Inference System for Role-Based Trust Management Languages

Anna Felkner, Warsaw University of Technology, Research and Academic Computer Network (25.09.2007, prof. Krzysztof Sacha, Warsaw University of Technology)

Abstract

Role-based Trust management (RT) languages are used for representing policies and credentials in decentralized, distributed access control systems. They combine trust management and Role-Based Access Control features. The subject of this paper is an operational semantics for the family of RT languages, in which credentials can be established using a simple set of inference rules. A credential provides information about the keys, rights and qualifications from one or more trusted authorities. The inference system applied not only to the basic language of the Role-based Trust management family (RT₀), but also to other members of the family up to RT⁵, which is much more sophisticated. RT⁵ provides manifold roles and role-product operators to express threshold and separation-of-duty policies. A manifold role defines sets of entities whose cooperation satisfies the manifold role. It enables to express such a condition, which need more than one member of a role to effectively fulfil the particular task.

1. Introduction

Traditional access control models, like Mandatory Access Control (MAC), Discretionary Access Control (DAC) and Role-Based Access Control (RBAC), make authorization decisions based on the identity, or the role of the requester, who must be known to the resource owner. The most flexible of those models is RBAC [16, 9, 10], which groups the access rights by the role name and grants access to a resource to those users only, who are assigned to a particular role. This type of access control is often used in enterprise environments. It works well in a centralized system.

Problem arises in large-scale decentralized, distributed and open systems, where the identity of the users is not known in advance and the set of users can change dynamically.

To overcome the drawbacks of traditional access control models, trust management models have been proposed [1-5, 14], as an approach to make access decisions in decentralized and distributed systems. Trust management is a specific kind of access control, in which decisions are based on credentials issued by multiple principal. A credential is an attestation of qualification, competence or authority, issued to an individual by a third party. Examples of credentials in real life include identification documents, social security cards, driver's licenses, membership cards, academic diplomas, certifications, passwords and user names, keys, etc. A credential in a computer system can be a digitally signed document.

The potential and flexibility of trust management approach stems from the possibility of delegation: a principal may transfer limited authority over a resource to other principals. Such a delegation is implemented by means of an appropriate credential. This way, a set of credentials defines the access control strategy and allows of deciding on who is authorized to access a resource, and who is not.

To define a trust management system, a language is needed for describing entities (principals and requesters), credentials and roles, which the entities play in the system. Responding to this need, a family of Role-based Trust management languages has been introduced in [13, 3, 6]. The family consists of five languages: RT₀, RT₁, RT₂, RT⁵, and RT⁵, with increasing expressive power and complexity. All the languages have a precise syntax definition, but a satisfactory semantics definition is still missing. A set-theoretic semantics, which defines the meaning of a set of credentials as a function from the set of roles into the power set of entities, has been originally defined for RT₀ by Li et al. in [14] and modified by Gorla et al. in [11]. In [6, 8] we defined a semantics, which applies not only to RT₀ but also to other members of the family up to RT⁵. The semantics has been defined as a relation between roles and set of sets of entities. Members of such a set must cooperate in order to satisfy the role. This way, our definition covers not only the basic RT₀ language, but also the more powerful RT⁵, which provides the notion of manifold roles and is able to
express structure of threshold and separation-of-duty policies.

This paper presents another type of semantics. It is operational semantics for the RT family languages in which credentials can be established using a simple set of inference rules. By means of simple but realistic examples, the expressiveness and usability of this system is demonstrated.

The paper is structured as follows. The family of Role-based Trust management languages is shortly described in Section 2 (including examples). Section 3, which is the core part of this paper, presents the inference system of the family of RT languages up to RT. Final remarks are given in Conclusions.

2. Role-based Trust management languages

Role-based Trust management languages are used for representing policies in distributed authorization systems. The languages combine features from trust management and Role-Based Access Control, and define a family of models of trust management systems with varying expressiveness and complexity. Trust management is a kind of a distributed access control, which is a distinct and important component of a security system in network services. RBAC is the most flexible type of access control policies, often used in enterprise environments. It uses the notion of a role to control which users can access the particular system resources. Access rights are grouped by the role name and access to a resource is limited to those users who are assigned to a particular role.

All the RT languages use the notion of a role to define sets of entities, which are members of this role. Entities in RT languages correspond to users in RBAC. Roles in RT can represent both - roles and permissions from RBAC.

RT is the core language of RT family, described in detail in [15]. All the subsequent languages add new features to RT. A summary of the features supported by particular RT languages is shown in Table 1.

RT adds to RT parameterized roles, each of which can be described by a set of attributes. The attributes are typed, and can be integers, enumerations, floating point values, dates and times.

RT further extends RT to provide a notion of logical objects which can group logically related objects together so that permissions about them can be assigned together.

RT provides manifold roles and role-product operators, which can express threshold and separation-of-duty policies. A manifold role is a role that can be satisfied by a set of cooperating entities, e.g. in a requirement that two different bank cashiers must authorize a transaction. A single-element role can be treated as a special case of a manifold role, whose set of cooperating entities is the singleton.

Threshold policies require a specified minimum number of entities to agree on some fact. The concept of separation-of-duty is related to threshold policies. In the case of a separation-of-duty policy, entities from different sets must agree before access is granted. It means that some transactions cannot be completed by a single entity. This implies that no single entity can have all the access rights required to complete such a transaction.

RT provides mechanism to describe delegation of rights and role activations, which can express selective use of capacities and delegation of these capacities. The inference system of this language is not covered in this paper.

RT and RT can be used, together or separately, with each of RT, RT, and RT.

There are few more languages which are based on RT, but they are not so useful from the author's point of view, so they will not be described.

A more detailed overview of the Role-based Trust management family framework can be found in [13].

<table>
<thead>
<tr>
<th>Supported features of RT languages</th>
<th>Supported features</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT language</td>
<td></td>
</tr>
<tr>
<td>RT</td>
<td>– localized authorities for roles, – role hierarchies, – delegation of authority over roles, – attribute based delegation of authority, – role intersections.</td>
</tr>
<tr>
<td>RT</td>
<td>logical objects.</td>
</tr>
<tr>
<td>RT</td>
<td>features of RT plus: – selective use of role membership, – dynamic credential delegation.</td>
</tr>
</tbody>
</table>

Tab. 1.
2.1 The Syntax of RT Languages

Basic elements of RT languages are entities, role names, roles and credentials. **Entities** represent principals that can define roles and issue credentials, and requesters that can make requests to access resources. An entity can be identified by a user account in a computer system or a public key. **Role names** represent permissions that can be issued by entities to other entities or groups of entities. **Roles** represent sets of entities that have permissions issued by particular issuers. A role is described as a pair composed of an entity and a role name. **Credentials** define roles by pointing a new member of the role or by delegating authority to the members of other roles.

The syntax of Role-based Trust management family is shown in Table 2.

<table>
<thead>
<tr>
<th>Language element</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity name</td>
<td>$A, B, C, D, E, X \ldots \in \varepsilon$</td>
</tr>
<tr>
<td>Role name</td>
<td>$r, s, t, \ldots \in \mathbb{R}$</td>
</tr>
<tr>
<td>Role</td>
<td>$A.r, B.s, C.t, \ldots \in \varepsilon \times \mathbb{R}$</td>
</tr>
<tr>
<td>Role expression</td>
<td>$e := B \mid B.s \mid B.s \cap C.t \mid B.s \odot C.t \mid B.s \odot C.t$</td>
</tr>
<tr>
<td>Credential</td>
<td>$c := A.r \leftarrow e$</td>
</tr>
</tbody>
</table>

Role expressions and credentials above should be interpreted in the following way:

- $A.r \leftarrow B$ - simple member - entity B is a member of role A.r.
- $A.r \leftarrow B.s$ - simple inclusion – role A.r includes (all members of) role B.s.
- $A.r \leftarrow B.s.t$ - linking inclusion – role A.r includes role C.t for every C that is a member of role B.s. The expression B.s.t is called a linked role.
- $A.r \leftarrow B.s \cap C.t$ - intersection inclusion – role A.r includes every principal who is a member of both role B.s and role C.t. The expression $B.s \cap C.t$ is called an intersection role.
- $A.r \leftarrow B.s \odot C.t$ - role A.r includes one member of role B.s and one member of role C.t. This allows expressing the structure of a threshold.
- $A.r \leftarrow B.s \odot C.t$ - role A.r includes one member of role B.s and one member of role C.t, but those members of roles have to be different. It enables to express separation-of-duty policies.

**Example 1 (RT<sub>5</sub>)**
A person has the right to attend a lecture given at a university $U$, when he or she is a student registered to a faculty of this university. To be able to fulfill the role of a faculty, an organization ought to be a division of the university and should conduct research activities. John is a student registered to F, which is a division of U, and which conducts research activities. The following credentials prove that John has the right to attend a lecture:

$U.lecture \leftarrow U.faculty.student$ (1)
$U.faculty \leftarrow U.division \cap U.research$ (2)
$U.division \leftarrow F$ (3)
$U.research \leftarrow F$ (4)
$F.student \leftarrow John$ (5)

**Example 2 (RT<sub>7</sub>)**
The following example has been adopted from [12]. A bank B has three roles: manager, cashier and auditor. Security policy of the bank requires an approval of certain transactions from a manager, two cashiers, and an auditor. The two cashiers must be different. However, a manager who is also a cashier can serve as one of the two cashiers. The auditor must be different from the other parties in the transaction.

Such a policy can be described using the following credentials:

- $B.twoCashiers \leftarrow B.cashier \odot B.cashier$ (6)
- $B.mgrCashiers \leftarrow B.manager \odot B.twoCashiers$ (7)
- $B.approval \leftarrow B.auditor \odot B.mgrCashiers$ (8)

Now, assume that the following credentials have been added:

- $B.cashier \leftarrow Alice$ (9)
- $B.cashier \leftarrow Doris$ (10)
- $B.manager \leftarrow Doris$ (11)
- $B.auditor \leftarrow Kate$ (12)

Then one can conclude that, according to the policy of B, the following set of entities can cooperatively approve a transaction: {Alice, Doris, Kate}.
3. Inference system of Role-based Trust management family

We now provide an operational semantics for RT family languages up to RT via a very intuitive inference system. The six kinds of credentials described in section 2.1 are handled by the six rules in Table 3, where judgment \( \mathcal{P} > c \) should be read as: “using the credentials in \( \mathcal{P} \), we can infer the credential \( c \).” The rules should be self-explanatory.

Tab. 3.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (W_1) )</td>
<td>( c \in \mathcal{P} )  ( \mathcal{P} &gt; c )</td>
</tr>
<tr>
<td>( (W_2) )</td>
<td>( \mathcal{P} &gt; A.r \leftarrow B.s \quad \mathcal{P} &gt; B.s \leftarrow X )  ( \mathcal{P} &gt; A.r \leftarrow X )</td>
</tr>
<tr>
<td>( (W_3) )</td>
<td>( \mathcal{P} &gt; A.r \leftarrow B.s \land C.t \quad \mathcal{P} &gt; B.s \leftarrow C \quad \mathcal{P} &gt; C.t \leftarrow X )  ( \mathcal{P} &gt; A.r \leftarrow X )</td>
</tr>
<tr>
<td>( (W_4) )</td>
<td>( \mathcal{P} &gt; A.r \leftarrow B.s \land C.t \quad \mathcal{P} &gt; B.s \leftarrow X \quad \mathcal{P} &gt; C.t \leftarrow X )  ( \mathcal{P} &gt; A.r \leftarrow X )</td>
</tr>
<tr>
<td>( (W_5) )</td>
<td>( \mathcal{P} &gt; A.r \leftarrow B.s \lor C.t \quad \mathcal{P} &gt; B.s \leftarrow X \quad \mathcal{P} &gt; C.t \leftarrow Y )  ( \mathcal{P} &gt; A.r \leftarrow X \cup Y )</td>
</tr>
<tr>
<td>( (W_6) )</td>
<td>( \mathcal{P} &gt; A.r \leftarrow B.s \lor C.t \quad \mathcal{P} &gt; B.s \leftarrow X \quad \mathcal{P} &gt; C.t \leftarrow Y \quad X \cap Y = \emptyset )  ( \mathcal{P} &gt; A.r \leftarrow X \cup Y )</td>
</tr>
</tbody>
</table>

3.1 Examples

Example 3 (Inference system for Example 1)

We use the inference system to formally derive a credential authorising John to attend a lecture.

Using credentials \( (1)-(5) \) according to rule \( (W_1) \) we can infer:

1) \( u.lecture \leftarrow u.faculty.student \in \mathcal{P} \)  \( \mathcal{P} > u.lecture \leftarrow u.faculty.student \)
2) \( u.faculty \leftarrow u.division \cup u.research \in \mathcal{P} \)  \( \mathcal{P} > u.faculty \leftarrow u.division \cup u.research \)
3) \( u.division \leftarrow F \in \mathcal{P} \)  \( \mathcal{P} > u.division \leftarrow F \)
4) \( u.research \leftarrow F \in \mathcal{P} \)  \( \mathcal{P} > u.research \leftarrow F \)
5) \( f.student \leftarrow john \in \mathcal{P} \)  \( \mathcal{P} > f.student \leftarrow john \)

Thus using credential \( (2), (3), (4) \) and rule \( (W_1) \) we infer:

6) \( \mathcal{P} > (2) \quad \mathcal{P} > (3) \quad \mathcal{P} > (4) \)  \( \mathcal{P} > u.faculty \leftarrow F \)

In next step we use it and additionally credentials \( (1), (5) \) and rule \( (W_1) \):

7) \( \mathcal{P} > (1) \quad \mathcal{P} > u.faculty \leftarrow F \quad \mathcal{P} > (5) \)  \( \mathcal{P} > u.lecture \leftarrow john \)

which proves that John has the right to attend a lecture.

Example 4 (Inference system for Example 2)

We use the inference system to formally derive a set of entities which can cooperatively approve a transaction.

Using credentials \( (6)-(12) \) according to rule \( (W_1) \) we can infer:

1) \( B.twoCashiers \leftarrow B.cashier \otimes B.cashier \in \mathcal{P} \)  \( \mathcal{P} > B.twoCashiers \leftarrow B.cashier \otimes B.cashier \)
2) \( B.mgrCashiers \leftarrow B.manager \otimes B.twoCashiers \in \mathcal{P} \)  \( \mathcal{P} > B.mgrCashiers \leftarrow B.manager \otimes B-twoCashiers \)
3) \( B.approval \leftarrow B.auditor \otimes B.managerCahiers \in \mathcal{P} \)  \( \mathcal{P} > B.approval \leftarrow B.auditor \otimes B.managerCashiers \)
4) \( B.cashier \leftarrow Alice \in \mathcal{P} \)  \( \mathcal{P} > B.cashier \leftarrow Alice \)
Than using credentials (6), (9), (10) and rule (W₂) we infer:

\[
\frac{P > (6) \quad P > (9) \quad P > (10)}{P > B.\text{twoCashiers} \leftarrow \{\text{Alice, Doris}\}}
\]

In next step we use it and additionally credentials (7), (11) and rule (W₂):

\[
\frac{P > (7) \quad P > B.\text{twoCashiers} \leftarrow \{\text{Alice, Doris}\} \quad P > (11)}{P > B.\text{mgrCashiers} \leftarrow \{\text{Alice, Doris}\}}
\]

Than we use that credential and add credentials (8) and (12) and using rule (W₂) we can infer:

\[
\frac{P > (8) \quad P > B.\text{mgrCashiers} \leftarrow \{\text{Alice, Doris}\} \quad P > (12)}{P > B.\text{approv} \leftarrow \{\text{Alice, Doris, Kate}\}}
\]

what shows that the set of entities that can cooperatively approves a transaction is: \{Alice, Doris, Kate\}.

Soundness and completeness of inference system was shown in [7], what proves that the inference system provides an alternative way of presenting the semantics of RT₁.

4. Conclusion

This paper deals with modelling of trust management systems in decentralized and distributed environments. The modelling framework is a family of Role-based Trust management languages. The core part of the paper is a definition of formal operational semantics of the family of Role-based Trust management up to RT₁, in which credentials can be established using a simple set of inference rules.

Using RT₁ one can define credentials stating that an action is allowed if it gets approved by members of more than one role. This improves the possibility of defining complex trust management models in real environment.

Inference systems presented in this paper are simple, but well-founded theoretically. It turns out to be fundamental mainly in large-scale distributed systems, where users have only partial view of their execution context.

Bibliography


Author:

M. Sc. Anna Felkner
Warsaw Univ. of Technology
Institute of Control & Computation Engineering
Nowowiejska 15/19
00 - 665 Warsaw
phone: +48 22 825 09 95
fax: +48 22 825 37 19
email: anna.felkner@gmail.com