Using Architecture to Reason about Information Security

Ron van der Meyden

**(University of New South Wales
Sydney, Australia)**

**Joint work with
Stephen Chong
(Harvard)**

December 3, 2012

Overview

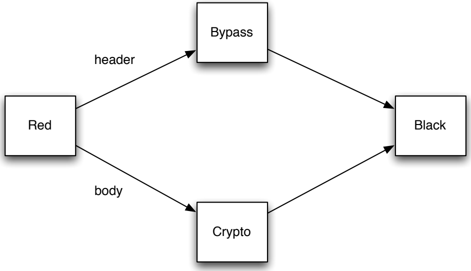
- Motivation: MILS Security
- Recap of Intransitive noninterference theory
- Extended theory for architectural specifications
- Using architecture to reason about information flow properties
- Connections to Access Control

Introduction: Rushby view of MILS

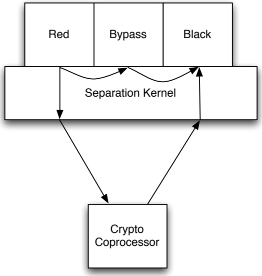
A two level design process comprised of

- **Policy Level:** an architectural design identifying components and their connections/permitted causal relationships.
- **Resource Sharing Level:** components implemented so as to share resources (processors, memory, network) with enforcement of architectural causality constraints using a variety of mechanisms (e.g., separation kernels, periods processing, crypto)

Policy Level:



Resource Sharing Level:



Further Objectives for MILS

- isolation of safety/security critical functionality in (small, formally verifiable) *trusted components*
- (formal) compositional derivation of global properties from architecture + local properties of the trusted components
- These global properties preserved by the resource sharing implementation

Overview of this talk

Questions concerning this vision:

- What is the formal syntax and semantics of architectural designs?

- Is it really possible to prove interesting security properties in this architecture + trusted component, local to global, pattern?

- How does the theory ground out in concrete resource sharing mechanisms?

Overview of this talk

Questions concerning this vision: (& our answers, in this talk)

- What is the formal syntax and semantics of architectural designs?
  - Abstract syntax based on an extension of intransitive noninterference policies
  - A new semantics based on a knowledge-based approach to intransitive noninterference of van der Meyden (ESORICS 2007)
- Is it really possible to prove interesting security properties in this architecture + trusted component, local to global, pattern?
- How does the theory ground out in concrete resource sharing mechanisms?

Overview of this talk

Questions concerning this vision: (& our answers, in this talk)

- What is the formal syntax and semantics of architectural designs?
  - Abstract syntax based on an extension of intransitive noninterference policies
  - A new semantics based on a knowledge-based approach to intransitive noninterference of van der Meyden (ESORICS 2007)
- Is it really possible to prove interesting security properties in this architecture + trusted component, local to global, pattern?
  - Examples indicating the answer is 'Yes'.
- How does the theory ground out in concrete resource sharing mechanisms?

Overview of this talk

Questions concerning this vision: (& our answers, in this talk)

- What is the formal syntax and semantics of architectural designs?
  - Abstract syntax based on an extension of intransitive noninterference policies
  - A new semantics based on a knowledge-based approach to intransitive noninterference of van der Meyden (ESORICS 2007)
- Is it really possible to prove interesting security properties in this architecture + trusted component, local to global, pattern?
  - Examples indicating the answer is 'Yes'.
- How does the theory ground out in concrete resource sharing mechanisms?
  - Sufficient condition for architectural compliance in access control systems.

Noninterference policies (Goguen and Meseguer 1982)

Let D be a set of security domains/components/agents.

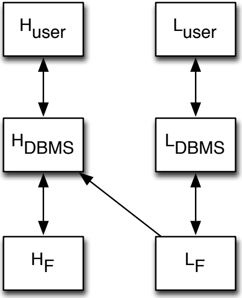
A noninterference policy is a reflexive relation $\rightarrowtail \subseteq D \times D$

$u \rightarrowtail v$ means

> "actions of u are permitted to interfere with v", or
>
> "actions of u are permitted to have effects observable to v", or
>
> "information is permitted to flow from u to v"

Hinke-Schaeffer Architecture

One of the proposed architectures for multi-level secure databases, as an intransitive noninterference policy:



(Strictly speaking, Hinke-Schaeffer = this policy level architecture, enforced at resource sharing level by the operating system.)

Deterministic System Model

Machines have the form $M = \langle S, s_0, Actions, D, \mathtt{dom}, \mathtt{step}, O \rangle$ where

- S is a set of states,

Deterministic System Model

Machines have the form $M = \langle S, s_0, Actions, D, \texttt{dom}, \texttt{step}, O \rangle$ where

- S is a set of states,
- $s_0 \in S$ is the *initial state*,

Deterministic System Model

Machines have the form $M = \langle S, s_0, Actions, D, \mathtt{dom}, \mathtt{step}, O \rangle$ where

- S is a set of states,
- $s_0 \in S$ is the *initial state*,
- $Actions$ is a set of actions,

Deterministic System Model

Machines have the form $M = \langle S, s_0, Actions, D, \mathtt{dom}, \mathtt{step}, O \rangle$ where

- S is a set of states,
- $s_0 \in S$ is the *initial state*,
- $Actions$ is a set of actions,
- D is a set of *domains*

Deterministic System Model

Machines have the form $M = \langle S, s_0, Actions, D, \texttt{dom}, \texttt{step}, O \rangle$ where

- S is a set of states,
- $s_0 \in S$ is the *initial state*,
- *Actions* is a set of actions,
- D is a set of *domains*
- $\texttt{dom} : Actions \longrightarrow D$ associates each action to a domain in D,

Deterministic System Model

Machines have the form $M = \langle S, s_0, Actions, D, \mathtt{dom}, \mathtt{step}, O \rangle$ where

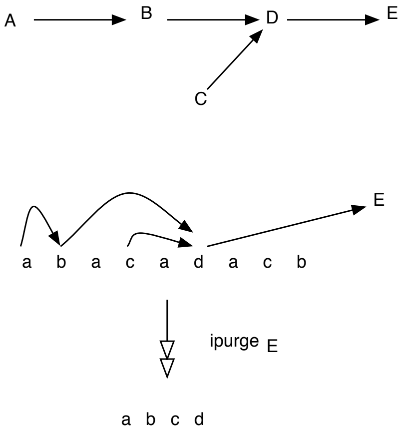
- S is a set of states,
- $s_0 \in S$ is the *initial state*,
- *Actions* is a set of actions,
- D is a set of *domains*
- $\mathtt{dom} : Actions \longrightarrow D$ associates each action to a domain in D,
- $\mathtt{step} : S \times Actions \longrightarrow S$ is a deterministic transition function, and

Deterministic System Model

Machines have the form $M = \langle S, s_0, Actions, D, \mathtt{dom}, \mathtt{step}, O \rangle$ where

- S is a set of states,
- $s_0 \in S$ is the *initial state*,
- $Actions$ is a set of actions,
- D is a set of *domains*
- $\mathtt{dom} : Actions \longrightarrow D$ associates each action to a domain in D,
- $\mathtt{step} : S \times Actions \longrightarrow S$ is a deterministic transition function, and
- $O_u : S \longrightarrow Obs$ — the observation of/output to domain $u \in D$ at a state

Haigh and Young's Intransitive Purge Function



Haigh and Young's definition: IP-security

A system M is IP-secure with respect to a (possibly intransitive) policy $\rightarrowtail$ if for all $u \in D$ and all sequences $\alpha, \alpha' \in A^*$ with $\mathtt{ipurge}_u(\alpha) = \mathtt{ipurge}_u(\alpha')$, we have $O_u(s_0 \cdot \alpha) = O_u(s_0 \cdot \alpha')$.

Alternate definition: van der Meyden - ESORICS'07

Given a policy $\rightarrowtail$, define, for each domain $u \in D$, the function $\mathtt{ta}_u$, with domain *Actions**, inductively by $\mathtt{ta}_u(\epsilon) = \epsilon$, and, for $\alpha \in \mathit{Actions}^*$ and $a \in \mathit{Actions}$,

$$\mathtt{ta}_u(\alpha a) = \begin{cases} \mathtt{ta}_u(\alpha) & \text{if } \mathtt{dom}(a) \not\rightarrowtail u \\ (\mathtt{ta}_u(\alpha), \mathtt{ta}_{\mathtt{dom}(a)}(\alpha), a) & \text{if } \mathtt{dom}(a) \rightarrowtail u \end{cases}$$

Alternate definition: van der Meyden - ESORICS'07

Given a policy $\rightarrowtail$, define, for each domain $u \in D$, the function $\mathtt{ta}_u$, with domain $Actions^*$, inductively by $\mathtt{ta}_u(\epsilon) = \epsilon$, and, for $\alpha \in Actions^*$ and $a \in Actions$,

$$\mathtt{ta}_u(\alpha a) = \begin{cases} \mathtt{ta}_u(\alpha) & \text{if } \mathtt{dom}(a) \not\rightarrowtail u \\ (\mathtt{ta}_u(\alpha), \mathtt{ta}_{\mathtt{dom}(a)}(\alpha), a) & \text{if } \mathtt{dom}(a) \rightarrowtail u \end{cases}$$

Define a system M to be TA-secure with respect to a policy $\rightarrowtail$ if for all domains $u \in D$, and all $\alpha, \alpha' \in Actions^*$ such that $\mathtt{ta}_u(\alpha) = \mathtt{ta}_u(\alpha')$, we have $O_u(s_0 \cdot \alpha) = O_u(s_0 \cdot \alpha')$.

Reasons to believe this definition is better

Results from ESORICS-07:

- It does not admit a disturbing example from ESORICS-07
- TA-security $\Rightarrow$ IP-security
- TA-security $\equiv$ IP-security for transitive policies
- Rushby unwinding conditions for IP-security $\Rightarrow$ TA-security
- A system bisimilar to M satisfies Rushby unwinding conditions $\Rightarrow$ M is TA-secure
- (A similar equivalence for a variant of Rushby's access control results.)

AND: It leads to the generalization of the present paper ...

Reasoning about Knowledge

Given a system M, define the view of domain u of a sequence $\alpha \in \mathit{Actions}^*$ to be the sequence $\mathtt{view}_u(\alpha)$ of all actions and observations of that domain, with stuttering of observations eliminated (to model asynchrony).

E.g. if $\alpha = hhlh$ generates (Low observations only):

$O_1 h O_1 h O_1 l O_2 h O_2$

then $\mathtt{view}_{Low}(\alpha) = O_1 l O_2$

Propositions

A proposition ϕ is a fact about sequences of actions.

Formally $\phi \subseteq Actions^*$, and we say ϕ holds at $\alpha \in Actions^*$ if $\alpha \in \phi$.

ϕ is non-trivial if it is not $\emptyset$ or $Actions^*$.

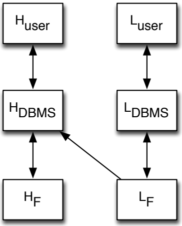
ϕ is G-local, for $G \subseteq D$, if it depends only on actions of the domains G

Formally, $\alpha|G = \beta|G$ implies $\alpha \in \phi$ iff $\beta \in \phi$.

Reasoning about Knowledge

Say domain u knows a proposition ϕ after a sequence $\alpha \in \mathit{Actions}^*$ in a system M if ϕ holds at β for all sequences $\beta \in \mathit{Actions}^*$ such that $\mathtt{view}_u(\alpha) = \mathtt{view}_u(\beta)$.

Notation: $M, \alpha \models \mathtt{Knows}_u(\phi)$



Theorem: Suppose that M is TA-secure with respect to the Hinke-Schaeffer policy. Then for all $\{H_{user}, H_{DBMS}, H_F\}$-local propositions ϕ, and $\alpha \in Actions^*$

$$M, \alpha \models \neg \mathtt{Knows}_{L_{user}}(\phi)$$

Extended Architectures

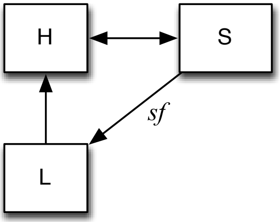
(To model restrictions on the behaviour of trusted components)

An extended architecture is a pair $(D, \rightarrowtail)$ where $\rightarrowtail \subseteq D \times D \times \mathcal{L}$, where

- D is a set of domains
- $\mathcal{L}$ is a set of function names, including the special name $\top$
- $(u, v, f) \in \rightarrowtail$ means "information is permitted to flow from u to v, but must be *filtered* through the function denoted by f".
- $\top$ means "no constraints on information flow across this edge"

Example: Starlight Interactive Link

(Anderson et al. – A switch that allows a user to alternate their keyboard between High and Low level windows.)



Architectural Interpretations

An *interpretation* of an extended architecture consists of

- A set of actions *Actions*
- A domain assignment $\mathtt{dom} : \mathit{Actions} \rightarrow D$
- An interpretation function $\mathtt{I}$, such that for each $f \in \mathcal{L} \setminus \{\top\}$, $\mathtt{I}(f)$ is a function with domain $\mathit{Actions}^*$

Intuitively, if $(u, v, f) \in \rightarrowtail$ and $\alpha \in \mathit{Actions}^*$ and $a \in \mathit{Actions}$ with $\mathtt{dom}(a) = u$, then $\mathtt{I}(f)(\alpha a)$ is "the information permitted to flow from u to v when u does a after α."

Semantics

Given an extended architecture $\mathcal{A} = (D, \rightarrowtail)$ and an architectural interpretation $\mathcal{I} = (Actions, \mathtt{dom}, \mathtt{I})$, we can define for each $u \in D$ the function $\mathtt{tff}_u$ with domain $Actions^*$ by $\mathtt{tff}_u(\epsilon) = \epsilon$ and

$$\mathtt{tff}_u(\alpha a) = \begin{cases} \mathtt{tff}_u(\alpha) & \text{if } \mathtt{dom}(a) \not\rightarrowtail u \\ \mathtt{tff}_u(\alpha)\ (\mathtt{tff}_{\mathtt{dom}(a)}(\alpha), a) & \text{if } \mathtt{dom}(a) \overset{\top}{\rightarrowtail} u \\ \mathtt{tff}_u(\alpha)\ \mathtt{I}(f)(\alpha a) & \text{if } \mathtt{dom}(a) \overset{f}{\rightarrowtail} u \end{cases}$$

Semantics

Given an extended architecture $\mathcal{A} = (D, \rightarrowtail)$ and an architectural interpretation $\mathcal{I} = (Actions, \mathtt{dom}, \mathtt{I})$, we can define for each $u \in D$ the function $\mathtt{tff}_u$ with domain $Actions^*$ by $\mathtt{tff}_u(\epsilon) = \epsilon$ and

$$\mathtt{tff}_u(\alpha a) = \begin{cases} \mathtt{tff}_u(\alpha) & \text{if } \mathtt{dom}(a) \not\rightarrowtail u \\ \mathtt{tff}_u(\alpha)\ (\mathtt{tff}_{\mathtt{dom}(a)}(\alpha), a) & \text{if } \mathtt{dom}(a) \overset{\top}{\rightarrowtail} u \\ \mathtt{tff}_u(\alpha)\ \mathtt{I}(f)(\alpha a) & \text{if } \mathtt{dom}(a) \overset{f}{\rightarrowtail} u \end{cases}$$

M complies with the interpreted architecture $(\mathcal{A}, \mathcal{I})$ if for all $u \in D$ and $\alpha, \beta \in Actions^*$, if $\mathtt{tff}_u(\alpha) = \mathtt{tff}_u(\beta)$ then $O_u(s_0 \cdot \alpha) = O_u(s_0 \cdot \beta)$.

Architectural Specifications

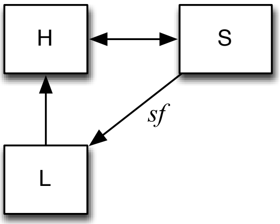
An architectural specification consists of

- An extended architecture $\mathcal{A}$
- A set $\mathcal{C}$ of interpretations for this architecture

This captures architecture + constraints on the behaviour of trusted components.

M complies with an architectural specification $(\mathcal{A}, \mathcal{C})$ if it complies with $(\mathcal{A}, \mathcal{I})$ for some $\mathcal{I} \in \mathcal{C}$.

Example: Starlight Interactive Link



$+\ \mathcal{C}$ = all interpretations such that

- *Actions* contains a toggle action t with $\mathtt{dom}(t) = S$
- $\mathtt{I}(sf)(\alpha a) = a$ if $a = t$ or $\mathtt{dom}(a) = S$ and α contains an odd number of t's,
  otherwise $\mathtt{I}(sf)(\alpha a) = \epsilon$ (no information flow)

Reasoning based on Architectural Specifications

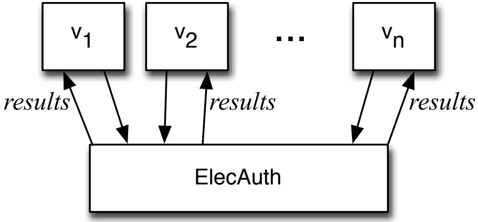
Say that a proposition ϕ is *toggle-High dependent* if it depends only on the subsequence of α consisting of

- all actions a with $\mathtt{dom}(a) = H$
- all occurrences of actions a with $\mathtt{dom}(a) = S$ that occur between an even numbered occurrence of t and any subsequent occurrence of t.

Theorem: Suppose that M complies with the Starlight architectural specification. Let ϕ be toggle-High dependent and non-trivial. Then

$$M, \alpha \models \neg\mathtt{Knows}_L(\phi)$$

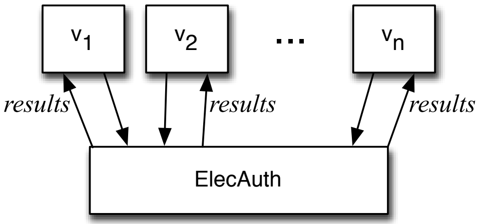
Electronic Election



Specification:

- *Actions* consists of actions of `ElecAuth` plus actions a_v for v a voter and $a \in \texttt{VoterActions}$; with $\texttt{dom}(a_v) = v$.
- For a permutation P of $\{v_1, \ldots, v_n\}$ and $\alpha \in \mathit{Actions}^*$, let $P(\alpha)$ be the result of replacing each a_v in α by $a_{P(v)}$.
- For all α, $\texttt{results}(\alpha) = \texttt{results}(P(\alpha))$.

Electronic Election



Specification:

- *Actions* consists of actions of `ElecAuth` plus actions a_v for v a voter and $a \in \mathtt{VoterActions}$; with $\mathtt{dom}(a_v) = v$.
- For a permutation P of $\{v_1, \ldots, v_n\}$ and $\alpha \in Actions^*$, let $P(\alpha)$ be the result of replacing each a_v in α by $a_{P(v)}$.
- For all α, $\mathtt{results}(\alpha) = \mathtt{results}(P(\alpha))$. (Examples: $\mathtt{results}(\alpha) =$ number of votes cast for each candidate $\mathtt{results}(\alpha) =$ the candidate(s) with the most votes.)

Theorem: If M complies with this specification, P is a permutation of voters and v a particular voter such that $P(v) = v$ and ϕ is a proposition, if

$$M, \alpha \models \neg \texttt{Knows}_v(\neg\phi)$$

then

$$M, \alpha \models \neg \texttt{Knows}_v(\neg P(\phi))$$

Example: "if Alice considers it possible that
Bob voted for Obama and Charlie voted for McCain,
then Alice considers it possible that
Charlie voted for Obama and Bob voted for McCain."

Access Control

Following Rushby 92, a *system with structured state* is a machine $\langle S, s_0, Actions, \mathtt{step}, O, \mathtt{dom}\rangle$ together with

- a set N of *names*,
- a set V of *values*, and functions
- $\mathtt{contents} : S \times N \to V$, with $\mathtt{contents}(s, n)$ interpreted as the value of object n in state s,
- $\mathtt{observe} : D \to \mathcal{P}(N)$, with $\mathtt{observe}(u)$ interpreted as the set of objects that domain u can observe, and
- $\mathtt{alter} : D \to \mathcal{P}(N)$, with $\mathtt{alter}(u)$ interpreted as the set of objects whose values domain u is permitted to alter.

Reference Monitor Conditions

Define a binary relation $\sim_u^{oc}$ of *observable content equivalence* on S for each domain $u \in D$, by $s \sim_u^{oc} t$ if $\mathtt{contents}(s, n) = \mathtt{contents}(t, n)$ for all $n \in \mathtt{observe}(u)$.

RM1. If $s \sim_u^{oc} t$ then $O_u(s) = O_u(t)$.

RM2′ For all actions a states s, t and names $n \in \mathtt{alter}(dom(a))$, if $s \sim_{\mathtt{dom}(a)}^{oc} t$ and $\mathtt{contents}(s, n) = \mathtt{contents}(t, n)$ we have $\mathtt{contents}(s \cdot a, n) = \mathtt{contents}(t \cdot a, n)$.

RM3. If $\mathtt{contents}(s \cdot a, n) \neq \mathtt{contents}(s, n)$ then $n \in \mathtt{alter}(dom(a))$.

(RM2′ a variant, from van der Meyden - ESORICS 2007, of Rushby's RM2)

Consistency of access control with a noninterference policy:

AOI. If $\texttt{alter}(u) \cap \texttt{observe}(v) \neq \emptyset$ then $u \rightarrowtail v$.

Proposition

If M is a system with structured state satisfying RM1-RM3 and AOI with respect to noninterference policy $\rightarrowtail$ then M is TA-secure (hence IP-secure) for $\rightarrowtail$.

Adapting the Conditions to Extended Architectures

AOI′. If $\mathtt{alter}(u) \cap \mathtt{observe}(v) \neq \emptyset$ then $u \overset{f}{\rightarrowtail} v$ for some f.

Extra conditions for filtered edges:

I1. If $\mathtt{dom}(a) \overset{f}{\rightarrowtail} u$ for $f \neq \top$ and $\mathtt{I}(f)(\alpha, a) = \epsilon$ and $x \in \mathtt{observe}(u) \cap \mathtt{alter}(\mathtt{dom}(a))$ then $(s_0 \cdot \alpha a)(x) = (s_0 \cdot \alpha)(x)$.

I2. If $\mathtt{dom}(a) \overset{f}{\rightarrowtail} u$ with $f \neq \top$ and $\mathtt{dom}(b) \overset{g}{\rightarrowtail} u$ with $f \neq \top$ and $\mathtt{I}(f)(\alpha, a) = \mathtt{I}(g)(\beta, b) \neq \epsilon$ and $x \in \mathtt{observe}(u) \cap (\mathtt{alter}(\mathtt{dom}(a)) \cup \mathtt{alter}(\mathtt{dom}(b)))$ and $(s_0 \cdot \alpha)(x) = (s_0 \cdot \beta)(x)$ then $(s_0 \cdot \alpha a)(x) = (s_0 \cdot \beta b)(x)$.

Theorem

Let $\mathcal{AI}$ be an interpreted architecture. Suppose that M is a system with structured state satisfying RM1-RM3, AOI$'$ and I1-I2. Then M is TFF-compliant with $\mathcal{AI}$.

Conclusion/Future Work

The examples demonstrate cases where it is feasible to formally derive global properties from an abstract level of specification of architecture + properties of trusted components
Many issues remain:

- Are there classes of specifications that can be straightforwardly implemented?
- Connections to other implementation patterns: e.g., periods processing, network partitioning.
- Richer semantics of architectures, e.g., for timing, probabilistic attacks.
- Syntax for architectural specifications, efficiently automatable cases of verification.