An Introduction to Role Based Trust Management Framework

Samaneh Dehghan
Motivation

- Problem of guaranteeing that confidential data is not disclosed to unauthorized users
  - Access control techniques

- Access control in decentralized collaborative systems present difficult problems
  - Resources and the subjects requesting them belong to different security domains
Motivation (2)

- Traditional access control mechanism
  - make authorization decisions based on the identity of the resource requester.

- Decentralized environments
  - The resource owner and the requester often are unknown to one another
  - Making access control based on identity ineffective
Example

- A bookstore adopts the policy of giving 10% discount to students of accredited universities.
- A certificate authority may assert that the requester’s name is John Smith.
- If the name is unknown to the bookstore!
- What is needed is:
  - Info about the rights,… assigned to John smith by authorities.
  - Trust info about the authority itself.
Trust Management (TM)

- TM is a flexible approach to access control in decentralized distributed systems
  - access control decisions based on policy statements made by multiple principals.

- A key aspect of trust management is delegation:
  - A principal may transfer limited authority over one or more resources to other principals
AC vs. TM

- Typical access control mechanism
  - subject shows ID lookup authorization

- TM alternative
  - subject has Credentials infers authorization
We have a chain of credentials
- The subject of one is the issuer of the other one.
TM languages

- Early TM languages
  - PolicyMaker
  - KeyNote
  - SPKI (Simple Public Key Infrastructure) / SDSI (Simple Distributed Security Framework)

- Datalog-based TM languages
  - Delegation Logic
  - SD3 (Secure Dynamically Distributed Datalog)
  - Binder
  - RT: A Family of Role-based Trust-management Languages
To simplify authorization in collaborative environments, we need a system in which:
- Access control decisions are based on authenticated attributes of the subject
- Attribute authority is decentralized

We call such systems:
- Attribute-based Access Control (ABAC)
Attribute-based Access Control (2)

- ABAC systems should be able to express the following:
  1. **Decentralized attributes**
     - An entity asserts that another entity has an attribute
  2. **Delegation of attribute authority**
     - An entity delegates (trusts the judgment of) another entity on an attribute
  3. **Inference of attributes**
  4. **Attribute fields**
     - Attribute may have field values e.g. age, credit limit
  5. **Attribute-based delegation of attribute authority**
     - Delegate to entities that are certified universities the authority to identify students
ABAC and TM languages

- Keynote, SPKI 1.0, X.509 cannot express 3 & 5
- SDSI 1.0, SPKI/SDSI 2.0 cannot express 4
- RT expresses all 5
Role-based Trust Management (RT)

- RT is a proposal for meeting the requirements of ABAC systems.
  - Describes an access control policy as a set of policy statements

- RT uses the notion of roles to represent attributes.
  - A role in RT defines a set of entities who are members of this role.
  - An entity is a member of a role if and only if it has the attribute identified by the role.
Role-based Trust Management (2)

- RT combines the strengths of RBAC and trust management (TM) systems.
  - From RBAC:
    - notion of roles
  - From TM:
    - principles of managing distributed authority through the use of credentials.
    - notation denoting relationships between those authorities.
RT Framework

- A family of TM languages
  - Simplest, $RT_0$, has no parameterization, thresholds, or separation of duty.
  - More complex members:
    - $RT_1$, $RT_2$, $RT^T$, $RT^D$
RT₀ Syntax

- **A, B, D:** principals
- **r, r₁, r₂:** role names
- **A.r:** a role (a principal + a role name)

Four types of credentials:

- **A.r ← D**  
  Role A.r contains principal D as a member

- **A.r ← B.r₁**  
  A.r contains role B.r₁ as a subset

- **A.r ← A.r₁.r₂**  
  A.r ⊇ B.r₂ for each B in A.r₁

- **A.r ← A₁.r₁ ∩ A₂.r₂**  
  A.r contains the intersection

More complex versions have parameters (RT₁), constraints (RT₃), and can model thresholds and separation of duty (RT₇)
Expressive Features (I)

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
    \[ \text{EPub.student} \leftarrow \text{EPub.university.stuID} \]

- **Role**: set of entities
- **Credentials**: define role membership
Expressive Features (II)

V. Conjunction

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

VI. Attributes with fields

- \[ \text{StateU.stuID (name=.., program=.., ...)} \leftarrow \text{Alice} \]
- \[ \text{EPub.access} \leftarrow \text{StateU.stuID(program=“graduate”)} \]
Languages in the RT Framework

RT₀: Decentralized Roles

RT₁: Parameterized Roles

RT₁⁺: for Separation of Duties

RT₂: Logical Objects

RT₂⁺: structured resources

RT₀, RT₁, RT₂, RT₁⁺, and RT₂⁺ can be used (either together or separately) with any of the five base languages: RT₀, RT₁, RT₂, RT₁⁺, and RT₂⁺
Thank You!