Design of A Role-based Trust-management Framework

Ninghui Li (Stanford University)
John Mitchell (Stanford University)
Will Winsborough (NAI Labs)

Outlines

- What is Role-based TM and why Role-based TM?
- RT: A Role-based TM framework.

What We Mean by Trust-Management?

- A phrase coined in [BFL’96]
- Groups together several principles in dealing with authorization in distributed systems:
  - authorization information is in the form of digitally signed credentials and local policies
  - a language for expressing authorization information (content of credentials and policies)
  - the language has an application-independent semantics

Permission-based (capability-style) Systems

- Examples: KeyNote
  - A credential delegates certain permissions
    - from an issuer to a subject
  - A chain of credentials can be viewed as a capability authorizing the subject at the end of the chain
  - cannot delegate to entities who have a certain property

What Is Missing in Permission-based Systems?

- Consider a simple example scenario:
  - a book store gives a discount to students of a university
- A natural solution:
  - the book store has one policy statement saying that anyone who shows a student ID gets the discount
  - the university issues student IDs to students
- Cannot use the natural solution with permission-based systems:
  - unable to represent the policy to grant discounts to students

What Is Missing in Permission-based Systems? (Continued)

- Solution 1 in KeyNote:
  - bookstore delegates the discount permission to university
  - university delegates the discount permission to each student
  - university has to do this for every business!
- Solution 2 in KeyNote:
  - university creates a key pair to represent the student group; a student ID is a complete delegation from this key to a student
  - bookstore delegates the discount permission to this group key for student
  - bookstore needs to know which public key corresponds to the student group; requires another TM system for this!
Expressive Features in Role-based TM Languages (first slide)

I. Decentralized attributes (roles)
   StateU.student \(\not\in\) Alice

II. Delegation of attribute authority
    Store.student \(\not\in\) StateU.student

III. Attribute inferencing
     Store.discount \(\not\in\) Store.student

IV. Attribute-based delegation of authority
    Store.student \(\not\in\) Store.university.student
    member(Store.student) \(\subseteq\)
    member(Store.university) member(Store.student)

Expressive Features in Role-based TM Languages (second slide)

V. Conjunction
   - Store.deepdiscount \(\not\in\) Store.student \(\cap\) IEEE.member

VI. Attributes with fields (or parameterized roles)
    - California.digital_Driver_License (name=.., bday=.., ...
    \(\not\in\) K_Alice
    - Store.discount \(\not\in\) Store.student(type="graduate")

This list is incomplete, other features, including separation of duties, selective use of role memberships, will be discussed later.

What about SPKI/SDSI?

- SPKI 5-tuples (similar limitations as KeyNote)
  - lacks attribute inferencing, conjunction, attribute-based delegation
- SDSI name definition certificates (name certs)
  - lacks conjunction, attribute fields
- SPKI/SDSI combines 5-tuples and name certs
  - still lacks conjunctions, attribute with fields, e.g.,
    Store.deepdiscount \(\not\in\) Store.student \(\cap\) IEEE.member
    Store.discount \(\not\in\) Store.student(type="graduate")
- No support for selective use of role memberships

What about Other TM Systems?

- Statement-based systems, e.g., Delegation Logic, QCM, SD3
  - lack selective use of role memberships
  - lack explicit abstraction of subjects and attributes/roles

Why Explicit Subjects and Attributes?

- The explicit abstraction of subjects and attributes in RT
  - allows simple representation of delegation concepts
  - enables distributed credential chain discovery
  - enables the borrowing of concepts from RBAC, e.g., session and role activations

Outlines

- What is Role-based TM and why Role-based TM?
- RT: A Role-based TM framework.
Overview of the RT Framework

- A family of TM languages that borrows from previous TM systems (SDSI, DL), RBAC, and SRC logic
- supports the six expressive features listed earlier
- supports separation of duties
- supports selective use of role memberships
- Each credential is translated straightforwardly to one Datalog rule
- defines semantics as well as shows tractability
- Infrastructure support for agreement on role name meanings and distributed credential storage and discovery

RT Framework: Concepts and Credentials

- Concepts:
  - Entities (A, B, D) and Role names (r, r₁, r₂, ...)
  - Roles: A.r, B.r...
- Each role has a value that is a set of entities

- Credentials:
  - Type-1: A.r ← D  (ABU.university ← StateU)
  - Type-2: A.r ← B₁.r₁  (Store.university ← ABU.university)
  - Type-3: A.r ← A. r₁  (Store.student ← Store.university.student)
  - Type-4: A.r ← B₁. r₁ ∩ B₂. r₂ ∩ ... ∩ Bₖ. rₖ  (Store.discount ← Store.student.Ç IEEE.member)

RT₁: Concepts and Credentials

- Example 1: Alpha allows manager of an employee to evaluate the employee
  \[\text{Alpha.evaluatorOf}(?y) \leftarrow \text{Alpha.managerOf}(?y)\]
- Example 2: Alpha gives a pay raise to an employee if his evaluator says that his performance was good
  \[\text{Alpha.payRaise} \leftarrow \text{Alpha.evaluator}(this).\text{goodPerformance}\]
- Example 3: StateU gives special privileges to alumni who graduated during certain years:
  \[\text{StateU.foundingAlumni} \leftarrow \text{StateU.diploma}(\text{?degree}, \text{?Year ÷ [1958..1962]})\]

The Languages in the RT Framework

- RT₀: Decentralized Roles
- RT₁: Parameterized Roles
- RT₂: Logical Objects

RT₀ and RT₁ can be used together or separately with any of RT₀, RT₁, or RT₂.

RT₁ = RT₀ + Parameterized Roles

- Motivations: to represent
  - role templates, e.g., course instructors, project leaders
  - relationships between entities, e.g., manages
  - roles and attributes that have fields, e.g., digital ids, diplomas

- Approach:
  - a role name \( R \) takes the form \( r(h₁, ..., hₙ) \), in which
    - \( r \) is a role identifier and each of \( h₁, ..., hₙ \) is a data term (a constant or a variable) of the appropriate data type
  - data types include integer types, float types, enumeration types, and data/time types

- Each credential takes one of the following form:
  1. \( A.r(h₁, ..., hₙ) \leftarrow D \)
  2. \( A.r(h₁, ..., hₙ) \leftarrow B₁.r₁ ∩ B₂.r₂ ∩ ... ∩ Bₖ.rₖ \)
  3. \( A.r(h₁, ..., hₙ) \leftarrow A.r(t₁, ..., tₖ) r₁(t₁, ..., tₖ) \)
  4. \( A.R \leftarrow B₁.R₁ ∩ B₂.R₂ ∩ ... ∩ Bₖ.Rₖ \)

- 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
RT\textsubscript{2} = RT\textsubscript{1} + 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 RT\textsubscript{1}

RT\textsuperscript{T} (Motivation): Separation of Duties (SoD)

- SoD requires two or more different persons be responsible for the completion of a sensitive task
- Previous TM systems use threshold structures for SoD
  - k out of one set of entities
  - cannot express: require a manager and an accountant
- RBAC uses mutually exclusive roles
  - no entity is allowed to occupy two mutually exclusive roles
  - a sensitive task requires a set of mutually exclusive roles
  - cannot express: require 3 managers
  - is non-monotonic

RT\textsuperscript{T} (Examples)

- Example 1: require a manager and an accountant
  - \texttt{A.approval} \textsuperscript{\&} \texttt{A.manager} \textsuperscript{\&} \texttt{A.accountant}
  - members(A.approval) \supset \{\{x,y\} | x \in A.manager, y \in A.accountant\}
- Example 2: require a manager and a different accountant
  - \texttt{A.approval} \textsuperscript{\&} \texttt{A.manager} \textsuperscript{\&} \texttt{A.different accountant}
  - members(A.approval) \supset \{\{x,y\} | x \in A.manager, y \in A.accountant, x \neq y\}
- Example 3: require three different managers
  - \texttt{A.approval} \textsuperscript{\&} \texttt{A.manager} \textsuperscript{\&} \texttt{A.manager} \textsuperscript{\&} \texttt{A.manager}
  - members(A.approval) \supset \{\{x,y,z\} | x \neq y \neq z \in A.manager\}

RT\textsuperscript{T} (The Approach)

- Introduce manifold roles
  - each member of a manifold role is a set of entities
- Introducing two new types of credentials using two new operators \textsuperscript{\&} and \textsuperscript{\&}
  5. \texttt{A.R} \textsuperscript{\&} \texttt{B1.R1} \textsuperscript{\&} ... \textsuperscript{\&} \texttt{Bk.Rk} Multiple role concurrence
  6. \texttt{A.R} \textsuperscript{\&} \texttt{B1.R1} \textsuperscript{\&} ... \textsuperscript{\&} \texttt{Bk.Rk} Separation of duties

RT\textsuperscript{D}: Selective Use of Role Memberships

- Motivation: Selective use of role memberships (capacities) when making a request, and delegation of these capacities
- RBAC has session and role activation (user-to-session delegation)
  - but not process-to-process delegation
- SRC logic has program-to-program delegation
  - but quite complex and intractable in the general case
- Approach in RT\textsuperscript{D}: introducing dynamic delegation credentials
  7. \texttt{B1} \rightarrow \texttt{B2} \rightarrow \texttt{D as A.R}
An Example Taken from Papers on SRC Logic, Expressed in RT_{DT}

\[ \text{S.delete(fileA)} \land \text{S.user} \rightarrow \text{S.goodWorkStation} \]

\[ \text{S.user} \land \text{K.Alice} \rightarrow \text{K.ws} \land \text{K_Alice as S.user} \]

\[ \text{K.ws} \land \text{K.p} \land \text{K.ws as S.goodWorkStation} \]

\[ \text{K.p} \land \text{K.ch} \land \text{K.ws as S.goodWorkStation, K.Alice as S.user} \]

\[ \text{K.ch} \land \text{delete(fileA)} \land \text{K.ws as S.goodWorkStation, K.Alice as S.user} \]

Summary of the RT Framework

- RT supports
  - expressive features for role-based (attribute-based) TM
  - decentralized attributes
  - delegation of attribute authorities
  - attribute inferencing
  - attribute-based delegation
  - conjunction
  - attribute with fields (RT_{1D})
- logical objects (RT_{2})
- separation of duties (RT_{T})
- selective use of role memberships (RT_{D})

Technical Results

- Each credential in RT is translated into one Datalog rule
- defines semantics as well as shows tractability
- Given a set \( C \) of credentials with total size \( M \), the worst-case time complexity for computing the implications of \( C \) is the \( O(M^{v+3}) \)
- \( RT_{1} \) and \( RT_{2} \) adds a factor of \( O(M^{v+3}) \)
- \( RT_{T} \) adds a factor of \( O(M^{2t}) \)
- \( RT_{D} \) adds a factor of \( O(M) \)

Related and Future Work

- Related work: distributed credential chain discovery in RT_{0}
  - extended abstract presented at CCS’01
  - full paper available from http://crypto.stanford.edu/~ninghui
- Ongoing and future work
  - An XML representation of credentials and domain specification documents
  - Extending the algorithm for RT_{0} to other components