ECS 235B, Lecture 25

March 11, 2019
Model

• System as state machine
  • Subjects $S = \{ s_i \}$
  • States $\Sigma = \{ \sigma_i \}$
  • Outputs $O = \{ o_i \}$
  • Commands $Z = \{ z_i \}$
  • State transition commands $C = S \times Z$

• Note: no inputs
  • Encode either as selection of commands or in state transition commands
Functions

• State transition function $T: C \times \Sigma \rightarrow \Sigma$
  • Describes effect of executing command $c$ in state $\sigma$

• Output function $P: C \times \Sigma \rightarrow O$
  • Output of machine when executing command $c$ in state $\sigma$

• Initial state is $\sigma_0$
Example: 2-Bit Machine

• Users Heidi (high), Lucy (low)

• 2 bits of state, $H$ (high) and $L$ (low)
  • System state is $(H, L)$ where $H, L$ are 0, 1

• 2 commands: $xor0$, $xor1$ do xor with 0, 1
  • Operations affect both state bits regardless of whether Heidi or Lucy issues it
Example: 2-bit Machine

• \( S = \{ \text{Heidi, Lucy} \} \)
• \( \Sigma = \{ (0,0), (0,1), (1,0), (1,1) \} \)
• \( C = \{ \text{xor0, xor1} \} \)

<table>
<thead>
<tr>
<th>Input States ((H, L))</th>
<th>xor0</th>
<th>xor1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,0)</td>
<td>(0,0)</td>
<td>(1,1)</td>
</tr>
<tr>
<td>(0,1)</td>
<td>(0,1)</td>
<td>(1,0)</td>
</tr>
<tr>
<td>(1,0)</td>
<td>(1,0)</td>
<td>(0,1)</td>
</tr>
<tr>
<td>(1,1)</td>
<td>(1,1)</td>
<td>(0,0)</td>
</tr>
</tbody>
</table>
Outputs and States

• $T$ is inductive in first argument, as
  
  \[ T(c_0, \sigma_0) = \sigma_1; \ T(c_{i+1}, \sigma_{i+1}) = T(c_{i+1}, T(c_i, \sigma_i)) \]

• Let $C^*$ be set of possible sequences of commands in $C$

• $T^*: C^* \times \Sigma \rightarrow \Sigma$ and
  
  \[ c_s = c_0...c_n \Rightarrow T^*(c_s, \sigma_i) = T(c_n, ..., T(c_0, \sigma_i)...) \]

• $P$ similar; define $P^*: C^* \times \Sigma \rightarrow O$ similarly
Projection

• $T^*(c_s, \sigma_i)$ sequence of state transitions
• $P^*(c_s, \sigma_i)$ corresponding outputs
• $proj(s, c_s, \sigma_i)$ set of outputs in $P^*(c_s, \sigma_i)$ that subject $s$ authorized to see
  • In same order as they occur in $P^*(c_s, \sigma_i)$
  • Projection of outputs for $s$
• Intuition: list of outputs after removing outputs that $s$ cannot see
Purge

- $G \subseteq S$, $G$ a group of subjects
- $A \subseteq Z$, $A$ a set of commands
- $\pi_G(c_s)$ subsequence of $c_s$ with all elements $(s,z)$, $s \in G$ deleted
- $\pi_A(c_s)$ subsequence of $c_s$ with all elements $(s,z)$, $z \in A$ deleted
- $\pi_{G,A}(c_s)$ subsequence of $c_s$ with all elements $(s,z)$, $s \in G$ and $z \in A$ deleted
Example: 2-bit Machine

• Let $\sigma_0 = (0,1)$
• 3 commands applied:
  • Heidi applies $xor0$
  • Lucy applies $xor1$
  • Heidi applies $xor1$
• $c_s = ( (Heidi, xor0), (Lucy, xor1), (Heidi, xor1) )$
• Output is 011001
  • Shorthand for sequence (0,1) (1,0) (0,1)
Example

• $\text{proj}(\text{Heidi}, c_s, \sigma_0) = 011001$
• $\text{proj}(\text{Lucy}, c_s, \sigma_0) = 101$
• $\pi_{\text{Lucy}}(c_s) = (\text{Heidi, xor}0), (\text{Heidi, xor}1)$
• $\pi_{\text{Lucy,xor}1}(c_s) = (\text{Heidi, xor}0), (\text{Heidi, xor}1)$
• $\pi_{\text{Heidi}}(c_s) = (\text{Lucy, xor}1)$
• $\pi_{\text{Lucy,xor}0}(c_s) = (\text{Heidi, xor}0), (\text{Lucy, xor}1), (\text{Heidi, xor}1)$
• $\pi_{\text{Heidi,xor}0}(c_s) = \pi_{\text{xor}0}(c_s) = (\text{Lucy, xor}1), (\text{Heidi, xor}1)$
• $\pi_{\text{Heidi,xor}1}(c_s) = (\text{Heidi, xor}0), (\text{Lucy, xor}1)$
• $\pi_{\text{xor}1}(c_s) = (\text{Heidi, xor}0)$
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 $c_s = (\text{Heidi, xor0}), (\text{Lucy, xor1}), (\text{Heidi, xor1})$ and $\sigma_0 = (0, 1)$
  - As before
- Take $G = \{\text{Heidi}\}, G' = \{\text{Lucy}\}, A = \emptyset$
- $\pi_{\text{Heidi}}(c_s) = (\text{Lucy, xor1})$
  - So $\text{proj}(\text{Lucy, } \pi_{\text{Heidi}}(c_s), \sigma_0) = 0$
- $\text{proj}(\text{Lucy, } c_s, \sigma_0) = 101$
- So $\{\text{Heidi}\} :|\{\text{Lucy}\}$ 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 $0_H0_L1_H$
- $\pi_{Heidi}(c_s) = (Lucy, \text{ xor1})$
  - So $\text{proj}(Lucy, \pi_{Heidi}(c_s), \sigma_0) = 0$
- $\text{proj}(Lucy, c_s, \sigma_0) = 0$
- So $\{Heidi\} :| \{Lucy\}$ 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 $D = \{ d_1, \ldots, d_n \}$
• When command $c$ executed, it is executed in protection domain $\text{dom}(c)$
• Give alternate versions of definitions shown previously
Security Policy

- $D = \{ d_1, ..., d_n \}$, $d_i$ 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
  - $d_i \rightarrow d_j$ means info can flow from $d_i$ to $d_j$
  - $d_i \rightarrow d_i$ as info can flow within a domain
Projection Function

• $\pi'$ analogue of $\pi$, earlier
• Commands, subjects absorbed into protection domains
• $d \in D, c \in C, c_s \in C^*$
• $\pi'_d(\nu) = \nu$
• $\pi'_d(c_s c) = \pi'_d(c_s)c$ if $dom(c) \cap d$
• $\pi'_d(c_s c) = \pi'_d(c_s)$ 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

- $c \in C, \text{dom}(c) \in D$
- $\sim_{\text{dom}(c)}$ equivalence relation on states of system $X$
- $\sim_{\text{dom}(c)}$ output-consistent if
  \[ \sigma_a \sim_{\text{dom}(c)} \sigma_b \Rightarrow P(c, \sigma_a) = P(c, \sigma_b) \]
- Intuition: states are output-consistent if for subjects in $\text{dom}(c)$, 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 \( c \in C \)

• 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 $\text{dom}(c)$ 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 \textit{transition-consistent} under \( r \)
• Intuition: command \( c \) does not affect equivalence of states under 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 ML 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 $\text{dom}(c)$ being noninterfering with $d \in D$ implies $\sigma_a \sim^d T(c, \sigma_a)$
• Intuition: applying $c$ under policy $r$ to system $X$ has no effect on domain $d$ when $X$ locally respects $r$
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 $X$ 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 $\sigma_a \sim^d \sigma_b$ implies

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

• Induct on length of $c_s$

• Basis: $c_s = \nu$, so $T^*(c_s, \sigma_a) = \sigma_a$; $\pi'_d(\nu) = \nu$; claim holds

• Hypothesis: $c_s = c_1 \ldots c_n$; then claim holds
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:
  • $\text{dom}(c_{n+1})rd$ holds
  • $\text{dom}(c_{n+1})rd$ does not hold
dom(\(c_{n+1}\))rd Holds

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

• By definition of \(T^*\) and \(\pi'_d\)

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

• 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

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

• By substitution from earlier equality

\[ T^*(c_sc_{n+1}, \sigma_a) \sim^d T^*(\pi'_d(c_sc_{n+1}), \sigma_b) \]

• 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 \pi'_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 \sigma

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

• **value**: $L \times \Sigma \rightarrow V$
  • returns value $v$ stored in location $l$ when system in state $\sigma$

• **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 $o \in read(d)$
  • $w \in A[s, o]$ if $o \in write(d)$

• Equivalence relation:
  $$[\sigma_a \sim_{dom(c)} \sigma_b] \iff [\forall l_i \in read(d) \ [\ \text{value}(l_i, \sigma_a) = \text{value}(l_i, \sigma_b) \ ]]$$
  • You can read the *exactly* 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

- Output of command \( c \) executed in domain \( \text{dom}(c) \) depends only on values for which subjects in \( \text{dom}(c) \) have read access
  - \( \sigma_a \sim^{\text{dom}(c)} \sigma_b \Rightarrow P(c, \sigma_a) = P(c, \sigma_b) \)
- If \( c \) changes \( l_i \), then \( c \) can only use values of objects in \( \text{read}(\text{dom}(c)) \) to determine new value
  - \( [ \sigma_a \sim^{\text{dom}(c)} \sigma_b \land \) \( (\text{value}(l_i, T(c, \sigma_a)) \neq \text{value}(l_i, \sigma_a) \lor \text{value}(l_i, T(c, \sigma_b)) \neq \text{value}(l_i, \sigma_b)) ] \Rightarrow \) \( \text{value}(l_i, T(c, \sigma_a)) = \text{value}(l_i, T(c, \sigma_b)) \)
- If \( c \) changes \( l_i \), then \( \text{dom}(c) \) provides subject executing \( c \) with write access to \( l_i \)
  - \( \text{value}(l_i, T(c, \sigma_a)) \neq \text{value}(l_i, \sigma_a) \Rightarrow l_i \in \text{write}(\text{dom}(c)) \)
Enforcing Policies \( r \): Specific to Policy

• 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 \( v \)

\[
\begin{align*}
\left[ u, v \in D \land urv \right] & \Rightarrow \text{read}(u) \subseteq \text{read}(v) \\
\end{align*}
\]

• Subject \( s \) can write object \( o \) in \( v \), subject \( s' \) can read \( o \) in \( u \), then domain \( v \) can interfere with domain \( u \)

\[
\begin{align*}
\left[ l_i \in \text{read}(u) \land l_i \in \text{write}(v) \right] & \Rightarrow vru
\end{align*}
\]
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 $(\text{dom}(c), d) \notin r$ but $\sigma_a \sim^d T(c, \sigma_a)$ does not hold
- Some object has value changed by $c$:
  \[ \exists l_i \in \text{read}(d) \mid \text{value}(l_i, \sigma_a) \neq \text{value}(l_i, T(c, \sigma_a)) \]
- Condition 3: $l_i \in \text{write}(d)$
- Condition 5: $\text{dom}(c) \not\in \text{rd}$, contradiction
- 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 $\text{value}(l_i, T(c, \sigma_a)) = \text{value}(l_i, T(c, \sigma_b))$ for $l_i \in \text{read}(d)$

• 3 cases dealing with change that $c$ makes in $l_i$ in states $\sigma_a$, $\sigma_b$
  • $\text{value}(l_i, T(c, \sigma_a)) \neq \text{value}(l_i, \sigma_a)$
  • $\text{value}(l_i, T(c, \sigma_b)) \neq \text{value}(l_i, \sigma_b)$
  • Neither of the above two hold
Case 1: $\text{value}(l_i, T(c, \sigma_a)) \neq \text{value}(l_i, \sigma_a)$

- **Condition 3:** $l_i \in \text{write}(\text{dom}(c))$
- **As** $l_i \in \text{read}(d)$, **condition 5** says $\text{dom}(c)\text{rd}$
- **Condition 4:** $\text{read}(\text{dom}(c)) \subseteq \text{read}(d)$
  - **As** $\sigma_a \sim^d \sigma_b$, $\sigma_a \sim^{\text{dom}(c)} \sigma_b$
- **Condition 2:** $\text{value}(l_i, T(c, \sigma_a)) = \text{value}(l_i, T(c, \sigma_b))$
- **So** $T(c, \sigma_a) \sim^{\text{dom}(c)} T(c, \sigma_b)$, as desired
Case 2: \( \text{value}(l_i, T(c, \sigma_b)) \neq \text{value}(l_i, \sigma_b) \)

- Condition 3: \( l_i \in \text{write}(\text{dom}(c)) \)
- As \( l_i \in \text{read}(d) \), condition 5 says \( \text{dom}(c) \cap \text{read}(d) \)
- Condition 4: \( \text{read}(\text{dom}(c)) \subseteq \text{read}(d) \)
- As \( \sigma_a \sim^d \sigma_b, \sigma_a \sim^{\text{dom}(c)} \sigma_b \)
- Condition 2: \( \text{value}(l_i, T(c, \sigma_a)) = \text{value}(l_i, T(c, \sigma_b)) \)
- So \( T(c, \sigma_a) \sim^{\text{dom}(c)} T(c, \sigma_b) \), 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:
  \[
  \text{for } l_i \in \text{read}(d), \ value(l_i, \sigma_a) = value(l_i, \sigma_b)
  \]

• 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
  • $\text{cando}(w, s, z)$ holds if $s$ can execute $z$ in current state
  • Condition noninterference on $\text{cando}$
  • If $\neg\text{cando}(w, \text{Lara, “write } f\text{”})$, 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 $C^*$
- $c_s = (c_1, \ldots, c_n) \in C^*$
- $\pi''(v) = v$
- $\pi'''((c_1, \ldots, c_n)) = (c_1', \ldots, c_n')$, where
  - $c_i' = v$ 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 $\nu$
  • 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 $s \in G'$, $\text{proj}(s, c_s, \sigma_i) = \text{proj}(s, \pi''(c_s), \sigma_i)$
  - Written $A, G : | G'$ if $p$
Example

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

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

• If subject can’t execute command (the \(\neg \textit{cando}\) 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
  • $\text{pass}(s, z)$ gives $s$ right to execute $z$
  • $w_n = v_1, ..., v_n$ sequence of $v_i \in C^*$
  • $\text{prev}(w_n) = w_{n-1}$; $\text{last}(w_n) = v_n$
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 $s$

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

• Suppose $s_1 \in S$ can execute $\text{pass}(s_2, z)$
• For all $w \in C^*$, $\text{cando}(w, s_1, \text{pass}(s_2, z))$ holds
• Initially, $\text{cando}(v, s_2, z)$ false
• Let $z' \in Z$ be such that $(s_3, z')$ noninterfering with $(s_2, z)$
  • So for each $w_n$ with $v_n = (s_3, z')$, $\text{cando}(w_n, s_2, z) = \text{cando}(w_{n-1}, s_2, z)$
Effect

• Then policy says for all $s \in S$

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

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

• So $s_2$’s first execution of $z$ does not affect any subject’s observation of system