Formal Foundations for Security Architecture

Ron van der Meyden

(University of New South Wales
Sydney, Australia)

May 5, 2010
Overview

- Some recent Australian events
- MILS Security
- Towards a formal theory of architecture
Military Grade Security, coming to an enterprise near you

An Australian example:

- Australian iron ore exports to China: $22 billion in 2009
- Annual price setting negotiations, large increases (2010, 100%)
- (Australian) Stern Hu, chief negotiator for Rio Tinto, arrested & convicted by China on charges of bribery and ‘stealing state secrets’
- ‘sophisticated’ hacking attacks on the major mining companies
- Defence Signals Directorate steps in to assist with network security
MILS Security

- DOD/NSA Initiative within Crypto Modernization Program
- Rushby (SRI) proposed Separation Kernel 1981
- Initial (vague) view: MILS = use of Rushby Separation Kernel
- Standardization through Open Group Real Time and Embedded Systems Forum
- Research track: OG RTES Forum Annual Research Day since 2008, possibly to be integrated with Layered Assurance Workshop
- Under discussion: what is it really?
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
Questions concerning this vision:

- What is the formal semantics of architectural designs?
- Is it really possible to prove interesting security properties in this architecture + trusted component, local to global, pattern?
- Refinement theory for the extension
Semantics of Architectures


A system satisfies an architecture if whenever you compare a possibly history of the system, with a variant where all events not permitted by the architecture to affect agent A have been removed, the two histories look the same from A’s point of view.

Cf. relativity theory: an event has no effect on you unless you are in its causal cone.
A problem with the classical theory
(van der Meyden ESORICS-07)

There exists a system that

- is secure with respect to this policy
- enables $L$ to learn a fact about $H_1, H_2$ that would not be known to $D_1, D_2$ even if they were to combine their information
Revised Security definition
(van der Meyden ESORICS-07)

Build a concrete operational model of the maximal knowledge that each agent is permitted to have, according to the security policy/architecture.

A system satisfies the architecture if no agent ever has more than its maximal permitted information.

This

- is more intuitive
- fixes the problem
- leads to a more pleasant theory, with stronger connections to access control implementations.
Extensions

(current work with Steve Chong)

A class of architectural specifications consisting of

- policies with labelled edges $u \xrightarrow{f} v$
- a set of constraints defining possible interpretations of $f$ a function; $f(history, action) = \text{maximal information permitted to flow from } u \text{ to } v \text{ when } u \text{ performs } action$ after $history$
Example: Starlight Interactive Link

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

Specification: \( sf(history, action_S) = \text{nothing}, \) if \( S \) toggled to \( H \), else \( action_S \)

**Theorem:** If a system satisfies this architectural specification, then \( L \) never knows anything about \( H \), or about \( S \) while toggled to \( H \).
Electronic Election

Specification:

- All voters have the same types of actions available.
- For all permutations $P$ of $\{v_1, \ldots, v_n\}$ and all sequences of actions $\alpha$, $\text{results}(\alpha) = \text{results}(P(\alpha))$, where $P(\alpha)$ is the result of replacing each action $a$ done by $v$ in $\alpha$ by $a$ done by $P(v)$. 
Electronic Election

Specification:

- All voters have the same types of actions available.
- For all permutations $P$ of $\{v_1, \ldots, v_n\}$ and all sequences of actions $\alpha$, $\text{results}(\alpha) = \text{results}(P(\alpha))$, where $P(\alpha)$ is the result of replacing each action $a$ done by $v$ in $\alpha$ by $a$ done by $P(v)$.

(E.g. $\text{results}(\alpha) = \text{number of votes for each candidate}$, or $\text{results}(\alpha) = \text{the candidate(s) with the most votes}$.)
**Theorem: (Anonymity)** If a system satisfies this architectural specification, $P$ is a permutation of voters and $v$ a particular voter such that $P(v) = v$ and $\phi$ is a proposition, if $v$ considers $\phi$ possible then $v$ considers $P(\phi)$ possible.

**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.”
Hinke-Schaeffer as a refinement of HL
A Refinement Theory

Formal definition of refinement

Results stating that if

1. architecture $A$ is a refinement of architecture $B$ and
2. $M$ implements $A$

then

1. $\text{abstract}(M)$ implements $B$
2. if set of components $\alpha$ in $A$ correspond to component $\beta$ in $B$ and $\phi$ is not known to $\beta$ in $\text{abstract}(M)$ then $\phi$ is not distributed knowledge to $\alpha$ in $M$
Example: A downgrader

 Specification: \( rel \) transmits only information that \( D \) would have were the architecture restricted to \( \{ H_P, H, D \} \).

 \( H_P \): the “publicly acknowledged, declassifiable High domain”

 \( H_C \): the “covert High domain”

 **Theorem:** \( L \) does not know anything about \( H_C \)
A downgrader - refinement

Specification: \textit{reqRel} transmits only information that \textit{Req} has approved for release and that \textit{Rel} would have were the architecture restricted to \{\textit{HP}, \textit{H}, \textit{D}\}.

**Theorem:** \textit{L} does not know anything about \textit{HC}

(This can be proved straightforwardly by showing that this architecture is a refinement of the previous).
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 architectural specifications that can be implemented using standard patterns?
- Connections to concrete mechanisms, e.g., access control.
- How to implement such specifications in general? (E.g., downgrader requires secure provenance mechanisms within High domain.)
- Richer semantics of architectures, e.g., for timing, probability
- Syntax for architectural specifications, efficiently automatable cases of verification.
Our focus is information flow (noninterference) and knowledge, cf.

- Garlan, behavioural rather than security properties
- Moriconi, Qian, Riemenschneider, c. 1995, Bell La Padula
- Jan Jurjens: UML based, crypto protocol focus
- Jurgen Hansson: Bell La Padula labels, role based access control in AADL (Architecture Analysis and Design Language)