What is this talk about?

- What is the right setting for considering information flow security for interactive programs?
- We consider a recent paper that proposes non-deducibility on strategies in a language-based setting.
- We investigate some variations and restrictions of the definitions.
- The aim is to distinguish what is essential from what is not.
Related work 1

- “Information flow in non-deterministic systems” Wittbold & Johnson [Security and Privacy 1990]
  - defined Non-Deducibility on Strategies: prohibits active channels as well as passive ones
  - synchronous statemachines setting
  - provided an example (originally due to Shannon)
    - satisfies non-deducibility on input strings but not non-deducibility on strategies
Related work 2

- “A logical approach to multilevel security of probabilistic systems” Gray & Syverson [Distributed Computing 1998]
- “Information flow security for interactive programs” O’Neill, Clarkson, & Chong [CSFW 2006]
  - “Non-Interference” properties for interactive programming languages
  - based on non-deducibility on strategies
  - setting is simple imperative language extended with non-deterministic and probabilistic choice
Framework

- A simple *while* language with input and output primitives
- A set of security types, \( \mathcal{L} \) ordered by \( \leq \) with \( \tau \in \mathcal{L} \).
  (we assume \( \mathcal{L} \) is the two point lattice \( L \leq H \) for this talk)
- An *event* is the transmission of an input to or an output from the program:
  - \( \text{Ev}(\tau) \triangleq \{ \text{in}(\tau, v), \text{out}(\tau, v) \mid v \in \mathbb{Z} \} \)
  - \( \text{Ev} \triangleq \bigcup_{\tau \in \mathcal{L}} \text{Ev}(\tau) \)
- A *trace*, \( t \), is a finite list of events
Operational semantics (selected rules)

[Assign] \[
\begin{align*}
(x := e, \sigma, t, \omega) & \rightarrow (\text{skip}, \sigma[x := \sigma(e)], t, \omega)
\end{align*}
\]

[In] \[
\begin{align*}
\omega_\tau(t \upharpoonright \tau) = v \\
(\text{input } x \text{ from } \tau, \sigma, t, \omega) & \rightarrow (\text{skip}, \sigma[x := v], t^{\wedge}\langle \text{in}(\tau, v)\rangle, \omega)
\end{align*}
\]

[Out] \[
\begin{align*}
\sigma(e) = v \\
(\text{output } e \text{ to } \tau, \sigma, t, \omega) & \rightarrow (\text{skip}, \sigma, t^{\wedge}\langle \text{out}(\tau, v)\rangle, \omega)
\end{align*}
\]

• \(\omega\) is a pair of strategies \(\omega_L, \omega_H\)
• \(t \upharpoonright \tau\) denotes \(t\) with all non-\(\tau\) events removed
while (true) do
    x := 0 | x := 1;
    output x to H;
    input y from H;
    output (x xor y) to L;
i_H(y) abbreviates in(H, y), o_H(x) abbreviates out(H, x), etc

\begin{align*}
\omega(\langle o_H(x) \rangle) &= x \text{ xor } 1 \\
\omega(\langle o_H(x) i_H(y) o_L(z) o_H(x') \rangle) &= x' \text{ xor } 0 \\
\omega(\langle o_H(x) i_H(y) o_L(z) o_H(x') i_H(y') o_L(z') o_H(x'') \rangle) &= x'' \text{ xor } 1
\end{align*}

Transmits 101 to Low, even though Low has no idea what High actually inputs.
Non-deducibility on strategies

- Security condition is an instance of *non-deducibility on strategies* as defined by Wittbold & Johnson
- A program is secure if, for every initial state, $\sigma$, any trace of events seen on channel $L$ is consistent with every possible user strategy for channel $H$.
- Shannon example clearly does *not* have this property
Non-deducibility on strategies

- For $m = (c, \sigma, \langle \rangle, \omega)$ we write $m \leadsto t$ to mean
  $\exists c', \sigma'. (c, \sigma, \langle \rangle, \omega) \xrightarrow{\ast} (c', \sigma', t, \omega)$
- we write $t =_{\tau} t'$ to mean $t |_{\tau} = t' |_{\tau}$
- [OCC 2006] A command $c$ satisfies non-deducibility on strategies iff for all $m = (c, \sigma, \langle \rangle, \omega)$ and $m' = (c, \sigma, \langle \rangle, \omega')$:
  
  if $\omega_L = \omega'_L$ and $m \leadsto t$
  then $\exists t'. t =_{L} t'$ and $m' \leadsto t'$
What is a strategy?

- A mapping from traces to values (inputs)
- OCC define $\omega_H$ as a function on traces of $H$-events and $\omega_L$ as a function on traces of $L$-events
- What happens if:
  - we allow High to see $L$-events?
  - we allow strategies to be non-deterministic? probabilistic?
  - we restrict strategies even further (streams)?
Pure High Strategies
Deterministic Strategies

Pure High Strategies
Unrestricted deterministic strategies

- Slight reformulation of the semantics
- Strategies get applied to the full trace so far:

\[\text{In} \quad \frac{\omega_\tau(t) = v}{(\text{input } x \text{ from } \tau, \sigma, t, \omega) \rightarrow (\text{skip}, \sigma[x := v], t \langle \text{in}(\tau, v) \rangle, \omega)}\]
Low strategies and Pure High strategies

- Low strategies can see only $L$-events:
  \[ t =_L t' \Rightarrow \omega_L(t) = \omega_L(t') \]

- A High strategy is pure if it can see only $H$-events:
  \[ t =_H t' \Rightarrow \omega_H(t) = \omega_H(t') \]
Streams as strategies

- Define $t \xRightarrow{\tau} t'$ iff $t$ and $t'$ contain the same number of $\tau$-input events.
- A strategy $\omega$ is a $\tau$-stream iff:

$$t \xRightarrow{\tau} t' \Rightarrow \omega(t) = \omega(t')$$
Streams Suffice for Deterministic Programs

Proposition:

For deterministic programs it makes no difference whether we define Non-Deducibility with respect to arbitrary non-deterministic High strategies, arbitrary deterministic High strategies, Pure High strategies or Streams.
Proof Sketch (Part 1)

A command \( c \) fails non-deducibility on strategies iff there exist some \( \sigma, \omega_L, \omega_H, \omega'_H \) such that:

\[
(c, \sigma, \langle \rangle, (\omega_H, \omega_L)) \rightsquigarrow t \\
\text{but } \forall t'. (c, \sigma, \langle \rangle, (\omega'_H, \omega_L)) \rightsquigarrow t' \text{ implies } t \neq_L t'
\]

- Given such a counterexample with deterministic strategies, construct stream-strategies \( \hat{\omega}_L, \hat{\omega}_H, \hat{\omega}'_H \) such that:
  - \( (c, \sigma, \langle \rangle, (\hat{\omega}_H, \hat{\omega}_L)) \rightsquigarrow t \)
  - if \( (c, \sigma, \langle \rangle, (\hat{\omega}'_H, \hat{\omega}_L)) \rightsquigarrow t' \) then \( t \neq_L t' \)
- Non-deterministic case dealt with separately (easy)
Proof Sketch (Part 2)

\[ \hat{\omega}_H(t) = \begin{cases} v & \text{if } \exists t' = S^H_H t. (c, \sigma, \langle \cdot \rangle, (\omega_H, \omega_L)) \leadsto t' \langle \text{in}(H, v) \rangle \\ 0 & \text{otherwise} \end{cases} \]

\[ \hat{\omega}'_H(t) = \begin{cases} v & \text{if } \exists t' = S^H_H t. (c, \sigma, \langle \cdot \rangle, (\omega'_H, \omega_L)) \leadsto t' \langle \text{in}(H, v) \rangle \\ 0 & \text{otherwise} \end{cases} \]

\[ \hat{\omega}_L(t) = \begin{cases} v & \text{if } \exists t' = S^L_L t. (c, \sigma, \langle \cdot \rangle, (\omega_H, \omega_L)) \leadsto t' \langle \text{in}(L, v) \rangle \\ 0 & \text{otherwise} \end{cases} \]

We can show:

- well-defined (for deterministic \(c\))
- behave as required for the proof
Future Work

- Deterministic programs + probabilistic strategies?
- Non-deterministic programs:
  - Shannon example shows that streams do not suffice
  - OCC consider Pure High strategies
  - What happens if we consider arbitrary High strategies? probabilistic High strategies?
  - Can we always transform out the non-determinism by adding additional parameters?