A Trust-based Access Control Model for Virtual Organizations

Aizhong Lin, Erik Vullings, and James Dalziel
MELCOE, Macquarie University
Sydney, NSW 2109, Australia
{alin, erik.vullings, james}@melcoe.mq.edu.au

Abstract

Virtual organizations normally use role-based access control mechanisms to assign permissions that allow users to access resources or services. Role-based access control mechanisms, however, have three limitations. First, as only one type of trust relationship --- resource trusts role --- exists in the mechanisms, more trust relationships that support more types of access controls in virtual organizations cannot be established. Second, as roles are created in and limited to specific collaborative work places, the permissions only take effects in the local work places, and no global permissions can be set up. Finally, the attributes of users or groups, as important resources, have no control in the mechanisms. In other words, those attributes cannot be released to any other user or group. To overcome these limitations, our research provides a trust-based access control model for virtual organizations. This paper presents the model, algorithm, implementations, and experimental results.

1. Introduction

A Virtual Organization (VO), as defined in [1], is “a temporary or permanent coalition of geographically dispersed individuals, groups, organizational units or entire organizations that pool resources, capabilities and information to achieve common objectives”. A VO provides functions such as “upload” and “download” to support collaborations, and mechanisms such as “permit” and “deny” to realize resource and service (R&S) access controls. Therefore, access control mechanisms play the important roles on virtual organizations.

Virtual organizations normally use role-based access control mechanisms [2] [3] to assign permissions that allow users to access resources. A role-based control mechanism has three limitations: (1) it supports only a one-way trust relationship --- a resource trusts a role --- that ignores many other possible trust relationships between VO other parties; (2) its trust relationship takes only effects in a local collaborative work space but does not support global trust relationships; and (3) it does not support the access of an important resource type --- users’ attributes. These limitations cause the low flexibility and efficiency when users access R&Ss.

Our research provides a new access control mechanism --- trust-based access control mechanism (TbACM) --- for virtual organizations. Firstly, in TbACM, a number of possible trust relationships between VO parties are supported. Each trust relationship is leveled. Different level of trust relationship will be mapped to different set of access behaviors. Secondly, global trust relationships are supported. A user’s behaviors are controlled by the combination of global trust relationships and local trust relationships. And finally, the attributes of parties in virtual organizations can be accessed by other trust parties based on the setting of trust levels or policies.

This paper describes the trust-based access control mechanism designed for virtual organizations. Firstly, it introduces the basic concepts and properties of “trust”. In second section, a complete trust-based access control model is presented in detail. An implementation of the model is described in the third section. Then we describe the TbACM algorithm. Finally, we give the implementation and experimental results of VOs exploiting TbACM.

2. Trust

Trust, as defined in dictionaries, is “the firm reliance on the integrity, ability, or character of a person or thing”. It describes a specific relationship between parties. Trust relationships are normally established between parties after they have communicated and collaborated for a certain time. For example, that Alice trusts Bob means that “from a certain time of observation, Alice understands and accepts most behaviors of Bob”. Once a trust
relationship is established in two parties, it will strongly influence the future behaviors of their interactions. When parties trust each other, they like to share services and resources in a certain level. For instance, if Alice trusts Bob, Bob can borrow money from Alice.

Formally, the value $T_{p_1p_2}$ that determines how party $p_1$ trusts party $p_2$ after they have interacted for $n$ times is calculated by a formula:

$$T_{p_1p_2} = \frac{\sum_{i=1}^{n} [o_i(B_i) - e_i(B_i)]}{t}$$

Where:
- $T_{p_1p_2}$ is the trust value reflects how $p_1$ trusts $p_2$
- $t$ is the time of their previous $n$ times interactions
- $B_i$ is a set of behaviors in $i^{th}$ interaction
- $o_i$ is the observed value of the behaviors in $i^{th}$ interaction
- $e_i$ is the expected value of the behaviors in $i^{th}$ interaction

Trust relationships between parties have some noticeable properties which will considerably affect the generation of trust models:

1. First, base level trusts can be established when users show their certificates signed by authorities. Such users, because their behaviors are traceable, are trustable in a base level.
2. Second, high level trusts are subjective and individual judgments between parties. Even though Alice trusts Bob does not mean that Bob will be trusted by other persons. Therefore, when forming the trust relationships in a group of people, each person has its own judgment to each of other persons in the group.
3. Third, a trust relationship has only one direction, i.e., if Alice trusts party B does not mean that party B also trusts party A. Hence, from a trust link from party A to party B, we can not automatically get the link from party B to party A.
4. Next, trust relationships are leveled. In different situations, trust relationships may be changed. For example, party A trusts party B when sharing some documents, however, party A does not trust party B when sharing money.
5. Furthermore, trust relationships between parties can be strengthened or weakened when more interactions happened between the parties.
6. Last, a trust relationship is easier to destroy than to establish.

3. The Trust-based Access Control Model

This research provides a trust-based access control mechanism for virtual organizations. Firstly, we define three categories of parties in a VO:
- **user**: an individual, a user has identity attributes, can access R&Ss
- **group**: a group of users, a group has identity attributes and a leader
- **vroom**: a virtual room or workspace that maintains groups and R&Ss for a specific goal, a vroom has identity attributes and an owner

As our VO prototype is built on the Shibboleth®-based technology [6], two additional parties need to be introduced:
- **IdP**: a Shibboleth Identity Provider (IdP), which maintains, verifies and asserts user attributes for a community or organization, e.g. a university
- **SP**: a Shibboleth Service Provider (SP), which offers services to users that belong to an IdP, e.g. an institutional repository or a Wiki

3.1. Possible Trust Relationships

Trust relationships can be established between these parties. Table 1 shows the possibilities of trust relationships between parties, in which “x” means that trust is possible. In this table, we define that the column parties trust the row parties.

<table>
<thead>
<tr>
<th>user</th>
<th>group</th>
<th>vroom</th>
<th>IdP*</th>
<th>SP*</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>group</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>vroom</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>IdP*</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>SP*</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

* used in our VO prototype

3.2. Resource Access Behaviors

That more trust relationships are provided in VOs is used to control parties to access R&Ss. The possible access behaviors in a VO could be a small set or a large set depending on the purpose of the VO system. In our VO prototype, we provide the following set of R&Ss accessing behaviors:
- **search**: find all R&Ss by keywords
3.3. Trust Levels

According to the trust properties, trust is leveled. Different level of trust results in different permissions to access resources. In this model, it has six trust levels and they are mapped to six permitted behavior lists:

- level_0: no access
- level_1: can search and list R&Ss
- level_2: can search, list, and read R&Ss
- level_3: can search, list, read, execute, and modify R&Ss
- level_4: can search, list, read, execute, modify, and create R&Ss
- level_5: can search, list, read, execute, modify, create, grant, delete R&Ss

3.4. Trust Policy

It is more flexible to specify the permissions from trust relationships to access behavior lists in a trust policy file rather than the trust levels. The trust policy file is a XML-based file containing a collection of trust rules wrapped by the tags <TrustPolicy> and <Rule>.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<TrustPolicy xmlns="mams:tvo:trust:tp:1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="mams:tvo:trust:tp:1.0 trust-tp-1.0.xsd">
  <Rule>
    <Behavior name="search">
      <Permission access="permit"/>
    </Behavior>
    <Behavior name="list">
      <Permission access="permit"/>
    </Behavior>
    <Behavior name="read">
      <Permission access="permit"/>
    </Behavior>
    <Behavior name="execute">
      <Permission access="deny"/>
    </Behavior>
    <Behavior name="update">
      <Permission access="permit"/>
    </Behavior>
    <Behavior name="create">
      <Permission access="deny"/>
    </Behavior>
    <Behavior name="grant">
      <Permission access="deny"/>
    </Behavior>
    <Behavior name="delete">
      <Permission access="deny"/>
    </Behavior>
  </Rule>
</TrustPolicy>
```

3.5. Trust-based Access

Following, we define the possible trust relationships in Table 2 in details based on the description of Table 1.

<table>
<thead>
<tr>
<th>Trust relationship</th>
<th>Explanations</th>
</tr>
</thead>
</table>
| user A trusts user B       | 1) the attributes of user A can be released to user B
 |                            | 2) all R&Ss owned by user A can be accessed by user B                      |
| user A trusts group B      | 1) the attributes of user A can be released to the leader of group B
 |                            | 2) all R&Ss owned by user A can be accessed by the leader of group B        |
 |                            | 3) the leader of group B has the rights to control the R&Ss based on the trust level or trust policy set by user A |
| user A trusts SP B*        | The attributes of user A can be released to SP B based on attribute release policies |
| user A trusts vroom B      | 1) the attributes of user A can be released to the owner of vroom B
 |                            | 2) all R&Ss owned by user A can be accessed by the owner of vroom B        |
 |                            | 3) the owner of vroom B has the rights to control the R&Ss based on the trust level or trust policy set by user A |
| group A trusts user B      | 1) the attributes of group A can be released to user B
 |                            | 2) the R&Ss owned by group A can be accessed by user B                       |
| group A trusts group B     | 1) the attributes of group A can be released to the leader of group B
 |                            | 2) the R&Ss owned by group A can be accessed by the leader of group B     |
A global trust is a trust relationship established in the creation of a party in the VO system. It is used to control any behavior of the other parties related to the trust no matter where (which VRoom) the behavior happens.

3.7. “Goal-based” Trust

A goal-based trust is a local trust relationship established in a VRoom for determining the permissions of users accessing resources in a VRoom. When parties start to live in the VRoom, their access rights are controlled by not only global trust relationships but also the local trust relationships. If the VRoom is removed, all the goal-based (called “local” for simple) trust relationships in this VRoom are destroyed automatically.

3.8. Trust Combination

A trust-based control function maps the trust relationships to a behavior list of each R&S. If an R&S is controlled by a global trust relationship as well as a goal-based trust relationship, the final behavior list is the logic “or” operation of the behavior list (a set of behavior) based on the global trust relationship and the behavior list (a set of behavior) based on the goal-based trust relationship.

An example will help to understand the trust combination. Suppose we have two parties: user A and user B. User A sets a global level_2 trust relationship linking to user B. The trust-based access control function maps the level_2 trust to the access list \{list, read/execute\}. Meanwhile, in a VRoom C, user A sets a goal-based level_5 trust relationships to user B. Based on the level_5 trust, user B can get another access list \{list, read/execute, modify, write, propagate\} to the R&Ss of user A. Then the final access
functions user B get from the global level 2 trust and the goal-based level 5 trust is in the set {list, read/execute} or {list, read/execute, modify, write, propagate} = {list, read/execute, modify, write, propagate}.

4. Trust-based Access Control Algorithm

TbACM is used in VOs to determine the access permissions when users manipulate resources. An algorithm is designed in TbACM. The algorithm generates permitted access behavior list for specific resource r, user p, and VRoom vr. Whenever a user p intends to access a resource r in a VRoom vr, the permitted behavior list tells the VRoom which behavior is permitted. For example, if a user “Alice” is going to download a document “Australian National Defense Guide.pdf” in a VRoom vr, the vr starts up the TbACM to get the permitted behavior list of Alice to “Australian National Defense Guide.pdf” in vr, and if the behavior “download” is in the permitted behavior list, the resource is released to Alice, otherwise, Alice cannot download “Australian National Defense Guide.pdf”.

```c
//r is a resource id, p is the party id to access the resource r, vr is the id of a VRoom
METHOD PermittedBehaviorList GeneratePermittedBehaviorList(r, p, vr)
BEGIN
    PermittedBehaviorList pbl = {};
    Party owner = getResourceOwner(r);

    //generate access list for global trust relationships
    TrustRelationshipSet TRS = getGlobalTrustRelationships(owner, p);
    PermittedBehaviorList tpbl = {};
    FOR_EACH(tr ∈ TRS)
    BEGIN
        TrustPolicy tp = getTrustPolicy(tr);
        IF(tp ≠ NULL)
        BEGIN
            tpbl = tpbl ∪ getPermittedBehaviorList(tp);
        END
        ELSE
        BEGIN
            INT level = getTrustLevel(tr);
            tpbl = tpbl ∪ getPermittedBehaviorList(level);
        END
    END
    pbl = pbl ∪ tpbl;

    //generate access list for local trust relationships
    TRS = getLocalTrustRelationships(owner, p, vr);
    tpbl = {};
    FOR_EACH(tr ∈ TRS)
    BEGIN
        TrustPolicy tp = getTrustPolicy(tr);
        IF(tp ≠ NULL)
        BEGIN
            tpbl = tpbl ∪ getPermittedBehaviorList(tp);
        END
        ELSE
        BEGIN
            INT level = getTrustLevel(tr);
            tpbl = tpbl ∪ getPermittedBehaviorList(level);
        END
    END
    pbl = pbl ∪ tpbl;
RETURN pbl;
END
```

Above is the trust-based access control algorithm. The input parameters are a resource identifier r, a user identifier p, and a VRoom identifier vr. In the algorithm:

1. Set to be returned permitted behavior list to empty
2. A party owner who is the owner of the resource is got based on the resource identifier
3. Global trust relationships set by the owner to p is obtained in a trust relationship set
4. For each global trust relationship, try to get the policy file. If it is found, obtaining a permitted behavior list based on the trust policy. If it is not found, obtaining a permitted behavior list based on the trust level. Combining the permitted behavior list with the permitted behavior list obtained previously
5. Repeat above step until all global trust relationships are reached
6. Combining the permitted behavior list to the to be returned behavior list
7. Local trust relationships set by the owner to p is obtained in a trust relationship set
8. For each local trust relationship, try to get the policy file. If it is found, obtaining a permitted behavior list based on the trust policy. If it is not found, obtaining a permitted behavior list based on the trust level. Combining the permitted behavior list with the permitted behavior list obtained previously
9. Repeat above step until all local trust relationships are reached
10. Combining the permitted behavior list to the to be returned behavior list
11. Return the to be returned permitted behavior list

5. The Implementation and Experimental Results

The component implemented to fulfill the trust-based access control model includes a record to store trust relationships, a configuration function to set up global trust relationships, a configuration function to set up local trust relationships, and a function to check the permitted behavior list for each resource.

5.1. Trust Record

The trust record is able to store global trust relationships as well as local trust relationships. As described in Figure x, the record has eight attributes:

- **party_1**: The identifier of the party 1 who is a truster
- **party_1_type**: The type of the party 1. The type could be a user, a group, a VRoom, an IdP, or a SP
- **party_2**: The identifier of the party 2 who is a trustee
- **party_2_type**: The type of the party 2. The type could be a user, a group, a VRoom, an IdP, or a SP
- **scope**: The scope of the trust relationship. Its value could “global” or an identifier of a Vroom
- **trust_level**: The trust level of the relationship
- **trust_policy_name**: The file name that specifies the trust policy
- **description**: The description of the trust relationship

<table>
<thead>
<tr>
<th>Trust Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. party_1</td>
</tr>
<tr>
<td>2. party_1_type</td>
</tr>
<tr>
<td>3. party_2</td>
</tr>
<tr>
<td>4. party_2_type</td>
</tr>
<tr>
<td>5. scope</td>
</tr>
<tr>
<td>6. trust_level</td>
</tr>
<tr>
<td>7. trust_policy_name</td>
</tr>
<tr>
<td>8. description</td>
</tr>
</tbody>
</table>

Figure 1: The trust record

5.2. Set up Global Trust Relationships

When a party --- a user, a group, or a VRoom --- is created in a VO and signs on the VO, the party has a default permission to set up its global trust relationships. As described in the trust model in section 3, a user can trust other users, groups, VRooms, and SPs; a group can trust users, other groups, VRooms, and SPs; a VRoom can trust users, groups, other VRooms, IdPs, and SPs.

Figure 2 shows the interface of the function to set up global trust relationships. This function is used by a user who has signed on the VO. In the interface, a user can choose each of other parties and set up the trust level or trust policy. After all trust relationships are configured, they are saved in a trust repository of the VO.

Figure 2: The interface of the function to set up global trust relationships
5.3. Set up Local Trust Relationships

When a party --- a user or a group --- open a VRoom, it has a default permissions to set up local trust relationships with other parties in the VRoom.

![Figure 3: The interface of the function to set up local trust relationships in a VRoom](image1)

Figure 3 shows the interface of the function to set up local relationships. This function is used by a user who works in a VRoom. In the interface, a user can choose each of other parties and set up the trust level or trust policy. After all trust relationships are configured, they are saved in the trust repository of the VO.

5.4. Check Permitted Behaviors List

When a user lists all resources in a VRoom, there is a permitted behavior list for the user and each resource. In Figure 4, we can check that the signed user can do "" for the resource “doc_001” but can only “search, list, and read” the resource “test_001”.

![Figure 4: The interface shows the permitted behavior list for each resource](image2)

6. Related Work

Trust plays an important role in secure systems and critical information management and access. Grandison and Sloman [4] introduced the concepts and properties of trust, and trust being employed in Internet applications. Jøsang etc. [5] stressed the trust requirements in identity management. They indicated that “identity management requires an integrated and often complex infrastructure where all involved parties must be trusted for specific purposes depending on their role”. In this research, role-based trust was main trust, global trust is not discussed. Shibboleth [6] and Liberty [7] implemented the components that support the establishment and management of based trust relationships between identity providers and service providers. In their trust models, other possible relationships except for the trust relationships between IdPs and SPs have not been concerned.

Trust is also a major research theme in virtual organizations. Handy [8] asserts that interactions between parties in VOs require trust being built and maintained online. However, he did not propose how to build and maintain online trust. Holland and Lockett [9] proposed a business trust framework being used in virtual organizations to support the formation of VOs. Meanwhile, formal descriptions about modeling trust relationships in distributed environment are given in [10]. As Grid Computing becomes more and more popular, trust is a main issue to be paid
attention to. Matthews etc. [11] presented the requirements of trust on the Grid.

7. Future Work

This research provides a trust-based access control mechanism for virtual organizations to make more trust relationships being available in VOs and make VO access controls becoming more flexible and efficient. A VO prototype, which contains the component of the trust-based access control mechanism, is implemented. More attributes about its usability will be obtained after the VO prototype is employed in MAMS project [12].

Future research will focus on two directions. On the one hand, we will focus on the establishment and management of trust relationship between VO parties. In current stage, this VO prototype supports only that all trust relationships are established and modified by parties. In the future, a trust revision model will be provided and the VO system will be built with an autonomous function based on the trust revision model to modify the trust relationships between parties after they have a period of interactions. On the other hand, we will focus on the design and implementation of trust virtual organizations (TVO). A TVO is a VO in which all parties can be trusted in a certain level.

Acknowledgement

This work is a part of the MAMS [x] Meta Access Management System project, and funded by the Australian Department of Education, Science and Training as part of Backing Australia’s Ability program.

References


