The Security $\pi$-calculus and Non-interference

M. Hennessy, University of Sussex

- Background
- The Security $\pi$-calculus
- Types
- Behavioural Equivalences
- Non-Interference Results

Work in progress by EU Global Computing projects Mikado/Myths
Background

• Control of information flow in systems
cf. Denning, Goguen, Mesegeur

• **Integrity**: No High-to-Low information flow: High (security) level users should not be able to send high-level information to Low level users. (No Trojan horses)

• **Non-interference**: Formal property of systems which ensures their integrity.
High-to-Low Information Flow

**Explicit:** \( H \) sends high-level data \((\text{my visa no})\) to \( L \)

**Implicit:** \( H \) sends low-level data to \( L \)
\( H, L \) could have prearranged interpretation:
- 0 - Boss is in town
- 1 - Boss is away

**Implicit:** \( H \) may rendez-vous with \( L \)
- \( H \) turns up - Boss is away
- \( H \) absent - Boss is in town
How to Avoid H-to-L Information Flow

H can not send any data to L

Q?: What kind of data can L send to H?
Q?: How can rendez-vous’s be managed?

More General Q?: How can we SPECIFY behaviour of system which will ensure no H-to-L information flow?

ANSWER: Codify using Types cf. Volpano et al.

A system is safe if it can be typechecked
Safe Systems

• How do we prove safe systems contain no H-to-L information flow?
• Introduce **Interference-Freeness**: Formal verifiable concept, which informally implies no H-to-L information flow
• **Main Theorem**: \( \text{S is typeable (using my type system) implies S is interference-free} \)
Interference-Freeness

Requirements:

- concept of High-level process (specified using Type system)
- concept of behavioural equivalence $\simeq_\sigma$, relativised to security levels $\sigma$, ($= \text{bot}, \ldots, \text{top}$)

Definition: $S$ is Interference-Free if

$$S \mid H \simeq_{\text{bot}} S \mid K$$

for all High-level processes $H, K$. 
Remainder of Talk

- Language: $\pi$-calculus
- Types: input/output types, relativised to security levels
- Behavioural Equivalences $\simeq_\sigma$: based on testing
The Security $\pi$-calculus (asynchronous)

channels = resources = read once variables

- $u?(x : T) \ P$ - patterned input on channel $u$ to resource $u$
- $u!(v)$ - polyadic output on channel $u$ from resource $u$
  
  $v$ a tuple of values - may be channels

- $P \mid Q$ - concurrent code
- if $u = v$ then $P$ else $Q$ - value testing
- $(\text{new } n : T) \ P$ - generation of new names
- $\ast P$, $\mathbf{0}$ - iteration and termination
Reduction Semantics

Same as ever (for $\pi$ hackers):

(figsize) $a!\langle v \rangle \mid a?(x : A) P \leftrightarrow (P[v/x])$

(figsize) \[
\dfrac{P \equiv Q, \ P \leftrightarrow P', \ P' \equiv Q'}{
Q \leftrightarrow Q'}
\]  

(figsize) \[
\dfrac{P \leftrightarrow P'}{
P \mid Q \leftrightarrow P' \mid Q}
\]

(figsize) (etc.)
Reduction Semantics

Dynamic creation of communication links:

Interface between H and L processes must be managed.
Security Levels

A (static) complete lattice of security levels, $SL$.

- **bot**: lowest security level
  - the great unwashed
  - processes arriving off the web
  - processes at this level offer no security

- **top**: highest security level
  - the chosen few
  - processes owned by superuser on local machine

- **bot $\leq$ moderate $\leq$ top**:  
  - processes originating on local area network
  - processes which have demonstrated some reliability

$SL$ may have an arbitrary complicated structure.
Types - graded read/write capabilities

\(Type_\sigma\): Type for values accessible at security level \(\sigma\)

\(\{w_\sigma\langle A\rangle, r_{\rho_1}\langle B_1\rangle, r_{\rho_2}\langle B_2\rangle, \ldots r_{\rho_k}\langle B_k\rangle\}\)

provided

- \(\sigma \leq \rho_i\) - no write ups
- \(A \in Type_\sigma, B_i \in Type_{\rho_i}\)
- \(A\) subtype \(B_i\)

Example:

- Yes: \(\{w_{\text{bot}}\langle \text{int}\rangle, r_{\text{bot}}\langle \text{int}\rangle, r_{\text{top}}\langle \text{int}\rangle\}\) - multi-level type
- No: \(\{w_{\text{top}}\langle \text{int}\rangle, r_{\text{bot}}\langle \text{int}\rangle, r_{\text{top}}\langle \text{int}\rangle\}\)
- No: \(\{w_{\text{bot}}\langle w_{\text{bot}}\langle \text{int}\rangle\rangle, r_{\text{bot}}\langle \ldots \rangle, r_{\text{top}}\langle w_{\text{bot}}\langle \text{int}\rangle\rangle\}\)
Type Environment \( \Gamma = u_1 : A_1, \ldots , u_k : A_k \)

- \( \Gamma \vdash P \) - \( P \) well-typed wrt \( \Gamma \), ignoring security levels
- \( \Gamma \vdash_\sigma P \) - \( P \) well-typed, using \textit{at most} level \( \sigma \) resources
- \( \Gamma \vdash^\sigma P \) - \( P \) well-typed, using \textit{at least} level \( \sigma \) resources
- \( \Gamma \vdash_{r\sigma} P \) - \ldots , \textit{reading} from \textit{at most} level \( \sigma \) resources
- \( \Gamma \vdash^{w\sigma} P \) - \ldots , \textit{writing} to \textit{at least} level \( \sigma \)

**Thm: Subject Reduction:** \( \Delta \vdash P \) and \( P \leadsto^* Q \) implies \( \Delta \vdash Q \)
Type Inference

\[(\text{LT-IN})\]
\[
\Gamma, X : A \vdash_{\sigma} P \\
\Gamma \vdash u : r_\delta \langle A \rangle \\
\frac{\Gamma \vdash u ? (X : A) P}{\Gamma \vdash_{\sigma} u ? (X : A) P} \delta \preceq \sigma
\]

\[(\text{LT-OUT})\]
\[
\Gamma \vdash \nu : A \\
\Gamma \vdash u : w_\delta \langle A \rangle \\
\frac{\Gamma \vdash \nu ! \langle \nu \rangle}{\Gamma \vdash_{\sigma} \nu ! \langle \nu \rangle} \delta \preceq \sigma
\]

\[(\text{T-EQ})\]
\[
\Gamma \vdash \nu : A, \nu : B \\
\Gamma \vdash Q \\
\Gamma \sqcap \{ \nu : B, \nu : A \} \vdash P \\
\frac{\Gamma \vdash \text{if } \nu = v \text{ then } P \text{ else } Q}{\Gamma \vdash P}
\]

\[(\text{T-NEW})\]
\[
\Gamma, a : A \vdash P \\
\frac{\Gamma \vdash \text{(new } a : A \text{) } P}{\Gamma \vdash P}
\]
Examples

\[
\begin{align*}
\boxed{H} & \leftarrow \text{lh}?(x) \ x!\langle 3pm \rangle \\
\boxed{L} & \leftarrow \text{lh}!\langle cvt \rangle \ \text{cvt}?(i) \ \text{broadcast}(i)
\end{align*}
\]

If \( \text{lh} \) is \( w_{\text{bot}}\langle \ldots \rangle \), \( r_{\text{top}}\langle \ldots \rangle \) then \( \Gamma \not\vdash L \mid H \)

\( L \mid H \) contains information flow

\[
\text{TrH} \leftarrow \text{h}?(x) \ . \text{if} \ x = \text{boss} \ \text{then} \ tr_1!\langle \rangle \ \text{else} \ tr_2!\langle \rangle
\]

If \( \text{h} \) high, \( tr_i \) low, then \( \text{TrH} \) can not be \( \text{High-level} \)

\( \text{TrH} \) represents a trojan horse
**Safe Systems** at last

**Definition:** \( S \) is \( \Gamma \)-safe if \( \Gamma \vdash_{\text{bot}} S \)

They can only read from low-level channels

**Claim:** If \( S \) is \( \Gamma \)-safe then

\[
\begin{array}{l}
S | H \sim_{\text{bot}} S | K
\end{array}
\]

for all High-level processes \( H, K \).

**Definition:** \( H \) is a High-level process if \( \Gamma \vdash_{\text{top}} H \)

They can only write to high-level channels
Behavioural Equivalences

Idea: \( S \simeq_{\sigma} U \) at level \( \sigma \), if no observer running at level at most \( \sigma \) can not distinguish between \( S \) and \( U \).

- An observation of \( S \) by \( O \) is a sequence
  \[ O \mid S \mapsto O_1 \mid S_1 \ldots \mapsto O_n \mid S_n \mapsto \ldots \]
- Successful if some \( O_k \), can report success
- \( S \) may \( O \) if there is some successful observation of \( S \) by \( O \)
- \( S \) must \( O \) if every observation of \( S \) by \( O \) is successful

Definition: \( \Gamma \triangleright_{\sigma} S \simeq_{may} U \) if for every \( \Gamma \vdash_{\sigma} O \),
\( S \) may \( O \) if and only if \( U \) may \( O \)

\( \Gamma \triangleright_{\sigma} S \simeq_{must} U \) if . . . . . .
Non-Interference

Idea: $S$ is interference-free if low-level observers/users cannot detect the presence/absence of high-level users in $S$.

Definition: $S$ is mayIntFree if

$$
\Gamma \triangleright_{\text{bot}} S \mid H \simeq_{\text{may}} S \mid K
$$

for all High-level process $H$, $K$

NonInterference for Free:

Thm: If $S$ is $\Gamma$-safe ($\Gamma \vdash_{\text{rbot}} S$) then $S$ is mayIntFree
Examples

Assume $H, K$ high-level ($\Gamma \vdash_{\text{wtop}} H, K$)

$S$ safe ($\Gamma \vdash_{\text{rbot}} S$)

\[
\begin{align*}
\text{H} &= h?(x) \text{ if } x = \text{boss} \text{ then } tr_1!\langle \rangle \text{ else } tr_2!\langle \rangle \\
\text{K} &= h?(x) \text{ tr}_1!\langle \rangle 
\end{align*}
\]

$\Gamma \triangleright_{\text{bot}} S \mid H \simeq_{\text{may}} S \mid K$ because write on $tr_i$ must be high

\[
\begin{align*}
\text{H} &= h?(x) \text{ if } x = \text{boss} \text{ then } tr_1?() \text{ else } tr_2?() \\
\text{K} &= h?(x) \text{ tr}_1?()
\end{align*}
\]

$\Gamma \triangleright_{\text{bot}} S \mid H \simeq_{\text{may}} S \mid K$ because communication is asynchronous
Example: Multi-level types

\[ \Gamma \text{ maps } \text{ml} \text{ to } \{w_{\text{bot}}(\ldots), r_{\text{bot}}(\ldots), r_{\text{top}}(\ldots)\} \text{ multi-level type} \]

\[ \boxed{S = \text{ml}(a) | \text{ml}(x) \ x!()} \]

\[ \Gamma \vdash_{\text{bot}} S | H \simeq_{\text{may}} S | K \text{ because } S \text{ is safe} \]

\text{BUT: } \Gamma \vdash_{\text{bot}} S | H \not\simeq_{\text{must}} S | K

eg with \( H = \textbf{0} \) and \( K = \text{ml}(x : \text{B}) \textbf{0} \)

observer \( a?(\) \( \omega!() \) sees a difference

**Thm:** Suppose \( \Gamma \) uses only single-level types.

If \( S \) is \( \Gamma \)-safe then it is \textbf{mustIntFree}
Wrap up

**Thesis:** Potential H-to-L information flow in concurrent systems can be detected by type systems

**Questions:**
- How difficult is type inference?
- How restrictive is the type system?
- Can types be extended to distributed systems?

**Technical Details:** Sussex technical reports