CS590U
Access Control: Theory and Practice

Lecture 18 (March 10)
SDSI Semantics & The RT Family of Role-based Trust-management Languages
Understanding SPKI/SDSI
Using First-Order Logic

Ninghui Li and John C. Mitchell
What is a Semantics?

- Elements of a semantics
  - syntax for statements
  - syntax for queries
  - an entailment relation that determines whether a query Q is true given a set P of statements
Why a Formal Semantics?

- What can we gain by a formal semantics
  - understand what queries can be answered
  - defines the entailment relation in a way that is precise, easy to understand, and easy to compute

- How can one say a semantics is good
  - subjective metrics:
    - simple, natural, close to original intention
  - defines answers to a broad class of queries
  - can use existing work to provide efficient deduction procedures for answering those queries
Summary of SDSI Semantics

- Rewriting based
  - can answer queries such as can one string rewrites into another one
- Set based
  - can answer queries such as which principals are in the valuation of a string
- Logic programming based
- First-Order Logic based
A Logic-Programming-based Semantics

- Translate each 4-tuple into a LP clause
  - Using a ternary predicate \( m \)
    - \( m(K, A, K') \) is true if \( K' \in V(K A) \)
    - \((K A \Rightarrow K')\) to \( m(K, A, K') \)
    - \((K A \Rightarrow K_1 A_1)\) to \( m(K, A, ?x) :- m(K_1, A_1, ?x) \)
    - \((K A \Rightarrow K_1 A_1 A_2)\)
      - to \( m(K, A, ?x) :- m(K_1, A_1, ?y_1), m(?y_1, A_2, ?x) \)
    - \((K A \Rightarrow K_1 A_1 A_2 A_3)\)
      - to \( m(K, A, ?x) :- m(K_1, A_1, ?y_1), m(?y_1, A_2, ?y_2), m(?y_2, A_3, ?x) \)
  - The minimal Herbrand model determines the semantics
Example

From

(k_C mit \Leftrightarrow k_M)
(k_M \text{ faculty} \Leftrightarrow k_{\text{EECS}} \text{ faculty})
(k_C \text{ access} \Leftrightarrow k_C \text{ mit faculty secretary})

To

m(k_C, \text{ mit}, k_M).
m(k_M, \text{ faculty}, Z) :- m(k_{\text{EECS}}, \text{ faculty}, Z).
m(k_C, \text{ access}, Z) :- m(k_C, \text{ mit}, Y_1),
\quad m(Y_1, \text{ faculty}, Y_2), m(Y_2, \text{ secretary}, Z).
Set semantics is equivalent to LP semantics

- The least Herbrand model of SP[P] is equivalent to the least valuation, i.e.,
  - $K' \in \mathcal{V}_p (K \ A)$ iff. $m(K,A,K')$ is in the least Herbrand model of SP[P]

- Same limitation as set-based semantics
  - does not define answers to containment between arbitrary name strings
An Alternative Way of Defining the LP-based Semantics (1)

- Define a macro `contains`
  - `contains[\omega][K']` means that $K' \in V(\omega)$
    - `contains[K][K']` $\equiv (K = K')$
    - `contains[K A][K']` $\equiv m(K, A, K')$
    - `contains[K A_1 A_2 \ldots A_n][K']` $\equiv$
      $\exists y (m(K, A_1, y) \land contains[y A_2 \ldots A_n][K'])$
      where $n > 1$
An Alternative Way of Defining the LP-based Semantics (2)

- Translates a 4-tuple \((K A \Rightarrow \omega)\) into a FOL sentence
  - \(\forall z (\text{contains}[K A][z] \iff \text{contains}[\omega][z])\)
- This sentence is also a Datalog clause
- A set \(P\) of 4-tuples defines a Datalog program, denoted by \(SP[P]\)
  - The minimal Herbrand model of \(SP[P]\) defines the semantics
An Example of Translation

From \((K_C \text{ access } \Rightarrow K_C \text{ mit faculty secretary})\)

\[ \forall z \ ( \text{contains}[K_C \text{ access}][z] \iff \text{contains}[K_C \text{ mit faculty secretary}][z] ) \]

\[ \forall z \ ( m(K_C, \text{ access}, z) \iff \exists y_1 \ (m(K_C, \text{ mit}, y_1) \land \text{contains}[y_1 \text{ faculty secretary}][z] ) ) \]

\[ \forall z \ ( m(K_C, \text{ access}, z) \iff m(K_C, \text{ mit}, y_1) \land \exists y_2 \ (m(y_1, \text{ faculty}, y_2) \land \text{contains}[y_2 \text{ secretary}][z] ) ) \]

\[ \forall z \ ( m(K_C, \text{ access}, z) \iff m(K_C, \text{ mit}, y_1) \land m(y_1, \text{ faculty}, y_2) \land m(y_2, \text{ secretary}, z) ) \]
A First-Order Logic (FOL) Semantics

- A set $P$ of 4-tuples defines a FOL theory, denoted by $\text{Th}[P]$
- A query is a FOL formula
  - “$\omega_1$ rewrites into $\omega_2$” is translated into $\forall z (\text{contains}[\omega_1][z] \iff \text{contains}[\omega_2][z])$
  - Other FOL formulas can also be used as queries
- Logical implication determines semantics
FOL Semantics is Extension of LP Semantics

- LP semantics is FOL semantics with queries limited to LP queries
  - $m(K,A,K')$ is in the least Herbrand model of $SP[P]$ iff. $Th[P] \models m(K,A,K')$
Theorem: for string rewriting queries, the string rewriting semantics is equivalent to the FOL semantics

Given a set $P$ of 4-tuples, it is possible to rewrite $\omega_1$ into $\omega_2$ using the 4-tuples in $P$ if and only if

$Th[P]^2 \forall z \ (\text{contains}[\omega_1][z] \iff \text{contains}[\omega_2][z])$
Advantages of FOL semantics: Computation efficiency

- A large class of queries can be answered efficiently using logic programs
  - including rewriting queries
  - e.g., whether \( \omega \) rewrites into \( K B_1 B_2 \) under \( P \) can be answered by determining whether \( SP[P \cup (K' A' \Rightarrow \omega) \cup (K B_1 \Rightarrow K'_1) \cup (K'_1 B_2 \Rightarrow K'_2)]^2 m(K',A', K'_2) \)
    - where \( K' \), \( K'_1 \), and \( K'_2 \) are new principals
    - this proof procedure is sound and complete
      - this result also follows from results in proof theory regarding Harrop Hereditary formulas
Advantages of FOL semantics: Extensibility

- Additional kinds of queries can be formulated and answered, e.g.,
  - \( \forall z \ (m(K_1, A_1, z) \iff m(K_1, A_2, z)) \)
    \( \iff \exists z \ (m(K_2, A_1, z) \land m(K_2, A_2, z)) \)

- Additional forms of statements can be easily handled, e.g.,
  - \( (K A \Rightarrow K_1 A_1 \cap K_2 A_2) \) maps to
    \( \forall z \ (m(K, A, z) \iff m(K_1, A_1, z) \land m(K_2, A_2, z)) \)
Summary: 4 Semantics for SDSI

- **String Rewriting:** difficult to extend
- **Set:** limited in queries
- **First-Order Logic**
- **Logic Programming**
Advantages of FOL Semantics: Summary

- Simple
  - captures the set-based intuition
  - defined using standard FOL

- Extensible
  - additional policy language features can be handled easily
  - allow more meaningful queries

- Computation efficiency
Design of A Role-based Trust-management Framework

Ninghui Li, John C. Mitchell & William H. Winsborough
IEEE S&P 2002
Features of the RT family of TM languages

- Expressive delegation constructs
- Permissions for structured resources
- A tractable logical semantics based on Constraint Datalog
- Strongly-typed credentials and vocabulary agreement
- Efficient deduction with large number of distributed policy statements
- Security analysis
Expressive Features (part one)

I. Simple attribute assignment
   \[\text{StateU.stuID} \leftarrow \text{Alice}\]

II. Delegation of attribute authority
   \[\text{StateU.stuID} \leftarrow \text{COE.stuID}\]

III. Attribute inferencing
    \[\text{EPub.access} \leftarrow \text{EPub.student}\]

IV. Attribute-based delegation of authority
    \[\text{EPub.student} \leftarrow \text{EPub.university.stuID}\]
Expressive Features (part two)

v. Conjunction

\[ \text{EPub.access} \leftarrow \text{EPub.student} \cap \text{ACM.member} \]

vi. Attributes with fields

- \( \text{StateU.stul D} \text{ (name} = \ldots, \text{ program} = \ldots, \ldots) \leftarrow \text{Alice} \)
- \( \text{EPub.access} \leftarrow \text{StateU.stul D} \text{(program}=\text{“graduate”}) \)

vii. Permissions for structured resources

- e.g., allow connection to any host in a domain and at any port in a range
The Languages in the RT Framework

RT\textsuperscript{T}: for Separation of Duties

RT\textsubscript{0}: Decentralized Roles

RT\textsubscript{1}: Parameterized Roles

RT\textsuperscript{T}: for Separation of Duties

RT\textsubscript{2}: Logical Objects

RT\textsubscript{2}\textsuperscript{C}: structured resources

RT\textsubscript{1}\textsuperscript{C}: structured resources

RT\textsubscript{D}: for Selective Use of Role memberships

RT\textsuperscript{T} and RT\textsuperscript{D} can be used (either together or separately) with any of the five base languages: RT\textsubscript{0}, RT\textsubscript{1}, RT\textsubscript{2}, RT\textsubscript{1}\textsuperscript{C}, and RT\textsubscript{2}\textsuperscript{C}
\( \mathbf{RT_1} = \mathbf{RT_0} + \text{Parameterized Roles} \)

- **Motivations:** to represent
  - attributes that have fields, e.g., digital ids, diplomas
  - relationships between principals, e.g., physicianOf, advisorOf
  - role templates, e.g., project leaders

- **Approach:**
  - a role term \( R \) has a role name and a list of fields
Example 1: Alpha allows manager of an employee to evaluate the employee:

\[
\text{Alpha.evaluatorOf(employee=y)} \leftarrow \\
\text{Alpha.managerOf(employee=y)}
\]

Example 2: EPub allows CS students to access certain resources:

\[
\text{EPub.access(action='read', resource='file1')} \leftarrow \\
\text{EPub.university.stuId(D(dept='CS'))}
\]
RT₁ (Technical Details)

- A credential takes one of the following form:
  1. K.r(h₁, ..., hₙ) ↴ K₂
  2. K.r(h₁, ..., hₙ) ↴ K₁.r₁(s₁, ..., sₘ)
  3. K.r(h₁, ..., hₙ) ↴ K₁.r₁(t₁, ..., tₗ).r₂(s₁, ..., sₘ)
  4. K.R ↴ K₁.R₁ ∩ K₂.R₂ ∩ ... ∩ Kₖ.Rₖ

- Each variable
  - must have a consistent data type across multiple occurrences
  - can have zero or more static constraints
  - must be safe, i.e., must appear in the body
Semantics and Complexity for RT₁

- LP semantics makes each role name a predicate
  - E.g., K.r(h₁, ..., hₙ) ← K₁.r₁(s₁, ..., sₘ) translates to r(K, h₁, ..., hₙ, ?X) :- r₁(K₁, s₁, ..., sₘ, ?X)

- Apply known complexity results: The atomic implications of SP(ᵰ) can be computed in O(Nᵩ₊³)
  - \( v \) is the max number of variables per statement
  - Each role name has a most \( p \) arguments
  - \( N = \max(N₀, pN₀) \), \( N₀ \) is the number of statements in \( ₰ \)
RT₂ = RT₁ + Logical Objects

- Motivations:
  - to group logically related objects together and assign permissions about them together

- Approach: introducing o-sets, which are
  - similar to roles, but have values that are sets of things other than entities
  - defined through o-set definition credentials, which are similar to role-definition credentials in RT₁
RT_2 (Examples)

- Example 1: Alpha allows members of a project team to read documents of this project
  \[
  \text{Alpha.documents}(\text{projectB}) \leftarrow \text{“design_Doc_for_projectB”} \\
  \text{Alpha.team}(\text{projectB}) \leftarrow \text{Bob} \\
  \text{Alpha.fileAccess(read, } ?F \in \text{Alpha.documents(?proj)}) \leftarrow \text{Alpha.team(?proj)}
  \]

- Example 2: Alpha allows manager of the owner of a file to access the file
  \[
  \text{Alpha.read(?F)} \leftarrow \text{Alpha.manager(?E} \in \text{Alpha.owner(?F)})
  \]
RT^T: Supporting Threshold and Separation-of-Duty

- Threshold: require agreement among k principals drawn from a given list
- SoD: requires **two or more different** persons be responsible for the completion of a sensitive task
  - want to achieve SoD without mutual exclusion, which is nonmonotonic
- Though related, neither subsumes the other
- RT^T introduces a primitive that supports both: manifold roles
Manifold Roles

- While a standard role is a set of principals, a manifold role is a set of sets of principals.
- A set of principals that together occupy a manifold role can collectively exercise privileges of that role.
- Two operators: $\oplus$, $\otimes$
  - $K_1.R_1 \oplus K_2.R_2$ contains sets of two distinct principals, one a member of $K_1.R_1$, the other of $K_2.R_2$
  - $K_1.R_1 \otimes K_2.R_2$ does not require them to be distinct
RT$^\top$ (Examples)

- Example 1: require a manager and an accountant
  - $\text{K.approval} \leftarrow \text{K.manager} \odot \text{K.accountant}$
  - $\text{members(K.approval)} \supseteq \{\{x,y\} \mid x \in \text{K.manager}, \ y \in \text{K.accountant}\}$

- Example 2: require a manager and a different accountant
  - $\text{K.approval} \leftarrow \text{K.manager} \otimes \text{K.accountant}$
  - $\text{members(K.approval)} \supseteq \{\{x,y\} \mid x \neq y, \ x \in \text{K.manager}, \ y \in \text{K.accountant}\}$
Example 3: require three different managers

- \( K.\text{approval} \leftarrow K.\text{manager} \otimes K.\text{manager} \otimes K.\text{manager} \)
- \( \text{members}(K.\text{approval}) \supseteq \{ \{x,y,z\} \mid x \neq y \neq z \in K.\text{manager} \} \)
Manifold roles can be used in basic RT statements

Also add two new types of policy statement

- \( K \cdot R \leftarrow K_1 \cdot R_1 \oplus K_2 \cdot R_2 \oplus \cdots \oplus K_k \cdot R_k \)
  
  \[
  \text{members}(K \cdot R) \oplus \{ s_1 \oplus \cdots \oplus s_k \mid s_i \oplus \text{members}(K_i \cdot R_i) \text{ for } 1 = i = k \}
  \]

- \( K \cdot R \leftarrow K_1 \cdot R_1 \oplus K_2 \cdot R_2 \oplus \cdots \oplus K_k \cdot R_k \)
  
  \[
  \text{members}(K \cdot R) \oplus \{ s_1 \oplus \cdots \oplus s_k \mid (s_i \oplus \text{members}(K_i \cdot R_i) \& s_i \cap s_i = \emptyset) \text{ for } 1 = i \oplus j = k \}
  \]
RT$^T$ Complexity

- ADSD must declare a size for each manifold role.

- Given a set $\mathbf{P}$ of RT$^T$ statements, let $t$ be the maximal size of all roles in $\mathbf{P}$. The atomic implications of $\mathbf{P}$ can be computed in time $O(MN^{v+2t})$. 
Implementation and Application Status of RT

- Java Implementation of inference engine for $RT_0$
- Preliminary version of RTML
  - an XML-based Encoding of RT statements
  - XML Schemas and parser exist
  - Used in an ATN demo
- Applications
  - U-STOR-IT: Web-based file storage and sharing
  - August: A Distributed Calendar Program
  - Automated Trust Negotiation Demo by NAI
Next Lecture

- Security analysis in Trust Management