

Lecture 12, April 24

ECS 235B, Foundations of Computer and Information Security
Spring Quarter 2026

Noninterference

- Intuition: If set of outputs Lucy can see corresponds to set of inputs she can see, there is no interference
- Formally: $G, G' \subseteq S, G \neq G'; A \subseteq Z$; users in G executing commands in A are *noninterfering* with users in G' iff for all $c_s \in C^*$, and for all $s \in G'$,

$$\text{proj}(s, c_s, \sigma_i) = \text{proj}(s, \pi_{G,A}(c_s), \sigma_i)$$

- Written $A, G :| G'$

Example: 2-Bit Machine

- Let $c_s = ((Heidi, xor0), (Lucy, xor1), (Heidi, xor1))$ and $\sigma_0 = (0, 1)$
 - As before
- Take $G = \{ Heidi \}$, $G' = \{ Lucy \}$, $A = \emptyset$
- $\pi_{Heidi}(c_s) = (Lucy, xor1)$
 - So $proj(Lucy, \pi_{Heidi}(c_s), \sigma_0) = 0$
- $proj(Lucy, c_s, \sigma_0) = 101$
- So $\{ Heidi \} :| \{ Lucy \}$ is false
 - Makes sense; commands issued to change H bit also affect L bit

Example

- Same as before, but Heidi's commands affect H bit only, Lucy's the L bit only
- Output is $0_H 0_L 1_H$
- $\pi_{\text{Heidi}}(c_s) = (\text{Lucy}, \text{xor}1)$
 - So $\text{proj}(\text{Lucy}, \pi_{\text{Heidi}}(c_s), \sigma_0) = 0$
- $\text{proj}(\text{Lucy}, c_s, \sigma_0) = 0$
- So $\{ \text{Heidi} \} :| \{ \text{Lucy} \}$ is true
 - Makes sense; commands issued to change H bit now do not affect L bit

Security Policy

- Partitions systems into authorized, unauthorized states
- Authorized states have no forbidden interferences
- Hence a *security policy* is a set of noninterference assertions
 - See previous definition

Alternative Development

- System X is a set of protection domains $D = \{ d_1, \dots, d_n \}$
- When command c executed, it is executed in protection domain $dom(c)$
- Give alternate versions of definitions shown previously

Security Policy

- $D = \{ d_1, \dots, d_n \}$, d_i a protection domain
- $r: D \times D$ a reflexive relation
- Then r defines a security policy
- Intuition: defines how information can flow around a system
 - $d_i r d_j$ means info can flow from d_i to d_j
 - $d_i r d_i$ as info can flow within a domain

Projection Function

- π' analogue of π , earlier
- Commands, subjects absorbed into protection domains
- $d \in D, c \in C, c_s \in C^*$
- $\pi'_d(v) = v$
- $\pi'_d(c_s c) = \pi'_d(c_s) c$ if $dom(c)rd$
- $\pi'_d(c_s c) = \pi'_d(c_s)$ otherwise
- Intuition: if executing c interferes with d , then c is visible; otherwise, as if c never executed

Noninterference-Secure

- System has set of protection domains D
- System is *noninterference-secure with respect to policy r* if

$$P^*(c, T^*(c_s, \sigma_0)) = P^*(c, T^*(\pi'_d(c_s), \sigma_0))$$

- Intuition: if executing c_s causes the same transitions for subjects in domain d as does its projection with respect to domain d , then no information flows in violation of the policy

Output-Consistency

- $c \in C, dom(c) \in D$
- $\sim_{dom(c)}$ equivalence relation on states of system X
- $\sim_{dom(c)}$ *output-consistent* if

$$\sigma_a \sim_{dom(c)} \sigma_b \implies P(c, \sigma_a) = P(c, \sigma_b)$$

- Intuition: states are output-consistent if for subjects in $dom(c)$, projections of outputs for both states after c are the same

Lemma

- Let $T^*(c_s, \sigma_0) \sim^d T^*(\pi'_d(c_s), \sigma_0)$ for $c \in C$
- If \sim^d output-consistent, then system is noninterference-secure with respect to policy r

Proof

- $d = \text{dom}(c)$ for $c \in \mathcal{C}$

- By definition of output-consistent,

$$T^*(c_s, \sigma_0) \sim^d T^*(\pi'_d(c_s), \sigma_0)$$

implies

$$P^*(c, T^*(c_s, \sigma_0)) = P^*(c, T^*(\pi'_d(c_s), \sigma_0))$$

- This is definition of noninterference-secure with respect to policy r

Unwinding Theorem

- Links security of sequences of state transition commands to security of individual state transition commands
- Allows you to show a system design is multilevel-secure by showing it matches specs from which certain lemmata derived
 - Says *nothing* about security of system, because of implementation, operation, etc. issues

Locally Respects

- r is a policy
- System X locally respects r if $dom(c)$ being noninterfering with $d \in D$ implies $\sigma_a \sim^d T(c, \sigma_a)$
- Intuition: when X locally respects r , applying c under policy r to system X has no effect on domain d

Transition-Consistent

- r policy, $d \in D$
- If $\sigma_a \sim^d \sigma_b$ implies $T(c, \sigma_a) \sim^d T(c, \sigma_b)$, system X is *transition-consistent* under r
- Intuition: command c does not affect equivalence of states under policy r

Theorem

- r policy, X system that is output consistent, transition consistent, and locally respects r
- Then X noninterference-secure with respect to policy r
- Significance: basis for analyzing systems claiming to enforce noninterference policy
 - Establish conditions of theorem for particular set of commands, states with respect to some policy, set of protection domains
 - Noninterference security with respect to r follows

Proof

Must show $\sigma_a \sim^d \sigma_b \Rightarrow T^*(c_s, \sigma_a) \sim^d T^*(\pi'_d(c_s), \sigma_b)$

- Induct on length of c_s
- Basis: if $c_s = v$, then $T^*(c_s, \sigma_a) = \sigma_a$ and $\pi'_d(v) = v$; claim holds
- Hypothesis: for $c_s = c_1 \dots c_n$, $\sigma_a \sim^d \sigma_b \Rightarrow T^*(c_s, \sigma_a) \sim^d T^*(\pi'_d(c_s), \sigma_b)$

Induction Step

- Consider $c_s c_{n+1}$. Assume $\sigma_a \sim^d \sigma_b$ and look at $T^*(\pi'_d(c_s c_{n+1}), \sigma_b)$
- 2 cases:
 - $dom(c_{n+1})rd$ holds
 - $dom(c_{n+1})rd$ does not hold

$dom(c_{n+1})rd$ Holds

$$T^*(\pi'_d(c_s c_{n+1}), \sigma_b) = T^*(\pi'_d(c_s) c_{n+1}, \sigma_b) = T(c_{n+1}, T^*(\pi'_d(c_s), \sigma_b))$$

- By definition of T^* and π'_d

$$\sigma_a \sim^d \sigma_b \Rightarrow T(c_{n+1}, \sigma_a) \sim^d T(c_{n+1}, \sigma_b)$$

- As X transition-consistent

$$T(c_{n+1}, T^*(c_s, \sigma_a)) \sim^d T(c_{n+1}, T^*(\pi'_d(c_s), \sigma_b))$$

- By transition-consistency and IH

$$T(c_{n+1}, T^*(c_s, \sigma_a)) \sim^d T^*(\pi'_d(c_s c_{n+1}), \sigma_b)$$

- By substitution from earlier equality

$$T^*(c_s c_{n+1}, \sigma_a) \sim^d T^*(\pi'_d(c_s c_{n+1}), \sigma_b)$$

- By definition of T^*

proving hypothesis

$dom(c_{n+1})rd$ Does Not Hold

$$T^*(\pi'_d(c_s c_{n+1}), \sigma_b) = T^*(\pi'_d(c_s), \sigma_b)$$

- By definition of π'_d

$$T^*(c_s, \sigma_a) = T^*(\pi'_d(c_s c_{n+1}), \sigma_b)$$

- By above and IH

$$T(c_{n+1}, T^*(c_s, \sigma_a)) \sim^d T^*(c_s, \sigma_a)$$

- As X locally respects r , $\sigma \sim^d T(c_{n+1}, \sigma)$ for any σ

$$T(c_{n+1}, T^*(c_s, \sigma_a)) \sim^d T^*(\pi'_d(c_s c_{n+1}), \sigma_b)$$

- Substituting back

proving hypothesis

Finishing Proof

- Take $\sigma_a = \sigma_b = \sigma_0$, so from claim proved by induction,

$$T^*(c_s, \sigma_0) \sim^d T^*(\pi'_d(c_s), \sigma_0)$$

- By previous lemma, as X (and so \sim^d) output consistent, then X is noninterference-secure with respect to policy r

Access Control Matrix

- Example of interpretation
- Given: access control information
- Question: are given conditions enough to provide noninterference security?
- Assume: system in a particular state
 - Encapsulates values in ACM

ACM Model

- Objects $L = \{ l_1, \dots, l_m \}$
 - Locations in memory
- Values $V = \{ v_1, \dots, v_n \}$
 - Values that L can assume
- Set of states $\Sigma = \{ \sigma_1, \dots, \sigma_k \}$
- Set of protection domains $D = \{ d_1, \dots, d_j \}$

Functions

- *value*: $L \times \Sigma \rightarrow V$
 - returns value v stored in location l when system in state σ
- *read*: $D \rightarrow 2^V$
 - returns set of objects observable from domain d
- *write*: $D \rightarrow 2^V$
 - returns set of objects observable from domain d

Interpretation of ACM

- Functions represent ACM
 - Subject s in domain d , object o
 - $r \in A[s, o]$ if $o \in \text{read}(d)$
 - $w \in A[s, o]$ if $o \in \text{write}(d)$

- Equivalence relation:

$$[\sigma_a \sim^{dom(c)} \sigma_b] \Leftrightarrow [\forall l_i \in \text{read}(d) [\text{value}(l_i, \sigma_a) = \text{value}(l_i, \sigma_b)]]$$

- You read *exactly* the same values from the same locations in both states

Enforcing Policy r

- 5 requirements
 - 3 general ones describing dependence of commands on rights over input and output
 - Hold for all ACMs and policies
 - 2 that are specific to some security policies
 - Hold for *most* policies

Enforcing Policy r : General Requirements

1. Output of command c executed in domain $dom(c)$ depends only on values for which subjects in $dom(c)$ have read access
 - $\sigma_a \sim^{dom(c)} \sigma_b \Rightarrow P(c, \sigma_a) = P(c, \sigma_b)$
2. If c changes l_j , then c can only use values of objects in $read(dom(c))$ to determine new value
 - $[\sigma_a \sim^{dom(c)} \sigma_b \wedge (value(l_j, T(c, \sigma_a)) \neq value(l_j, \sigma_a) \vee value(l_j, T(c, \sigma_b)) \neq value(l_j, \sigma_b))] \Rightarrow value(l_j, T(c, \sigma_a)) = value(l_j, T(c, \sigma_b))$
3. If c changes l_j , then $dom(c)$ provides subject executing c with write access to l_j
 - $value(l_j, T(c, \sigma_a)) \neq value(l_j, \sigma_a) \Rightarrow l_j \in write(dom(c))$

Enforcing Policies r : Specific to Policy

4. If domain u can interfere with domain v , then every object that can be read in u can also be read in v ; so if object o cannot be read in u , but can be read in v and object o' in u can be read in v , then info flows from o to o' , then to v

$$[u, v \in D \wedge urv] \Rightarrow read(u) \subseteq read(v)$$

5. Subject s can write object o in v , subject s' can read o in u , then domain v can interfere with domain u

$$[l_j \in read(u) \wedge l_j \in write(v)] \Rightarrow vru$$

Theorem

- Let X be a system satisfying these five conditions. Then X is noninterference-secure with respect to r
- Proof: must show X output-consistent, locally respects r , transition-consistent
 - Then by unwinding theorem, this theorem holds

Output-Consistent

- Take equivalence relation to be \sim^d , first condition *is* definition of output-consistent

Locally Respects r

- Proof by contradiction: assume $(dom(c), d) \notin r$ (meaning but $\sigma_a \sim^d T(c, \sigma_a)$ does not hold
- Some object has value changed by c :
$$\exists l_i \in read(d) [value(l_i, \sigma_a) \neq value(l_i, T(c, \sigma_a))]$$
- By condition 3: $l_i \in write(dom(c))$
 - Condition 3: $value(l_i, T(c, \sigma_a)) \neq value(l_i, \sigma_a) \Rightarrow l_i \in write(dom(c))$
- By condition 5: $dom(c)rd$, contradiction
 - Condition 5: $[l_i \in read(u) \wedge l_i \in write(v)] \Rightarrow vru$
- So $\sigma_a \sim^d T(c, \sigma_a)$ holds, meaning X locally respects r

Transition Consistency

- Assume $\sigma_a \sim^d \sigma_b$
- Must show $value(l_i, T(c, \sigma_a)) = value(l_i, T(c, \sigma_b))$ for $l_i \in read(d)$
- 3 cases dealing with change that c makes in l_i in states σ_a, σ_b :
 - $value(l_i, T(c, \sigma_a)) \neq value(l_i, \sigma_a)$
 - $value(l_i, T(c, \sigma_b)) \neq value(l_i, \sigma_b)$
 - Neither of the above two hold

Case 1: $value(l_i, T(c, \sigma_d)) \neq value(l_i, \sigma_d)$

- By condition 3: $l_i \in write(dom(c))$
 - Condition 3: $value(l_i, T(c, \sigma_d)) \neq value(l_i, \sigma_d) \Rightarrow l_i \in write(dom(c))$
- As $l_i \in read(d)$, condition 5 says $dom(c)rd$
 - Condition 5: $[l_i \in read(u) \wedge l_i \in write(v)] \Rightarrow vru$
- By condition 4: $read(dom(c)) \subseteq read(d)$
 - Condition 4: $[u, v \in D \wedge urv] \Rightarrow read(u) \subseteq read(v)$

Case 1: $value(l_i, T(c, \sigma_a)) \neq value(l_i, \sigma_a)$

- As $\sigma_a \sim^d \sigma_b$ and $dom(c)rd$, $\sigma_a \sim^{dom(c)} \sigma_b$
- By condition 2: $value(l_i, T(c, \sigma_a)) = value(l_i, T(c, \sigma_b))$
 - Condition 2: $[\sigma_a \sim^{dom(c)} \sigma_b \wedge (value(l_i, T(c, \sigma_a)) \neq value(l_i, \sigma_a) \vee value(l_i, T(c, \sigma_b)) \neq value(l_i, \sigma_b))] \Rightarrow value(l_i, T(c, \sigma_a)) = value(l_i, T(c, \sigma_b))$
- So $T(c, \sigma_a) \sim^{dom(c)} T(c, \sigma_b)$, as desired

Case 2: $value(l_i, T(c, \sigma_b)) \neq value(l_i, \sigma_b)$

- By condition 3: $l_i \in write(dom(c))$
 - Condition 3: $value(l_i, T(c, \sigma_a)) \neq value(l_i, \sigma_a) \Rightarrow l_i \in write(dom(c))$
- As $l_i \in read(d)$, condition 5 says $dom(c)rd$
 - Condition 5: $[l_i \in read(u) \wedge l_i \in write(v)] \Rightarrow vru$
- By condition 4: $read(dom(c)) \subseteq read(d)$
 - Condition 4: $[u, v \in D \wedge urv] \Rightarrow read(u) \subseteq read(v)$

Case 2: $value(l_i, T(c, \sigma_b)) \neq value(l_i, \sigma_b)$

- As $\sigma_a \sim^d \sigma_b$ and $dom(c)rd$, $\sigma_a \sim^{dom(c)} \sigma_b$
- By condition 2: $value(l_i, T(c, \sigma_a)) = value(l_i, T(c, \sigma_b))$
 - Condition 2: $[\sigma_a \sim^{dom(c)} \sigma_b \wedge (value(l_i, T(c, \sigma_a)) \neq value(l_i, \sigma_a) \vee value(l_i, T(c, \sigma_b)) \neq value(l_i, \sigma_b))] \Rightarrow value(l_i, T(c, \sigma_a)) = value(l_i, T(c, \sigma_b))$
- So $T(c, \sigma_a) \sim^{dom(c)} T(c, \sigma_b)$, as desired

Case 3: Neither of the Previous Two Hold

- This means the two conditions below hold:

- $value(l_i, T(c, \sigma_a)) = value(l_i, \sigma_a)$
- $value(l_i, T(c, \sigma_b)) = value(l_i, \sigma_b)$

- Interpretation of $\sigma_a \sim^d \sigma_b$ is:

$$(\forall l_i \in read(d)) [value(l_i, \sigma_a) = value(l_i, \sigma_b)]$$

- So $T(c, \sigma_a) \sim^d T(c, \sigma_b)$, as desired

In all 3 cases, X transition-consistent

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 $\neg cando(w, \text{Lara}, \text{"write } f\text{"})$, Lara can't interfere with any other user by writing file f

Generalize Noninterference

- $G \subseteq S$ set of subjects, $A \subseteq Z$ set of commands, p predicate over elements of C^*
- $c_s = (c_1, \dots, c_n) \in C^*$
- $\pi''(v) = v$
- $\pi''((c_1, \dots, c_n)) = (c_1', \dots, c_n')$, where
 - $c_i' = v$ if $p(c_1', \dots, c_{i-1}')$ 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

- $G, G' \subseteq S$ sets 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 $c_s \in C^*$ and for all $s \in G'$, $proj(s, c_s, \sigma_i) = proj(s, \pi''(c_s), \sigma_i)$
 - Written $A, G :| G'$ if p

Example

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

$$\forall (s \in S) \forall (z \in Z) [\{z\}, \{s\} : | S \text{ if } \neg \text{cando}(w, s, z)]$$

- If subject can't execute command (the $\neg \text{cando}$ part) in any state, 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
 - $w_n = v_1, \dots, v_n$ sequence of $v_i \in C^*$
 - $prev(w_n) = w_{n-1}; last(w_n) = v_n$

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

$$\{ z \}, \{ s \} : | S \text{ if } [\neg \text{cando}(\text{prev}(w), s, z) \wedge \\ [\text{cando}(\text{prev}(w), s', \text{pass}(s, z)) \Rightarrow \neg \text{last}(w) = (s', \text{pass}(s, z))]]]$$

Effect

- Suppose $s_1 \in S$ can execute $pass(s_2, z)$
- For all $w \in C^*$, $cando(w, s_1, pass(s_2, z))$ holds
- Initially, $cando(v, s_2, z)$ 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 $s \in S$

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

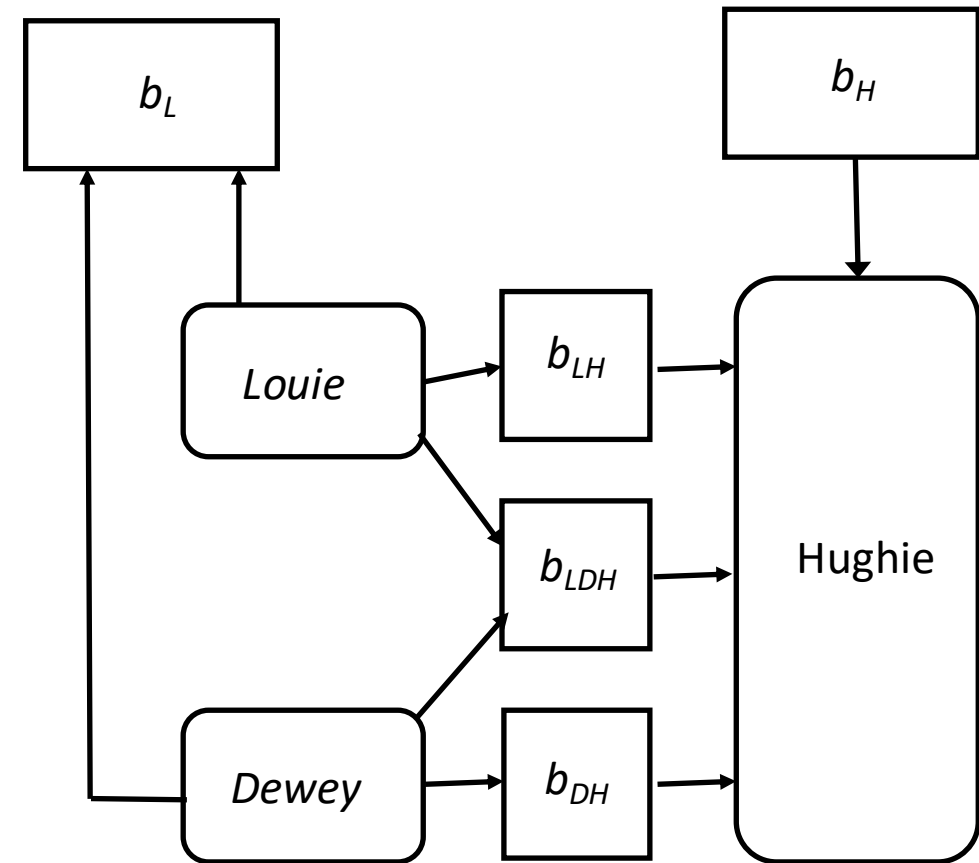
- So s_2 's first execution of z does not affect any subject's observation of system

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

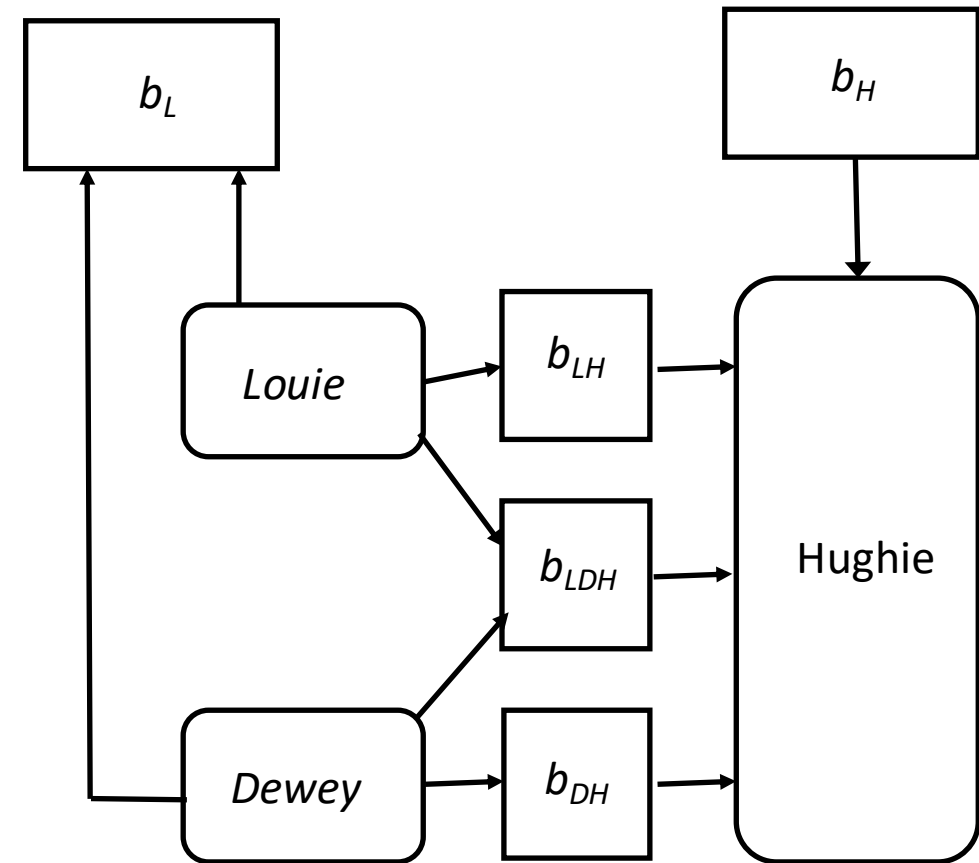
Compose Systems

- 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)



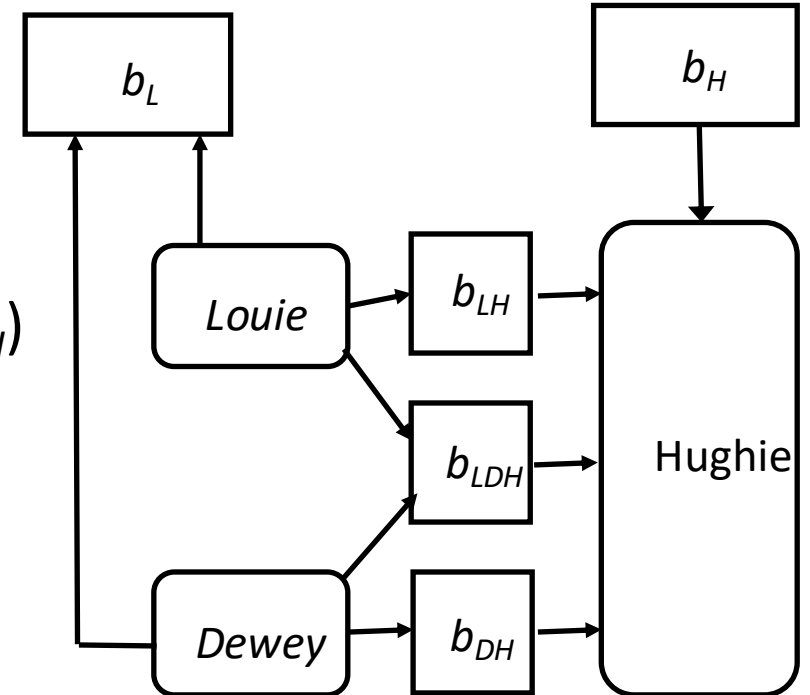
Systems Secure

- 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



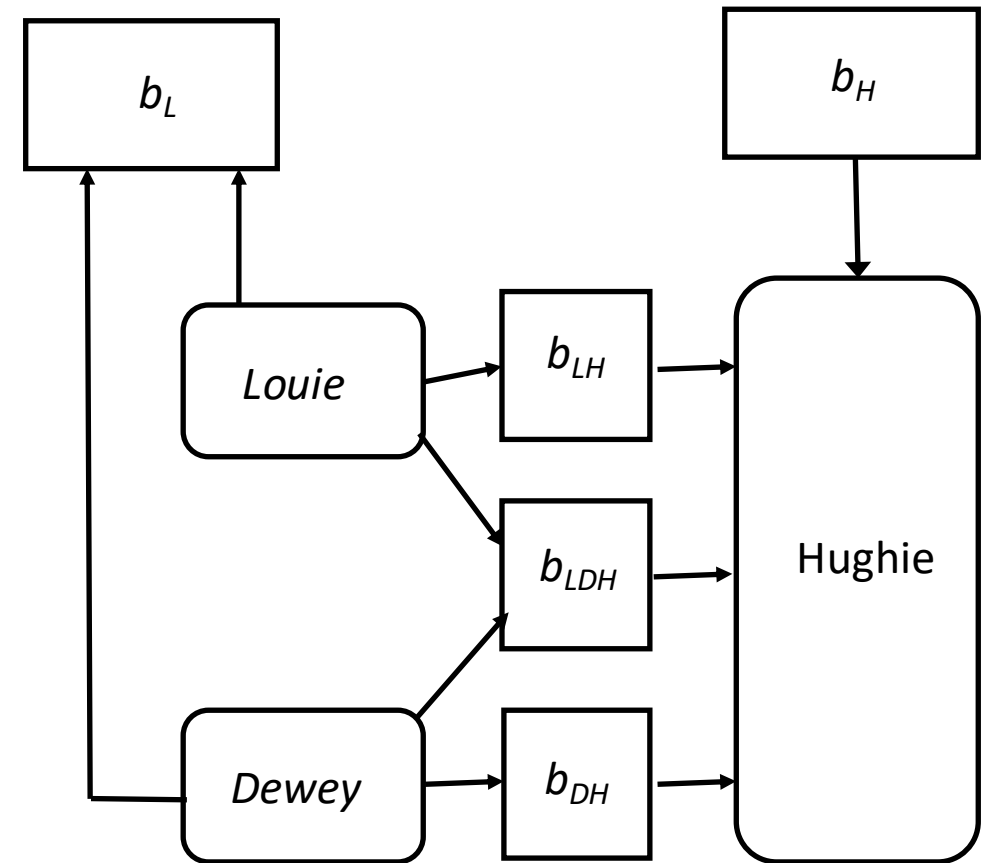
Security of Composition

- 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



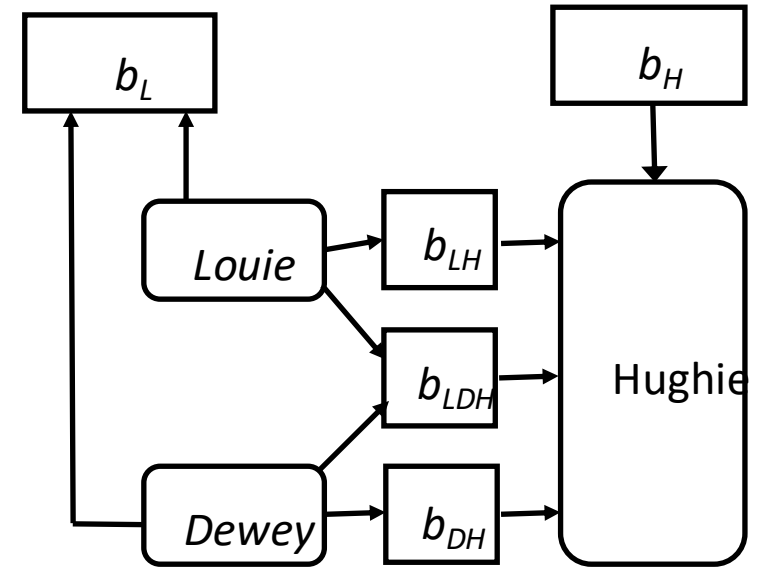
Hughie

- 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 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*
- $\sigma_0 = (0, 0)$
- 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
- $\pi_L: T \rightarrow T_{LOW}$ 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 $t_{LOW} \in T_{LOW}$, 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