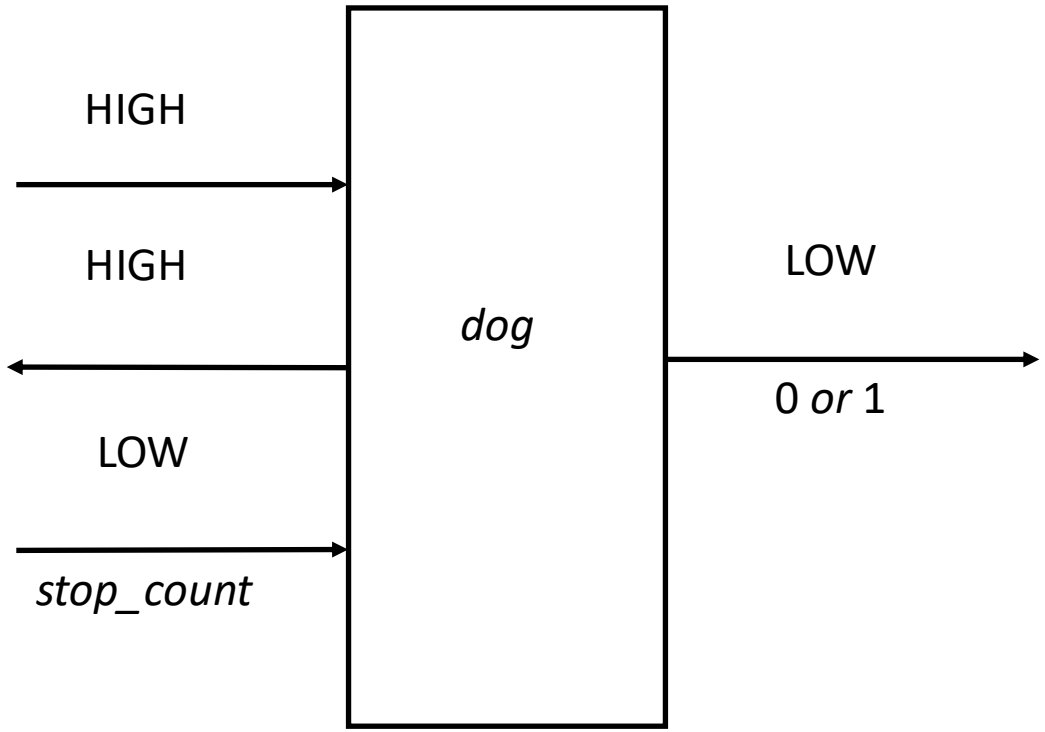
Noninterference-Secure?

- 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



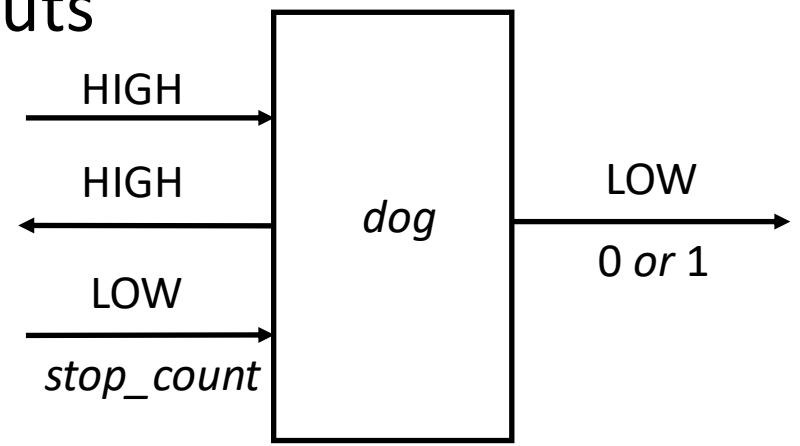
Second System: *dog*

- 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

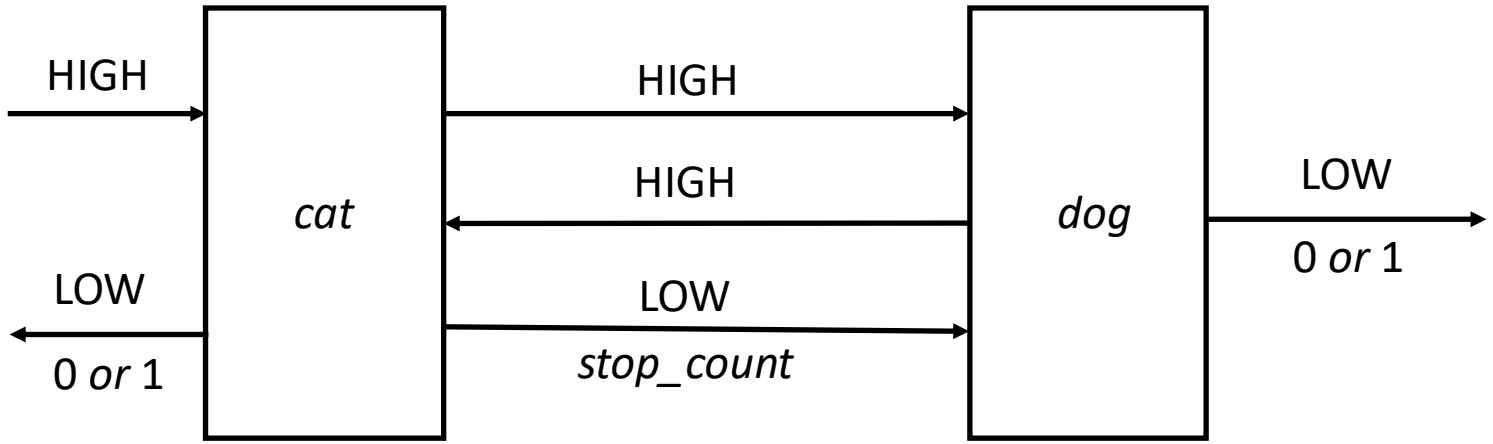


Noninterference-Secure?

- 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 (i.e., no HIGH inputs)
- So, *High* inputs affect *Low* outputs
  - Not noninterference-secure

Feedback-Free Systems

- System has *n* distinct components
- Components c_i, c_j are *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 output *o* to a HIGH input *i* if the arrival of *o* at the input of the next component is delayed until *after* the next LOW 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 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*, meet 4 properties:

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

   So T^* is total function, inputs and commands always move system to a different state

Property 2

2. There is an equivalence relation $\equiv$ such that:
   a. If system in state σ_i and HIGH sequence of inputs causes transition from σ_i to σ_j, then $\sigma_i \equiv \sigma_j$
      - 2 states equivalent if either reachable from the other state using only HIGH commands
   b. If $\sigma_i \equiv \sigma_j$ and LOW sequence of inputs i_1, …, i_n causes system in state σ_i to transition to σ_i', then there is a state σ_j' such that $\sigma_i' \equiv \sigma_j'$ and inputs i_1, …, i_n cause system in state σ_j to transition to σ_j'
      - States resulting from giving same LOW commands to the two equivalent original states have same LOW projection

$\equiv$ holds if LOW projections of both states are same

- If 2 states equivalent, HIGH commands do not affect LOW projections

Property 3

3. Let $\sigma_i \equiv \sigma_j$. If sequence of HIGH outputs $o_1, \ldots, o_n$ indicate system in state σ_i transitioned to state σ_i', then for some state σ_j' with $\sigma_j' \equiv \sigma_i'$, sequence of HIGH outputs $o_1', \ldots, o_m'$ indicates system in σ_j transitioned to σ_j'

- HIGH outputs do not indicate changes in LOW projection of states

Property 4

4. Let $\sigma_i \equiv \sigma_j$, let *c*, *d* be HIGH output sequences, *e* a LOW output. If output sequence *ced* indicates system in state σ_i transitions to σ_i', then there are HIGH output sequences *c*' and *d*' and state σ_j' such that *c*'*ed*' indicates system in state σ_j transitions to state σ_j'

- Intermingled LOW, HIGH outputs cause changes in LOW state reflecting LOW outputs only

Restrictiveness

- System is *restrictive* if it meets the preceding 4 properties

Composition

- Intuition: by 3 and 4, HIGH output followed by LOW output has same effect as the LOW input, so composition of restrictive systems should be restrictive

Composite System

- System M_1's outputs are acceptable as M_2's inputs
- μ_{1i}, μ_{2i} states of M_1, M_2
- States of composite system pairs of M_1, M_2 states (μ_{1i}, μ_{2i})
- *e* event causing transition
- *e* causes transition from state (μ_{1a}, μ_{2a}) to state (μ_{1b}, μ_{2b}) if any of 3 conditions hold

Conditions

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

Theorem

The system resulting from the composition of two restrictive systems is itself restrictive

Side Channels

A *side channel* is set of characteristics of a system, from which adversary can deduce confidential information about system or a composition.

- Consider information to be derived as HIGH
- Consider information obtained from set of characteristics as LOW
- Attack is to deduce HIGH values from LOW values only
- Implication: attack works on systems not deducibly secure

Types of Side Channel Attacks

- *Passive*: Only observe system; deduce results from observations
- *Active*: Disrupt system in some way, causing it to react; deduce results from measurements of disruption

Example of a Passive Attack

- Fast modular exponentiation:

```
x := 1; atmp := a;
for i := 0 to k-1 do begin
   if z_i = 1 then x := (x * atmp) mod n;
   atmp := (atmp * atmp) mod n;
end;
result := x;
```

- If bit is 1, there are 2 multiplications; if it is 0, only one
  - And the extra multiplication takes time
- Can determine bits of the confidential exponent by measuring computation time

Example of an Active Attack

Background

- Goal: derive information from characteristics of memory accesses in chip
- Intel x86 caches
  - Each core has 2 levels, L1 and L2
  - Chip itself has third cache (L3 or LLC)
  - These are hierarchical: miss in L1 goes to L2, miss in L2 goes to L3, miss in L3 goes to memory
  - Caches are inclusive (so L3 has copies of data in L2 and L1)
- Processes share pages

Example: Active Attack

Phase 1

- Flush a set of bytes (called a *line*) from cache to clear it from all 3 caches
  - The disruption

Phase 2

- Wait until victim has chance to access that memory line

Phase 3

- Reload the line
  - If victim did this already, time is short as data comes from L3 cache
  - Otherwise time is longer as memory fetch is required

Example: Active Attack

What happened

- Used to trace execution of GnuPG on a physical machine
- Derived bits of a 2048 bit private key; max of 190 bits incorrect
- Repeated experiment on virtual machine
- Error rates increased
  - On one system, average error rate increased from 1.41 bits to 26.55 bits
  - On another system, average error rate increased from 25.12 bits to 66.12 bits

Model

Components

- *Primitive*: instantiation of computation
- *Device*: system doing the computation
- *Physical observable*: output being observed
- *Leakage function*: captures characteristics of side channel and mechanism to monitor the physical observables
- *Implementation function*: instantiation of both device, leakage function
- *Side channel adversary*: algorithm that queries implementation to get outputs from leakage function

Example

- First one (passive attack) divided leakage function into two parts
  - *Signal* was variations in output due to bit being derived
  - *Noise* was variations due to other factors (imprecisions in measurements, etc.)
- Second one (active attack) had leakage function acting in different ways
  - Physical machine: one chip used more advanced optimizations, thus more noise
  - Virtual machine: more variations due to extra computations running the virtual machines, hence more noise

Example: Electromagnetic Radiation

- CRT video display produces radiation that can be measured
- Using various equipment and a black and white TV, van Eck could reconstruct the images
  - Reconstructed pictures on video display units in buildings
- E-voting system with audio activated (as it would be for visually impaired voters) produced interference with sound from a nearby transistor radio
  - Testers believed changes in the sound due to the interference could be used to determine how voter was voting