TRUST-ORBAC: A Trust Access Control Model in Multi-Organization Environments

Khalifa Toumi¹, César Andrés², Ana Cavalli¹

¹ IT/ TELECOM & Management SudParis, EVRY, F-91011
{Khalifa.Toumi, Ana.Cavalli}@it-sudparis.eu
² Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid, Spain
c.andres@fdi.ucm.es

Abstract. Access control in Multi-Organization Environment is a critical issue. Classical access control models like Role Based Access Control (RBAC) and Organization Based Access Control (ORBAC) need some improvements to be used in such environment, where the collaboration is established between organizations and not directly with the clients. In particular, some characteristics of this scenario are that the users may be unknown in advance and/or the behaviors of the users and the organization may change during the collaboration. Hence, in this context the use of trust management with an access control model is recommended.

To achieve this goal in this paper a new model called TRUST-ORBAC that adds the notion of trust management to ORBAC is presented. This approach consists in defining two dynamic trust vectors: one for the organizations and one for users which are based on different parameters such as knowledge, reputation and experience. Finally, we illustrate the use of TRUST-ORBAC with a case study.

1 Introduction

A Multi-Organization Environment (MOE) is a system composed of several organizations that collaborate, exchange data and interact among them in order to achieve a global goal. However, this collaboration is not perfect since different problems can appear such as malicious use of resources, disclosure of data or bad services [17]. One of the major security issues for any organization is the Access Control. This aims to protect the use of the resources by the definition of rules that determine whether a user can perform an action or can access to a resource. Different Access Control models such as Mandatory Access Control (MAC), Discretionary Access Control (DAC), Role Based Access Control (RBAC) and Organization Based Access Control (ORBAC) have been defined in the literature [6]. The latter is emerging as one of the most efficient model that can be used in a MOE [4,7,6]. The main benefits of ORBAC are: a) the use of a new concept organization, b) the usability of administration, c) the variety of different types of access (permission, prohibition and obligation) and d) the definition of a high abstraction level.

The improvement of ORBAC model is a highly active area of research. Therefore, different extensions of ORBAC have been proposed [4,7,6] in order to reap its benefits
Our proposal will be also based on the important concept of ORBAC model. Fundamentally, this issue is related to the establishment of a trust valuation between different entities of this system (the resource provider and the requester). During the last ten years, the idea of integrating a trust management system into access control models to enhance the security level of an organization has been widely studied \cite{2,4,15,17}.

Let us note that trust models are widely accepted as an innovative solution to guarantee a certain level of security between two entities belonging to a distributed system \cite{10,17}. When we refer to trust we need to introduce the notion of environment. This concept depends on the peculiar characteristics that provide relevant information. However the previous solutions are not defined for the properties that MOE provides. Therefore, so far we know, there are some open issues, that motivate our work, such as:

"How do we define trust in a MOE scenario?", "How can we integrate a trust model into ORBAC?", and "How could we provide a measure of the impact of an organization behavior in the control access of its users?"

Thus, in this paper we present a framework called TRUST-ORBAC, to address these issues. TRUST-ORBAC incorporates different concepts of ORBAC \cite{4,7,6} model in order to have its level of abstraction, dynamism and the definition of organization entity.

In Figure 1 the scheme of our proposal is presented. A user sends a request to the system. Instead of having a static function (only based on the collect of attributes) that decides which roles the user is mapped, we will have a Trust module, that dynamically assigns some roles to the user. This set means that at this period of time, taking into account different trust aspects (that will be discussed in Sections 4 and 5) the subject
has been associated to these roles. Finally, this module will provide this set of roles to O\textsuperscript{R}BAC and it will look for those rules that allow/deny the request.

Moreover \textit{Trust-ORBAC} uses the concept of trust vectors \cite{15,18} to represent the trust relationships. On the one hand, it defines a user trust vector and on the other hand organization trust vector. These vectors will be based on the relevant parameters defined in the literature that are \textit{experience, reputation} and \textit{knowledge}. An additional contribution of \textit{Trust-ORBAC} is the inclusion of a new attribute: \textit{the impact of the trust evaluation of an organization on its users} that is not previously presented in the literature. This parameter is defined as the trus evaluation of an organization.

The rest of the paper is structured as follows. In Section 2 some relates works are presented. In Section 3 the definition of \textit{Trust-ORBAC} concepts and their integration in O\textsuperscript{R}BAC are introduced. Next, in Sections 4 and 5 the notion of trust and its parameters for \textit{Trust-ORBAC} are presented. In Section 6 a case study implemented with \textit{Trust-ORBAC} and two others approaches (\textit{TrustBAC} and \textit{Xena}) is detailed and discussed. Finally, in Section 7 there are reported the conclusions and some lines of future work.

2 Related works

There are several extensions of MOE that use O\textsuperscript{R}BAC \cite{3,7,6}. For instance, in \cite{3} authors presents the basic elements needed to define contract between two organizations. This proposal allows to derive automatically a set of interoperability security policies having the local one already defined. In \cite{7}, two new concepts: \textit{virtual user} and \textit{image web service} are defined in order to use O\textsuperscript{R}BAC with web service technologies. This work also presents a methodology to check, in real time, the coherence and the compatibility of the policy with the requirements. For each organization, the security policy is written using a timed automaton and is installed in a specific gateway. However, the definition and the integration of \textit{trust} on O\textsuperscript{R}BAC were not the main interest in these models, and we note that our model could be integrated into these solutions.

The definition of trust in distributed systems \cite{14,13,9,15,19} has been widely studied. The definition of trust parameters depends on the application, the environment and the requirements of the administrator, etc. For instance, in \cite{9}, authors propose a trust evaluation mechanism of contributors in the development of text documents. The notion of trust is based on the experience parameter. Its definition and evaluation is well detailed in this paper. However, the generalization of this work in MOE needs a lot of work since the different evaluation functions depends on the application \textit{'development of document'}.

In \cite{13}, a trust model based on recommendation and experience is proposed. This paper presents these concepts with more interest to the optimism, tendency, forgetting factors and their integration in a trust model. In addition, different simulations were done in order to show the simplicity and the efficiency of this model in an open networks as internet service providers and wireless network and etc.

In \cite{14,2}, authors provide a general definition of trust vector and a mathematical evaluation of it. The integration of this trust vector into RBAC model is introduced in order to define a new framework called \textit{TrustBAC}. Indeed, this vector consists of a...
tuple of the three classical parameters of trust: experience, knowledge and recommendation.

These solutions are useful, however, they do not take into account the trust level of an O-grantee and its impact on the trust of the users in the evaluation process. Therefore, their definition and evaluation of trust are not compatible with the MOE requirements. In particular, the trust level of an organization can dynamically change, but this fact is not studied in previous works.

Finally, there are some frameworks [8,4] that share the same goals that our methodology. In [8], a trust chain in a multi-organization environment is defined for RBAC. It is based on the “ISO-IEC 17799 best practices” and “maturity level” concepts. This methodology accepts the notion of collaboration with another partner. Moreover, this solution can be also integrated with ORBAC. However it is a static solution. That is, during the collaboration, the concept of role does not change, so the evaluation process does not take into account any past behavior. Finally, in [4] the framework XENA is presented. In it, the evaluation of the trust is based on sharing a set of credentials (attributes). Nevertheless, this method still has a number of shortcomings in MOE. For instance, the validity and the duration of the information may be not adequate with the requirements of the application, or the influence of the previous interactions between the requester and his organization is not considered.

3 Preliminaries

In this section, we introduce the main concepts of ORBAC for MOE that will be used in our methodology.

3.1 Multi Organization Environment (MOE)

MOE is a structure that contains at least two organizations. These organizations are O-grantor and O-grantee. The O-grantor is the participant that offers resources. These resources are accessed by users of another organization called the O-grantee. An organization may be an O-grantor and O-grantee at the same time.

A contract is an agreement between two partners to precise constraints (the resources to be shared, duration of the contract, etc). It is signed between each two participants before the collaboration [6]. The sharing of resources is based on some restriction rules that forms an interoperability security policy in order to control the access of these resources. In our model, this policy is completely parameterized by the organization. In particular, an organization is an entity that is composed of different groups of active entities, such as subjects playing some role.

3.2 Concrete and Abstract Entities

An important notion of ORBAC is the classification of the entities into concrete and abstract. A model with high level of abstraction offers a more independent security policy [6]. In an informal way, when we refer to concrete we consider single units, or elements, and while we refer to abstract these are global units, or sets. In MOE there are three concrete and their respective abstract entities.
– The first one relates subjects (employees/users) and roles. The set of all subjects will be denoted by Subjects and the roles by Roles.
– The second one relates actions and activities. The set of actions will be denoted by Actions and the activities by Activities.
– The last one relates objects and views, being Objects and Views the set of all objects and views respectively.

3.3 Access Type

The access type offered in a security policy depends on the model that we are considering. For instance, with RBAC [6], the administrator can only give an authorization. However, with OrBAC [6], four access types are offered: permission, prohibition, obligation and recommendation. How to define the access type is out of the scope of this paper. Thus, we keep it abstract, denoting by Access the set of possible access type that can be offered.

In addition, the concept of rule allows us to represent the different rights that are given to a user in MOE.

Definition 1. A rule is a tuple \((a, act, v, c)\) where \(a \in \text{Access}\) is the type of access for this rule, \(act \in \text{Activities}\) is the activity to perform, \(v \in \text{Views}\) refers to the shared resource and \(c\) is a specific condition related to the application. The set of all rules will be denoted by Rules.

An interoperability security policy in MOE is defined by the following function, that assigns to each role a set of rules:

\[
 f_p : \text{Roles} \rightarrow \wp(\text{Rules})
\]

3.4 Request scheme

Finally, the classical MOE request-scheme model in OrBAC (also used in other frameworks such as Poly-OrBAC [7], Multi-OrBAC [6] and O2O [3]) is presented. Graphically the reader can find it in Figure 1 after deleting the Trust-OrBAC Module (Subtasks 3, 3b, 3c, 4 and 4b).

There are two parts in this scheme: the subject and the OrBAC model. On the one hand, a user can send several requests at the same time to an O-grantor in order to access to a shared resource. So, there are several requests, each one identified by a unique id, and these can be collected into logs. On the other hand, for each request, some attributes are collected by the O-grantor in order to offer the required service.

1. There is computed the set of roles \(\{r_1, \ldots, r_n\}\) where this entity is mapped (Subtask 5).
2. After that, it is computed the kind of access \(a\), the action \(ac\), and the object \(ob\) that appears in the request (Subtask 6).
3. Finally, it evaluates if there exists a rule for any role $r \in \{r_1, \ldots, r_n\}$ such that $(a, act, v, c) \in f_p(r)$ and $ac \in act$, $ob \in v$, and the evaluation of $c$ is true (Sub-task 7).

If the evaluation of the rule is true, then the verdict of this request is ACCEPT, otherwise DENY.

4 Trust integration

Trust is a relation between two entities: a truster and a trustee. The truster that is the O-grantor in MOE, relies on offering access to a specific resource for the trustee, i.e., the O-grantee. Based on the specification of MOE, we define two trust relationships. One with respect to the organizations and the other one with respect to subjects. Following, we introduce some concepts that are integrated in TRUST-ORBAC in order to manage trust. The concept of restriction allows us to define what and when the trust relationship is evaluated. Next, the notions of trust subject class, trust object class and trust class that simulate the behavior of the abstract entities of MOE are introduced. Finally, an algorithm that make use of the previous information in order to calculate the set of roles is presented.

4.1 Restriction

This relationship is always restricted to a situation and a time value, that is, the time of the trust evaluation [14,19]. We define a situation as a tuple $(a, v)$ where $a \in \text{Activities}$ and $v \in \text{Views}$. The set of all possible situations will be denoted by Situations.

In our framework, time is represented by using time intervals. Instead of fixed time values, the time is discretized into intervals of the same size, with the particular property that in each interval the trust valuation for an entity does not change. It changes only when the time interval finishes and starts the next one. The $i-th$ time interval will be denoted by $t_i = [t_i, t_{i+1})$, and it will contains all the time values between $t_i$ and $t_{i+1}$.

4.2 Dynamic assignment of classes

There are three concepts that are incorporated in order to manage with the dynamic assignment of the trust. These are trust subject class, trust organization class, and trust class. Basically, theses concepts propose a classification of the behavior of the abstract entities in MOE. A trust subject and a trust organization classes are intervals $[c_1, c_2] : -1 \leq c_1 \leq c_2 \leq 1$. Each organization might define several trust subject and trust organization classes. The set of all trust subject/organization classes are denoted by TSC and TOC respectively. Based on these two elements, we define a trust class as a tuple $(tsc, toc)$ where $tsc \in \text{TSC}$ and $toc \in \text{TOC}$, and the set of all trust classes will be denoted by TC.

The different parameters used to calculate the trust in MOE and their evaluation will be detailed in Section 5.
In **TRUST-OrBAC**, the administrator will associate to each role some trust classes. This mapping depends on the control access requirements. Next, it is defined the function $g : TC \rightarrow \wp(\text{Roles})$, that computes the set of roles that are assigned to a trust class.

### 4.3 Algorithm to compute dynamically the roles

After introducing the notion of trust classes in our model we present the algorithm that computes dynamically the roles in **TRUST-OrBAC**.

```plaintext
Data: request
Result: set of roles where the request entity is mapped

u ← subject(request);  
orq ← organization(u);  
trusts ← current_trust_value(u);  
trusts ← current_trust_value(orq);  
Classes_s ← \{tsc|tsc ∈ TSC ∧ evaluate(trusts) ∈ tsc\};  
Classes_o ← \{toc|toc ∈ TOC ∧ evaluate(trusts) ∈ toc\};  
roles ← ∅;  
foreach ts ∈ (Classes_s × Classes_o) do  
    roles ← roles ∪ g(ts);  
end  
return roles;
```

**Algorithm 1**: Algorithm to compute the set of roles where an entity is mapped by using trust classes.

This algorithm receives as input parameter the request of the user, and it will dynamically provide the set of roles where this subject is mapped. Let us note that the function `current_trust_value` of both: organizations and subjects, depends on time and they are evaluated under demand (maximum once in a period of time).

### 5 Trust parameters and their evaluations

In our framework we use trust vectors and trust policy to represent and calculate the trust. The trust vectors are tuples of elements where relevant parameters such as the experience, the knowledge or the reputation are presented. These parameters are combined in order to provide a trust value in $[-1, 1]$. These parameters are the most relevant parameters [16][15][12][15][19] for evaluating the trust. In addition to these parameters, we add the study of an additional parameter: the impact of the trust evaluation of an organization on its users.

#### 5.1 Trust parameters

There are three classical parameters used to define trust vectors: **Experience learning**, reputation, and knowledge. Experience learning is a process which aims to establish
wisdom on making decision. It is based on the evaluation of the previous interactions between the trustee and the truster related on a specific situation at a period of time. In our framework, there are considered two types of experiences. On the one hand the experience of the trustee organization that takes into consideration the previous behaviors of all users of this organization, that is, organization to organization experience. On the other hand the direct experience where only the previous behaviors between this user and the truster are considered, that is, user to organization experience.

Reputation aims to gather and aggregates feedbacks about an entity from another participant. According to [16], reputation provides an incentive for honest behavior and helps people to make decisions about who to trust. In this paper, we consider that the reputation parameter is only defined for an organization as a type of trustee. Indeed, it is not easy and suitable to measure this parameter of a user in MOE. According to [16], the evaluation of the reputation of an entity must respect some properties. One of them is the longevity of subjects (e.g. no modification of identity) that is not always offered for users in MOE. Since, the exchange of attributes of the same user with organizations may be different. In addition, the collect of the trust evaluation for each user from the several organizations requires the exchange of a huge number of messages which can pose some problems as the useless consumption of bandwidth.

Knowledge aims to evaluate the collection of a specific information and attributes about the trustee. The truster seeks to gather the maximum of attributes and information about the trustee. In our model, we deal with two types of knowledge: Knowledge about an organization (the contract that is signed with the trustee), and the knowledge about a user (the exchange of credentials or/and attributes).

5.2 Evaluation

In order to evaluate these parameters Trust-ORBAC needs to collect some information (some behaviors) into logs files. In this context we assume that any behavior, denoted \( b \), of this file can be valuated as a satisfactory or unsatisfactory behavior. If the valuation is unsatisfactory then it is considered as a bad behavior, that is, it will decrease the experience evaluation of the trustee. On the contrary, if the valuation is satisfactory it will increase the experience evaluation. This evaluation will be realized by a technique that depends on the application and the requirements of the administrator. We note that the output of this technique is a function \( sat(b) \) that associates a value in \([-1,1]\) to the behavior. An example of this function will be given in the case study.

**Definition 2.** We say that:

A request is a tuple \((u,\hat{t},s)\) where \( u \) is subject, \( \hat{t} \) is the time of the reception of this request and \( s \) is the situation that appear in the request.

A behavior is a tuple \((req, dec, sat)\) where the first element \( req \) is a request, \( dec \) is the decision of ORBAC, and \( sat \in [-1,1] \) is the evaluation of this behavior.

A sequence of a behaviors is called log. The log file for the organization \( org_A \) is denoted by \( log_A \), and the set of all logs by \( L \). Given a log file \( l \in L \), we will define three projections functions \( \pi_{\hat{t}}(l) \), \( \pi_s(l) \) and \( \pi_u(l) \). The first one returns those behaviors that were performed in a time in \( \hat{t} \). The second one computes all behaviors of the situation \( s \). Finally, the last one correspond to the set of behaviors of the subject \( u \).
Experience evaluation There are two different experience evaluations in TRUST-ORBAC. The first one, presented in Definition 3 relates the experience of a user with an organization. The second one, introduced in Definition 4 computes the experience of an organization with respect to another organization.

Let us note that the experience of a user related to a situation at the end of a period, taking into account the previous behaviors respects the following properties.

1st Rule The evaluation depends on the partial evaluation of all the behaviors of the user related to the same situation that previously happened.

2nd Rule The influence of an evaluation of any event will decrease with the flow of time. Thus, it needs an attenuation function which decreases the evaluation of an interval depending on the time.

Definition 3. For any $u \in Subjects$, $s \in Situations$, $\hat{t}_i \in I_{\mathbb{R}_{+}}$, $l \in L_{org_A}$, we define the experience evaluation function with respect to $org_A$ as:

$$eX_1(u, org_A, \hat{t}_n, s, l) = \frac{\sum_{i=0}^{n} h(i) \ast \frac{\sum_{b \in l_i} sat(b)}{|l_i|}}{n}$$

where $l_i = \pi_{l_i}(\pi_s(\pi_u(l)))$ and $h$ is an attenuation function.$^4$

Let us note that previous definition computes the experience of any user at a given time and situation as the weighted average of all the evaluations according to an attenuation function.

Next we deal with the second trustee that is the organization. The experience of an organization $org_B$ evaluated by an organization $org_A$, with respect to their users $employee(org_B, org_A)$ holds the following rules:

1st Rule Similar to the 1st and 2nd Rules of the user to organization experience evaluation description.

2nd Rule It depends on the experience evaluation of all users of the organization $org_A$ related to the same situation $s$ at the end of interval $t_n$.

Definition 4. For any $org_B \in Organizations$, $s \in Situations$, $\hat{t}_i \in I_{\mathbb{R}_{+}}$, and for any not empty log $l \in L_{org_A}$, we define the experience evaluation function of an organization $org_B$ with respect to $org_A$ as:

$$eX_2(org_B, org_A, \hat{t}_n, s, l) = \frac{\sum_{u \in employee(org_B, org_A)} eX_1(u, org_A, \hat{t}_n, s, l)}{|employee(org_B, org_A)|}$$

$^4$ The function $h$ is used to satisfy the 2nd Rule. It is chosen by the administrator. As an example, $h(i) = e^{-m(n-i)}$ may be used for this purpose.
Reputation evaluation In TRUST-ORBAC, reputation measures the global perception of the trustee based on recommendation value regarding the trustee of some organizations in this environment. The honesty of the different participants in these statistics must be considered to estimate a right prediction. The set of organizations that can participate in this evaluation, at a given time interval \( t_n \) and according to a situation, is called Friends Group (FG) of the O-grantor. The choice of the elements of FG may be based on the trust evaluation of the different organization of MOE at previous period \( t_{n-1} \). Only the organizations that have a trust value more than a defined threshold can belong to this set. The believe of an organization \( org_i \in FG \) regarding the O-grantee \( org_B \) at the period \( t_j \) related to the situation \( s \) is denoted by \( R(org_i, org_B, s, t_j) \).

**Definition 5.** Let \( org_B \in \text{Organizations}, t_n \in \mathbb{T}_{\mathbb{R}^+} \setminus \{t_0\} \) and \( s \in \text{Situations} \). We define for the O-grantor \( org_A \) the function that evaluates the reputation of an organization with respect to the MOE environment as:

\[
\text{rep}(org_B, t_n, s) = \begin{cases} 
\frac{\sum_{org_i \in FG} R(org_i, org_B, s, t_{n-1})}{|FG|} & \text{if } |FG| \neq 0 \\
0 & \text{otherwise}
\end{cases}
\]

Knowledge evaluation This parameter is evaluated as the classic solution credential based model for ORBAC presented in [4]. Finally, we note here that initially our model will be based on the knowledge and reputation evaluation when there is no logged behaviour between the trustee and the truster.

5.3 Trust vectors and Trust Policy

To complete this section, the notions of trust vectors and policy are introduced. A trust vector for an organization is a tuple \( (E_o, R_o, K_o) \) where \( E_o \in [-1, 1] \) is the experience evaluation of this organization (Formally defined in Definition 4), \( R_o \in [-1, 1] \) is the reputation of the trustee value and \( K_o \in [0, 1] \) is the valuation concerning the knowledge of this organization.

A trust policy aims to define the different weight associated to the parameters of the trust vector. Thus, in this paper we define a trust policy \( (w_{eo}, w_{ro}, w_{ko}) \) for the trust vector for an organization. The three elements of this vector belong to the range \([0, 1]\) and their sum is equal to 1. These values are specified by the administrator of the interoperability security policies. The evaluation of the trust vector is defined as:

\[
w_{eo} \ast E_o + w_{ro} \ast R_o + w_{ko} \ast K_o
\]

A trust vector of a user is a tuple \( (I_u, E_u, K_u) \) where \( I_u \in [-1, 1] \) is the evaluation of the trust vector \( otv \) of the requester organization at the previous period with respect to its internal trust policy, \( E_u \in [-1, 1] \) is the evaluation of the experience between \( u \) and the truster (Formally presented in Definition 3), and \( K_u \in [0, 1] \) is the evaluation of the knowledge concerning this user. As in previous case, there is a trust policy for the user model \( (w_{iu}, w_{eu}, w_{ku}) \), and its evaluation is:

\[
w_{iu} \ast I_u + w_{eu} \ast E_u + w_{ku} \ast K_u
\]
where \( tc_{i,j} = (Tsc_i, Toc_j) \)

**Fig. 2.** Relating trust classes and roles.

We recall that these different evaluations are used in order to determine the trust classes of an organization and a user. This mapping is dynamic since it depends on time. Interested readers may have more details about the trust policy and some comparison methods in [18].

### 6 Case study

Our case study aims to highlight some advantages of our work with respect to similar models like Xena [4] and TrustBAC [2,15]. As it was presented in the related work, these two approaches integrate a trust model to RBAC.

In this case study four organizations, denoted by lab1, lab2, indis1 and indis2, are working in a French project called ISER. The first lab aims to share several versions of the ISER documentation (views). In particular, these are: file code, deriver code, interface code.

The actions that can be performed in these views are: edited, validated, commented, copied, and deleted.

We denote by \( s_1 \) and \( s_2 \) the following situations edit interface code and edit driver code, being the possible roles in this project research engineer and engineer denoted respectively \( r_1 \) and \( r_2 \).

In Figure 2 are represented the trust classes for ISER, and the mapping function of the roles to these trust classes. The meaning of these data is that any user can be mapped to the role \( r_1 \), if the following two conditions hold: a) he provides the different attributes that permits this mapping and b) if both the trust evaluation of his organization and him belong to this set of trust classes \( \{tc_{3,1}, tc_{5,1}, tc_{5,2}\} \).

In this case study, the choice of the different thresholds and their mapping to the existent roles are defined based on some experiments and they are fixed by the administrator.

The reputation parameter will not be used in this case study \( (w_{ro} = 0) \) since we do not have the right to access in the trust evaluation of the other partners. The different participants would hide their collaboration statistics for confidentiality reasons. In addition, we use as evaluation function the one developed in [11]. This function allows us to detect some vulnerabilities of the source, after any modification of a C program.
Next, we present how the algorithm of computing dynamically the set of roles works taking into account previous behaviors works. Basically, in Trust-ORBAC after any time period we provide a new configuration table that contains the list of the different users and organizations, their trust values related to the previous period, the associated trust classes, and the situation. In the left part of the Figure 3 it is presented a part of this file from ISER project. For instance, the 6 – th row represents the trust information of the user u1, belonging to the organization indis1 related to the situation s1 at the end of the period t3. This valuation was computed using the Definition 3, and according to Figure 2 the valuation 0.4 can be mapped in the trust user classes tsc2 and tsc3.

The update of the configuration file will be done after the end of each period. For instance to create the configuration file of t4 Trust-ORBAC will take as input parameters the configuration file t3 and the behavior logs of the period t4. A behavior log of this system is presented on the right part of the Figure 3. Each line contains the request identity (req_id), the subject, his organization, the situation and the evaluation of the behavior (sat(b)) for a permitted request.

Next we detail in the Figure 4 how the reception process of a request in Trust-ORBAC is done. We consider that the user u1 of the organization indis1 applies to edit a file called app.c. Trust-ORBAC works as follows:

1. First of all, Trust-ORBAC determines the situation, that is edit_interface_code.
2. Trust-ORBAC collects the different attributes of u1 by using different negotiation strategies. Trust-ORBAC computes the following set of roles \{r1, r2\} that represents where u1 could be mapped.
3. Next, Trust-ORBAC checks if the trust evaluation of the user and his organization permits this mapping. Therefore the situation, the user and the organization will be forwarded to the trust model. Next, it retrieves the information related to the user u1 and the organization indis1 from the configuration file.
4. According to Figure 2, the trust subject and object classes associated to this request are toc1, toc2 and tsc2, tsc3. A scalar product of these classes gives us a set of trust classes tc2\#1, tc3\#1, tc2\#2, tc3\#2 that can be assigned to this request. The roles where this user can be mapped is \{r1\}, a subset of the previous one.

\[\text{Fig. 3. Configuration and logs files of the system.}\]

<table>
<thead>
<tr>
<th>Trustee type</th>
<th>Trustee Organization</th>
<th>sit Value</th>
<th>Trust Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>org indis1</td>
<td>-</td>
<td>s1 0.8</td>
<td>toc1, toc2</td>
</tr>
<tr>
<td>org indis1</td>
<td>-</td>
<td>s2 0.5</td>
<td>toc1, toc2</td>
</tr>
<tr>
<td>org indis2</td>
<td>-</td>
<td>s1 1</td>
<td>toc1, toc2</td>
</tr>
<tr>
<td>org indis2</td>
<td>-</td>
<td>s2 0.5</td>
<td>toc1, toc2</td>
</tr>
<tr>
<td>org lab1</td>
<td>-</td>
<td>s1 -0.2</td>
<td>toc3</td>
</tr>
<tr>
<td>sub u1</td>
<td>indis1</td>
<td>s1 0.4</td>
<td>tsc2, tsc3</td>
</tr>
<tr>
<td>sub u2</td>
<td>indis2</td>
<td>s1 0.33</td>
<td>tsc2, tsc3, tsc4</td>
</tr>
<tr>
<td>sub u1</td>
<td>indis1</td>
<td>s2 0.8</td>
<td>tsc3</td>
</tr>
<tr>
<td>sub u2</td>
<td>indis2</td>
<td>s2 0.1</td>
<td>tsc2, tsc3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Req_id</th>
<th>Subject</th>
<th>Organization</th>
<th>sit</th>
<th>sat(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1571</td>
<td>u1</td>
<td>indis1</td>
<td>s1</td>
<td>0.85</td>
</tr>
<tr>
<td>2110</td>
<td>u3</td>
<td>indis1</td>
<td>s1</td>
<td>0.3</td>
</tr>
<tr>
<td>2116</td>
<td>u5</td>
<td>lab1</td>
<td>s1</td>
<td>-0.2</td>
</tr>
<tr>
<td>3150</td>
<td>u4</td>
<td>lab1</td>
<td>s2</td>
<td>0.3</td>
</tr>
<tr>
<td>3145</td>
<td>u2</td>
<td>indis1</td>
<td>s1</td>
<td>0.5</td>
</tr>
<tr>
<td>3189</td>
<td>u4</td>
<td>lab1</td>
<td>s2</td>
<td>0.8</td>
</tr>
<tr>
<td>7355</td>
<td>u4</td>
<td>lab1</td>
<td>s1</td>
<td>0.3</td>
</tr>
</tbody>
</table>

\[\text{Fig. 4. Configuration and logs files of the system.}\]

---

5 since the file app.c belongs to the view interface_code.
5. All the abstract entities are ready, based on the interoperability security policy and the decision will be sent to the request and to the trust module. If this request is permitted, the user will be authorized to modify this file.

6.1 Discussion

We discuss in this part the relevant properties of TRUST-ORBAC with previous approaches as: Xena, and TRUSTBAC. The Figure 5(1) presents one diagram that illustrates the dynamic trust value of some users and organizations. It shows the trust level of the user $u_1$, $u_2$ and the organization $indis1$ related to the situation $s_1$ during the different periods $\hat{t}_0$ to $\hat{t}_{14}$. The Figure 5(2) represents the dynamic response of the same request sent by the user $u_1$ in order to perform the situation $s_1$ with the three models.

1. We say that a model is static if the responses regarding the same request with the same environment conditions and the same interoperability policy does not change during the time. TRUSTBAC and TRUST-ORBAC are not static model contrary to
Xena. For TrustBAC, the response may be changed based on the recent behaviors of the users. With Trust-OrBAC, this may be caused due the behaviors of the users or his organization. Indeed, in MOE, the collaboration is defined with the organization. Sometimes, we do not know the real user [7]. For this reason, the use of trust value only with the user cannot satisfy the needs of this environment.

2. Punishment: The behavior of the organization may have bad influence. With our solution, we may have two users from two organizations providing the same attributes and having the same trust level and they will have different responses regarding the same request. This situation happens because the two O-grantees that need the service do not belong to the same trust classes.

3. Rewards: In Trust-OrBAC the influence of the good behavior of the organization in trust is taken into account. For instance, in the Figure 52) after t0, the response of any request sent by u1 related to the situation s1 is always denied with TrustBAC. However, this response may be changed with Trust-OrBAC, i.e., at t10. The trust level of the indis1 increases based on some good interactions of other users. This offers the possibility to u1 to regain some rights that cannot be obtained with TrustBAC.

4. We report the important role of the situation for the same subject, in the same period. This is similar for the three models, the trust model depends on the situation. However, the definition of this concept is only detailed in our framework.

Finally, we summarize the three approaches in the following table:

<table>
<thead>
<tr>
<th></th>
<th>Xena</th>
<th>TrustBAC</th>
<th>Trust-OrBAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Influence of the organization behavior</td>
<td>×</td>
<td>×</td>
<td>✓ (punishment and rewards)</td>
</tr>
<tr>
<td>Access Control Model</td>
<td>Extended RBAC</td>
<td>RBAC</td>
<td>OrBAC</td>
</tr>
<tr>
<td>Influence of the user behavior</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Situation definition</td>
<td>Ambiguous</td>
<td>Ambiguous</td>
<td>Detailed</td>
</tr>
<tr>
<td>Knowledge</td>
<td>Collection of attributes with negotiation strategies</td>
<td>Ambiguous</td>
<td>Inspired from Xena</td>
</tr>
</tbody>
</table>

Table 1. Trust model in Role Based Access Control.

7 Conclusions and Future Work
In this paper, we present Trust-OrBAC, a model for access control in Multi-Organization Environments. This model is an extension of OrBAC and it is based on dynamic assignment of roles to a user. This assignment is realized with the use of two trust vectors. One for the organization and another for the users. This trust approach is defined according to the properties of MOE. For the best of our knowledge, Trust-OrBAC is the first model that defines a dynamic and multi-trust vector based on OrBAC concepts in MOE.

As future work, we are planning to study with more details the representation of satisfactory and unsatisfactory behaviors, and we aim also to integrate our approach into the eXtensible Access Control Markup Language (XACML) that was standardized by the OASIS community.
References


