An Access Control Method based on Scenario Trust

Shunan Ma\textsuperscript{1,2}
\textsuperscript{1}College of Computer Science and Technology
Beijing University of Technology, Beijing 100124, China
\textsuperscript{2}Institute of Information Engineering, Chinese Academy of Science
E-mail: mashunan@emails.bjut.edu.cn

Jingsha He\textsuperscript{3}
\textsuperscript{3}School of Software Engineering
Beijing University of Technology, Beijing 100124, China
E-mail: jhe@bjut.edu.cn

Xunbo Shuai\textsuperscript{4}
\textsuperscript{4}Research Institute of Petroleum Exploration & Development-Langfang
PetroChina, Langfang 065007, China
E-mail: shuaixb69@petrochina.com.cn

Received 30 November 2011
Accepted 19 June 2012

Abstract

Access control is an important element in network security. Meanwhile, trust provides a new direction for access control in open network environments. Based on the dynamic nature of trust, we study the temporal and spatial characteristics in the security of society and propose the concept of scenario trust in which we consider four factors: access time, place, history behavior and risk control strategy. First, we apply fuzzy clustering method and information entropy theory to the design of an algorithm for allocating weights to the factors of scenario trust and, consequently, quantify scenario trust. Then, we introduce the notion of trust threshold for fine-grained access permission management and describe the rules that map scenario trust values to access permissions. On this basis, we propose a dynamic access control model based on scenario trust. Simulation results show that the proposed method is effective and can defend joint fraud. The model exhibits good scalability and can meet the need of dynamic access control in open network environments.

Keywords: access control, scenario trust, permission, fuzzy clustering

1. Introduction

Access control, which is used to restrict access to network information and resources, is an important element in network security. In open and dynamic network environments, not every resource requestor can be known in advance by resource owners\textsuperscript{1}. Therefore, although traditional access control models are suitable for centralized and static environments, they can hardly meet the requirements of open and dynamic network environments because of the lack of sufficient information about subjects and objects required by most traditional models to make access control decisions. That is, traditional access control models are best suited for centralized and relatively static environments in which information about subjects and objects is known in advance, but these models face new challenges in open and dynamic network environments.

Trust is part of our daily life and thus can be relied upon when making access control decisions, which can be
realized by using trust to provide network security\textsuperscript{2,3}. In some early work\textsuperscript{4,5}, trust models have been proposed to control anonymity, unpredictability and uncertainty. In recent years, many researchers have applied trust in dynamic environments.

Access control is an important security mechanism, which is used to authorize and limit user access to information and resources to prevent users from making illegal access in violation of security policies. There are mostly three traditional models for access control: discretionary access control (DAC)\textsuperscript{6}, mandatory access control (MAC) and role-based access control (RBAC)\textsuperscript{7,8}. However, they are not very suitable for open and dynamic network environments due to the lack of flexibility in dealing with uncertainties in making access control decisions. This is mainly because it is not always possible to know the real identities of users who issue access requests in these environments. To overcome the shortcomings of the traditional access control models, some researchers have proposed credential-based access control models\textsuperscript{9,10} in which each user has to present a predetermined set of credentials to get access permissions. The credentials provide trust information about the authorities themselves. However, although credential-based models can solve the problem of access control in open systems to some extent, the shortcomings are equally obvious. For example, a credential doesn’t guarantee that its bearer really satisfies the claims in the credential. Neither does it convey any information about the behavior of the bearer between the time when the credential is issued and the time when it is actually presented. Such information may be crucial in making access control decisions.

Several access control models based on trust have been proposed in recent years. TrustBAC extends conventional RBAC with the notion of trust level\textsuperscript{11} in which users are first assigned to trust levels instead of roles based on user identity, history behavior, recommendation, etc. When the trust level of a user changes, the role assigned to the user also changes correspondingly. Some other work in this area also includes a trust model for access control in peer-to-peer environments\textsuperscript{12} and a Role-based Trust management framework (RT) to integrate RBAC with trust management system\textsuperscript{13}, thus making it suitable for credential-based access control systems, and a privacy-aware access control framework with trust management in web services\textsuperscript{14}. In all the work proposed above, although trust has been introduced into access control models, the issue of mapping trust values to access permissions has not been studied.

Measurement of trust relationships between entities and mapping methods from trust values to access permissions are two key issues for trust-based access control in open network environments. Although classical mathematical functions can be used to compute trust values, they often lead to inaccuracies or errors due to the fact that trust has the characteristics of subjectivity and dynamism. As for the issue of mapping trust values to access permissions, current practice often divides trust values into a finite number of intervals that correspond to the permission sets. Consequently, it is hard to achieve fine-grained access control, not to mention the lack of flexibility to dynamically adjust access permissions.

In this paper, we study the temporal and spatial characteristics of the social order and introduce the concept of scenario trust, which describes trust relationships between entities in open network environments. Our proposal of the scenario trust includes four factors: access time, place, history behavior and risk control strategy. We then apply fuzzy clustering method and entropy theory to the design of an algorithm for allocating weights to the factors of the scenario trust, thus quantifying trust. We also propose to use trust threshold for permissions to manage fine-grained access control and present rules for mapping scenario trust values to access permissions. Finally, we propose a dynamic access control model based on scenario trust, in which we describe the key data structures as well as data flow of the model. Based on simulation that we perform, we can show that our proposed model is suitable for dynamic access control in open network environments, which is also effective in defending joint fraud by collaborative network users. In addition, we show that our proposed model has high access control efficiency and good scalability.

The rest of this paper is organized as follows. In the next section, we introduce the notion of scenario trust and describe a method to quantify scenario trust. In Section 3, we present a method for mapping quantified trust values to access permissions. In Section 4, we combine scenario trust and access permission mapping together and propose a model for access control in open, dynamic network environments. In Section 5, we perform some simulation to demonstrate the effectiveness of our proposed access control method by showing how trust values change as the factors of scenario trust change and how access
permissions change when trust values change to meet the
dynamic changes in open network environments. Finally,
in Section 6, we conclude this paper in which we also
discuss some future research directions.

2. Scenario Trust

The purpose of applying trust to the control of access to
information and resources is to prevent “illegitimate”
subjects from gaining access to information and
resources as accurately as possible. From another
perspective, the access behavior of an “illegitimate”
subject is similar to crime in law since they both violate
the constraints defined for the actions. Research has
shown that crime has significant temporal and spatial features\(^{15,16}\). For example, November and December are
the period in which most criminal cases happen during a
year and most criminal cases happen between 6pm and
12pm during a day. Moreover, the crime rate is usually
higher in urban areas than in suburban areas. In
sociology, trust on an individual who has criminal record
is significantly lower than that on an ordinary individual.
In our study, we use the IP address of a subject to denote
the factor for place. On this basis, we go on to propose
the notion of scenario trust, which includes three other
factors: access time, history behavior and risk control
strategy. Following are several definitions and concepts
about scenario trust on which we rely in our access
control method.

Scenario trust: access scenario is comprised of four
factors: access time, place and history behavior of a
subject and risk control strategy of the objects. Scenario
trust refers to the trust evaluation by an object on a
subject in a specific scenario.

We can see from the above definition that scenario
trust exhibits three characteristics: dynamic, subjective
and ambiguous.

Risk control strategy: the ability of control or repair of
an object after unauthorized access takes place by a
subject.

The impact of the factor of risk control strategy on
trust evaluation is closely related to the security system
of an object. Generally speaking, the higher the security
level of an object, the stronger the risk control ability and
the higher the trust value of a subject that is required.

Scenario trust computation: given the factors of
scenario trust evaluation, suppose each factor’s trust
value is \( T_0 \), \( T_1 \), ..., \( T_{n-1} \), respectively, the weight of each
factor is \( W_0, W_1, \ldots, W_{n-1} \), respectively, and \( \sum_{i=0}^{n-1} W_i = 1 \),
then the scenario trust value is
\[
T = \sum_{i=0}^{n-1} (W_i \times T_i).
\]

We can see from above that scenario trust has good
scalability, i.e., to expand to include additional factors.
For different applications, factors of scenario trust can be
determined specifically. In addition, scenario trust
computation consists of two parts: to select each factor’s
trust evaluation method to get the trust value for the
factor and to determine each factor’s weight allocation.

2.1. Evaluation of Scenario Trust Factors

According to an object’s property and a subject’s history
behavior information, we can establish tables for the
temporal and spatial properties of the trust evaluation
rules. For example, the trust value of a subject who
accesses a recreation resource during working hours is
lower than that during off-work hours. For access to
education resources, trust value of a subject whose IP
address belongs to an education network segment is
higher than that of non-education network segment.

According to the property of each resource, time can
be divided into \( n \) periods. For each time period \([t_i,t_j]\), we
formulate corresponding trust interval \([T_i,T_j]\), which
means that when a subject accesses the resource at time \( t \),
for \( t_i \leq t \leq t_j \), and randomly generate a trust value
\( T \in [T_i,T_j] \). To avoid denial of access to an object for
preventing fraud in high crime periods, we use the
following method. For each time period \([t_i,t_j]\), when the
number of access reaches a certain value during this time
period, we count the total number of access as \( m \) and the
total number of fraud as \( k \) and express fraud probability
in this period as \( p = k/m \). Then, we randomly generate a
trust value \( T_x \) at time \( x \) where \( T_x \in [T_i,T_j] \). The trust value
for the time factor in the scenario trust is thus:
\[
T = T_x \times (1 - p) \tag{1}
\]

According to the time attribute of a resource, the trust
evaluation table for the time factor can be formed and is
shown in Table 1.

<table>
<thead>
<tr>
<th>Table 1. Trust evaluation table for the time factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time period</td>
</tr>
<tr>
<td>Trust interval</td>
</tr>
<tr>
<td>Fraud probability</td>
</tr>
</tbody>
</table>
We use the IP address to represent the place factor in our discussion. Subjects can then be classified based on the property of objects. For example, subjects can be classified into groups in the same subnet, same domain, important service segment and general service segment, etc. Then, we formulate the trust evaluation intervals for the groups. For each network segment, when the number of access times reaches a certain value, we calculate the fraud probability in this network segment. For a given IP address, we can use formula (1) to compute the trust value for the IP factor (i.e., the place factor). Table 2 shows the trust evaluation table for the IP factor.

<table>
<thead>
<tr>
<th>IP address</th>
<th>Same subnet</th>
<th>Same segment</th>
<th>Important service</th>
<th>General service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trust interval</td>
<td>[T₀,T₁)</td>
<td>[T₁,T₂)</td>
<td>[T₂,T₃)</td>
<td>...</td>
</tr>
<tr>
<td>Fraud probability</td>
<td>p₀</td>
<td>p₁</td>
<td>p₂</td>
<td>...</td>
</tr>
</tbody>
</table>

The purpose of introducing history behavior is to control the subject who commits fraud after getting access permissions. To support this functionality, each object would record some information about the subjects who have accessed it, which is similar to a “file” for each person. Objects can use these history behavior records as the foundation for a subject’s future trust computation. If a subject can access the object, the object will use access feedback policy in which we denote \( S \) as the access feedback satisfaction degree. An object would establish an access record table to record subjects’ access information. The access record table includes: access number, ID, scenario trust factors’ values and access feedback satisfaction degree. The access record table is regarded as a subject’s “file”. Such an access record table is shown in Table 3.

<table>
<thead>
<tr>
<th>Sequence number</th>
<th>ID</th>
<th>Scenario trust</th>
<th>Behavior feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Subject i</td>
<td>T₃₀</td>
<td>T₁₁</td>
</tr>
<tr>
<td>2</td>
<td>Subject k</td>
<td>T₃₀</td>
<td>T₂₁</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>n-1</td>
<td>Subject k</td>
<td>Tₙ₋₀</td>
<td>Tₙ₋₁</td>
</tr>
<tr>
<td>n</td>
<td>Subject j</td>
<td>Tₙ₀</td>
<td>Tₙ₁</td>
</tr>
</tbody>
</table>

In the access record table, \( T_{ij} \) is the product of \( j \)th factor’s trust value and its weight assigned to subject \( i \). The trust value of history behavior is the product of most recently accessed feedback value and its weight. To reduce access control risk, when a subject first accesses an object, the trust value for the history behavior is defined as 0.5, a neutral value.

Objects use different security policies and the degree of protection may differ, namely, objects have different risk control abilities. Objects formulate the trust value of the risk control factor based on its own policy. When such policy is formed, the basic principle is that the higher the level of security policy, the greater the trust value. Risk control table is shown in Table 4.

<table>
<thead>
<tr>
<th>Object</th>
<th>Policy 1</th>
<th>Policy 2</th>
<th>...</th>
<th>Policy n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object 1</td>
<td>T₁₁</td>
<td>T₁₂</td>
<td>...</td>
<td>T₁ₙ</td>
</tr>
<tr>
<td>Object 2</td>
<td>T₂₁</td>
<td>T₂₂</td>
<td>...</td>
<td>T₂ₙ</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Object m</td>
<td>Tₘ₁</td>
<td>Tₘ₂</td>
<td>...</td>
<td>Tₘₙ</td>
</tr>
</tbody>
</table>

2.2. Weight Allocation

Suppose that scenario trust consists of \( m \) factors, and each factor is denoted as \( x = \{ T₀, T₁, \ldots, Tₜ \} \). We can then design a weight allocation algorithm based on fuzzy clustering method\(^{17,19}\) for the scenario trust factors. The algorithm is described as follows:

(1) Each scenario trust is initialized to a separate category, which is denoted as \( v_i = \{ x_i \}, \ i = 0, 1, \ldots, n-1 \).

Then, these \( n \) history scenario trust values can form a \( m \times n \) matrix \( X = \begin{bmatrix} T₀₀ & T₀₁ & \cdots & T₀ₚ \ T₁₀ & T₁₁ & \cdots & T₁ₚ \ \vdots & \vdots & \ddots & \vdots \ Tₙ₋₀ & Tₙ₋₁ & \cdots & Tₙ₋₁ₚ \end{bmatrix} \in \mathbb{R}^{m \times n} \), in which each column denotes the value for each factor.
of the scenario trust. For example, $T_j$ represents the trust value for the $i$th factor in scenario trust $j$.

(2) For each row of $X$, $T_j = \frac{T_j}{\max(T_{i0}, T_{i1}, \ldots, T_{ik-1})}$ where $T_j \in [0, 1]$.

(3) Establish $n \times n$ fuzzy similarity matrix $R$. For any $\sum_{k=0}^{n-1} (T_k \wedge T_k)$, calculate $r_{ij} = \frac{\sum_{k=0}^{n-1} (T_k \vee T_k)}{\sum_{k=0}^{n-1} (T_k \vee T_k)}$.

(4) For the fuzzy similarity matrix $R$, use the Equivalence Closure method $20-22$ to get the fuzzy equivalence matrix $H$.

(5) Let set $C = \phi$. If $h_{ij}$ denotes the element of the $i$th row and the $j$th column in matrix $H$, for any given $h_{ij}$, if $h_{ij} \notin C$ and $h_{ij} \neq 1.0$, $C = C \cup \{h_{ij}\}$; $q = \frac{1}{m} \sum_{k=0}^{m-1} C_k$; $l = \max[C]$; and $g = \left[\frac{\lfloor q \times 10 - 0.5 \rfloor}{0.5}\right]$.

If $g \leq 0$, then $G = q$; Otherwise, $G = \left[\frac{q \times 10 - 0.5}{0.1 \times (i-1)}, i = 0, 1, \ldots, g-1\right]$.

(6) Construct the classification matrix $B$ from fuzzy equivalence matrix $H$. For any given element $b_{ij}$ in matrix $B$, $b_{ij} = \begin{cases} 1 & h_{ij} \geq G \\ 0 & h_{ij} < G \end{cases}$. Search each row of $B$ for any $b_{ij} = 1$, $j < i$. If it doesn’t satisfy $j < k < i$ and $b_{ik} = 1$, then the $i$th history scenario trust category is combined with the $j$th history scenario trust category, namely, $v_j = v_j \cup v_i$, $v_i = \phi$.

(7) Compute the entropy of the $n$ history scenario trust sets: $I = -\sum_{i=0}^{n-1} \frac{1}{n} \log_2 n$.

(8) Delete each row of the scenario trust matrix $X$ in turn, namely, delete each element of scenario trust, resulting in a new scenario trust matrix $Y_0, Y_1, \ldots, Y_m$. For matrix $Y_k$, $k=0,1,\ldots,m-1$, perform step (2) to step (7) repeatedly to get the values of $G_k$ and $I_k$.

If $G = G_k$, then $M_k = I_k / G_k$. Otherwise, $M_k = \frac{I_k - I_k}{G_k - G_k}$ in which $M_k$ denotes the dependence of $n$ scenario trust on the $k$th factor.

(9) According to each factor’s impact on history scenario trust, determine its weight $W_i = \frac{M_k}{\sum_{j=0}^{m-1} M_j}$.

(10) Output each factor’s weight and then terminate the algorithm.

2.3. Scenario Trust Realization

We now describe the data flow of the access record table and each factor’s trust table. Every object has an access record table and a trust table for each factor. We can get the trust value for each factor and apply the weight allocation algorithm to get the weight for each factor. We then compute the value of scenario trust for each subject and store the value in the object’s access record table, which provides the foundation for future trust computation. The data flow chart is shown in Fig. 1.

Fig. 1. Data flow chart
We can see from the above data flow chart that a subject can commit fraud in the following manner. The subject first deceives to get better behavior feedback so that it can gain high history behavior factor’s trust for future access. We then develop a method to fight against such fraud. First, in computing the trust value for the history behavior factor, we can use the average of a subject’s several behavior feedback values. If one history behavior feedback value is used to compute the trust value for the history behavior factor, the corresponding history behavior record won’t be used as the sample to get each factor’s weight. This strategy would then separate the behavior feedback value and weight computation. The worst case is that the malicious subject gets a high trust value for the history behavior factor through deception, but it would not get a high weight for the history behavior factor, thus reducing the influence of fraud to the scenario trust values.

3. Permission Mapping Method

3.1. Trust Threshold for Permissions

Suppose that an object has \( N \) distinct permissions for subjects to use for access to the object. Given a permission \( r \), when a subject’s trust value is \( T \), if \( T \geq T_n \), it can get the permission \( r \); if \( T < T_n \), it won’t get the permission \( r \). Therefore, we propose a concept called trust threshold for permission to realize the above access requirement. Namely, assuming that the minimum required trust value for getting access permission \( r \) is \( \varepsilon \), if trust value on a subject \( T \geq \varepsilon \), the object will assign permission \( r \) to the subject, \( \varepsilon \) is called the trust threshold for permission \( r \).

If the trust thresholds for more than one access permissions are the same, we will merge these permissions to form a permission set. The trust threshold for the permissions then becomes the trust threshold for the permission set.

We can see that trust threshold for permissions can be very dynamic. Set the trust threshold for permission is based on the experience of an object or on the records of access history, and the trust threshold can be dynamically adjusted. Consequently, every permission corresponds to a trust threshold. Thus, for any permission, access control can only compare the trust value on a subject and the trust threshold for permission to achieve fine-grained access control.

3.2. Permission Mapping Method based on Trust Threshold

The main idea of the mapping method from scenario trust values to access permissions is as follows. First, we determine the trust threshold for every permission and sort these trust thresholds in an ascending order, i.e., \( \varepsilon_0 < \varepsilon_1 < \varepsilon_2 < \cdots < \varepsilon_{n-1} \) where \( \varepsilon_i \) corresponds to \( r_i \). Then, for any given scenario trust value \( T \), if \( \varepsilon_{i-1} \leq T < \varepsilon_i \), where \( i = 1, 2, \ldots \), the permission set which can be assigned to the subject is \( \{ r_{i-1}, r_i \} \).

We now present a trust threshold for permission selection method. Suppose the trust interval is \([0,1]\) in which 0 indicates no trust, 1 indicates total trust and any value between them indicates partial trust. Using trust threshold for permission, an object classifies all the permissions. If more than one trust threshold for permission are the same, the permissions will be merged to form a permission set. Now, suppose there are \( n \) permission sets after the merge. We can then sort these \( n \) sets based on their trust thresholds. The object gives a minimum trust threshold \( \varepsilon \) to all the permission sets and the \( n \) trust thresholds for permission sets after the sorting thus become

\[
\varepsilon_0 = \varepsilon, \varepsilon_1 = \varepsilon + \frac{1 - \varepsilon}{n}, \varepsilon_{n-1} = \varepsilon + \frac{(1 - \varepsilon)(n-1)}{n},
\]

respectively. An object can build a permission distribution record table for each permission set. The permission distribution record table mainly includes the following elements: subject’s ID, trust value and behavior feedback value. The difference between the permission distribution record table and the access record table is that a permission distribution table records the authorization case for each permission set and an access record table records each access case. Every object has only one access record table, but the number of permission distribution record tables equals to the number of permission sets. For each permission distribution record table, whenever there is fraud, the trust threshold for permission set will be increased and, consequently, the permission distribution record table is updated. If there is no fraud behavior after a certain number of accesses to the object, the trust threshold for permission set can be reduced to allow more subjects to make access to the object. When the increased or reduced value becomes lower than \( 10^{-6} \), the present
threshold value is considered the final threshold value for the permission set.

Below we describe a method for iterative calculation to adjust the threshold values, either increasing or decreasing.

Suppose there are \( n \) sorted permission sets, given permission set \( r_i \) whose threshold is \( e_i \in [a, b] \). If \( i = 0 \), namely, \( r_i \) is the permission set with the smallest threshold value, then \( a = 0 \). If \( i = n-1 \), namely, \( r_i \) is the permission set with the largest threshold value, then \( b = 1 \). If \( i = 1, 2, ..., n-2 \), then \( a = e_{i-1}, b = e_{i+1} \).

When \( e_i \) needs an upward adjustment, the trust value of a subject who has fraud behavior is denoted as \( T \) and \( e_i < T < b \), then \( a = e_i \) and \( b = b - (T - a) / 2 \).

When \( e_i \) needs a downward adjustment, the smallest trust value of a subject who has performed access is denoted as \( T \) and \( e_i < T < b \), then \( b = e_i \) and \( a = a + (T - b) / 2 \).

We can see from the above iterative calculation method for threshold that every permission adjustment needs to read the thresholds for the adjacent permission sets, namely, to determine \( e_i \in [a, b] \). Then, gradually approach the best trust threshold for permission through iterative calculation. Every time after adjusting the trust threshold for permission, delete the content of the permission distribution table.

Following is the procedure of the algorithm for permission mapping based on threshold:
(1) According to the requirement for subjects’ trust values, the object classifies its permissions and initializes the threshold for each permission set.
(2) Build a permission distribution record table for each permission set if it doesn’t have one yet.
(3) For a scenario trust value \( T \), search the trust threshold sequence \( e_0 < e_1 < e_2 < ... < e_{n-1} \). If \( e_{i-1} \leq T < e_i \), where \( i = 1, 2, 3, ..., \), the subject has all the permissions in the permission set that corresponds to trust threshold \( e_{i-1} \).
(4) If the trust threshold \( e_{i-1} \) for permission set is the final threshold, go to step (3).
(5) Store the scenario trust value, authorized permission set and the behavior feedback value in the permission distribution record table.
(6) Based on the updated information in the permission distribution record table, adjust the threshold \( e_{i-1} \) for permission set.

(7) If the adjustment value is less than \( 10^{-6} \), the present threshold becomes the final threshold for the permission set, go to step (3). Otherwise, the trust threshold for permission set is replaced by the adjusted threshold and the old permission distribution table is deleted. Go to step (2).

4. Dynamic Access Control based on Scenario Trust

We now combine the scenario trust computation and the permission mapping method based on trust threshold and design a scenario trust based dynamic access control (STBDAC) model. The model includes nine functional modules: access request, access record table, weight allocation, permission classification, scenario trust computation, trust threshold adjustment, permission mapping, behavior feedback and permission distribution table. The STBDAC model is shown in Fig. 2 and the main functions of each module are described below.
distributes the weight for each factor of scenario trust based on the access record table.

(4) The scenario trust module computes scenario trust values based on the trust value and the weight for each factor.

(5) The permission classification module classifies and sorts all access permissions to form a finite number of permission sets according to the requirement of trust that an object has on subjects.

(6) The trust threshold adjustment module adjusts the trust threshold based on information in the permission distribution table.

(7) The permission mapping module compares a subject’s scenario trust value with the trust thresholds for permission sets and assigns corresponding permissions to the subject.

(8) The behavior feedback module provides feedback information to reduce the risk in the next round of access control.

(9) The permission distribution table records every permission set’s assignment.

The proposed STBDAC model has the advantages of dynamism and extendibility, and it can be used to defend joint fraud behavior.

5. Simulation and Analysis

5.1. Simulation Environment

In order to verify the dynamism and the control ability to fraud using the proposed STBDAC model, we have developed and performed some simulation in Java to simulate an open distance education management platform in which objects are the education resources in the platform. The object uses an encryption policy and the trust value for the risk control factor \( T = 0.75 \). The permission sets of the object are \( R_1 = \{ \text{read} \}, \ R_2 = \{ \text{print} \}, \ R_3 = \{ \text{download} \}, \) and \( R_4 = \{ \text{update} \}, \) respectively. In a period of access, trust thresholds for permission sets are \( \varepsilon_1 = 0.4 , \varepsilon_2 = 0.55 , \varepsilon_3 = 0.75 , \varepsilon_4 = 0.85 \), respectively. After each access, the behavior feedback value for an honest subject is \( S \in [0.88,1] \) and that for a malicious subject is \( S \in [0.08,1] \). Given certain permissions, if there is no fraud after continuous access for five times, access control decreases the trust threshold. On the other hand, if there is fraud, access control increases the trust threshold.

5.2. Access Control Result

In order to illustrate the effect of access control, we simulate access by seven subjects and select one among the seven to be the experiment target. We go through the access control process to verify the access control effect when access behavior changes.

(1) Randomly generate the history scenario trust values for six subjects and the trust value for every factor is shown in Table 5:

<table>
<thead>
<tr>
<th>sequence number</th>
<th>ID</th>
<th>Scenario trust time</th>
<th>Scenario trust IP</th>
<th>Scenario trust History behavior</th>
<th>Scenario trust Risk control</th>
<th>Behavior feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Subject i</td>
<td>i</td>
<td>0.4231</td>
<td>0.4493</td>
<td>0.5312</td>
<td>0.5099</td>
<td>...</td>
</tr>
<tr>
<td>2 Subject k</td>
<td>k</td>
<td>0.7205</td>
<td>0.4446</td>
<td>0.4551</td>
<td>0.5034</td>
<td>...</td>
</tr>
<tr>
<td>3 Subject p</td>
<td>p</td>
<td>0.4052</td>
<td>0.4774</td>
<td>0.5715</td>
<td>0.4261</td>
<td>...</td>
</tr>
<tr>
<td>4 Subject r</td>
<td>r</td>
<td>0.4066</td>
<td>0.4842</td>
<td>0.7909</td>
<td>0.6289</td>
<td>...</td>
</tr>
<tr>
<td>5 Subject q</td>
<td>q</td>
<td>0.6520</td>
<td>0.4086</td>
<td>0.4713</td>
<td>0.5792</td>
<td>...</td>
</tr>
<tr>
<td>6 Subject j</td>
<td>j</td>
<td>0.5119</td>
<td>0.3767</td>
<td>0.6217</td>
<td>0.4654</td>
<td>...</td>
</tr>
</tbody>
</table>

(2) According to step (3) of the weight allocation algorithm, we can get matrix \( X \):

\[
X = \begin{bmatrix}
0.4231 & 0.7205 & 0.4052 & 0.4066 & 0.6520 & 0.5119 \\
0.4493 & 0.4446 & 0.4774 & 0.4842 & 0.4086 & 0.3767 \\
0.5312 & 0.4551 & 0.5715 & 0.7909 & 0.4713 & 0.6217 \\
0.5099 & 0.5034 & 0.4261 & 0.6289 & 0.5792 & 0.4654 \\
\end{bmatrix}
\]

(3) From the weight allocation algorithm, we can get the fuzzy equivalence matrix \( H \) without deleting any element.

\[
H = \begin{bmatrix}
1.0 & 0.8449 & 0.9140 & 0.8293 & 0.8449 & 0.8449 & 0.8583 \\
0.8449 & 1.0 & 0.8449 & 0.8293 & 0.9096 & 0.8449 & 0.8449 \\
0.9140 & 0.8449 & 1.0 & 0.8293 & 0.8449 & 0.8583 & 0.8293 \\
0.8293 & 0.8293 & 0.8293 & 1.0 & 0.8293 & 0.8293 & 0.8293 \\
0.8449 & 0.9096 & 0.8449 & 0.8293 & 1.0 & 0.8449 & 0.8449 \\
0.8583 & 0.8449 & 0.8583 & 0.8293 & 0.8449 & 1.0 & 0.8583 \\
\end{bmatrix}
\]

(4) The classification of history record yields: \{1,3\}, \{2,5\}, \{4\}, \{6\} and system entropy \( I = 1.9183 \).

(5) The threshold average and the entropy after deleting each factor for scenario trust is shown in Table 6 from which we can compute: \( M_x = 40.6524 \), \( M_y = 28.1718 \), \( M_z = 52.7816 \), \( M_p = 39.5862 \).

(6) Finally, we get the weights for time, IP address, history behavior and risk control strategy...
respectively: 0.2522, 0.1748, 0.3274, 0.2456.

Table 6. Factor’s history record classification and information entropy

<table>
<thead>
<tr>
<th>time</th>
<th>IP</th>
<th>History behavior</th>
<th>Risk control</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F_k )</td>
<td>0.8876</td>
<td>0.8549</td>
<td>0.8798</td>
</tr>
<tr>
<td>( I_k )</td>
<td>1.2516</td>
<td>1.4591</td>
<td>1.4591</td>
</tr>
</tbody>
</table>

We can thus see that the weight allocation algorithm of STBDAC is feasible. To verify the access control effect using STBDAC, we distribute the weight for each trust factor based on the experiment result. We make a subject access the object continuously for ten times and the trust values for the time and the IP factors are randomly generated within [0.6, 0.7]. In our experiment, the behaviors for the first six accesses are honest so that the behavior feedback values gradually increase. Starting from the seventh access, however, the behavior feedback values gradually decrease. The corresponding changes on the subject’s scenario trust value and on the access permissions are shown in Fig. 3 and Fig. 4, respectively.

We can see from Fig. 3 and Fig. 4 that if access time and IP address remain the same, when the subject’s behavior is honest, the scenario trust values gradually increase. When there is no fraud during continuous access to a certain number, the object considers the current environment to be friendly and will consequently decreases the trust threshold to allow more subjects to make access. When the subject’s behavior becomes malicious, the behavior feedback decreases rapidly, resulting in the trust values for the history behavior factor and the risk control strategy factor to decrease. The trust threshold for permission is increased accordingly so that the subject will no longer have the access permission because of the rapidly decreased scenario trust value. We can thus see from the experiment result that the STBDAC model can perform dynamic access control.

5.3. Solution to Prevent Joint Fraud

Because scenario trust is dynamic, the scenario trust values for malicious subjects gradually decrease, and the trust threshold for permission increase, which can reduce the probability that the subjects’ continuously committing fraud. In order to show that our STBDAC model is able to defend joint fraud, we assume that in the worst case, there are different malicious subjects at different times and that adjacent trust thresholds won’t change.

We perform nine experiments in which the trust threshold at the beginning of each experiment is the adjusted threshold in the previous experiment. Given permission \( R_2 \), we select six subjects to commit joint fraud in each experiment and the first five subjects pretend to be honest to cause the trust threshold to decrease in order to provide the condition for the sixth subject to commit fraud. We can see from the permission mapping method based on trust threshold in Section 3.2 that, in each experiment, a subject’s scenario trust value \( T \in [a, b] \) where \( a \) and \( b \) are the intermediate variables for threshold adjustment. In the first experiment, suppose that the object is cheated and the trust threshold for permission is adjusted to \( \varepsilon_2 = 0.67705 \). In these nine experiments, the trust threshold is adjusted seventeen times and changes are shown in Fig. 5.

We can see from Fig. 5 that, in the worst situation, the STBDAC model can still dynamically adjust the trust threshold. Moreover, with the increase in the number of accesses, the trust threshold for permission approaches a certain value, i.e., the optimal value. In the worst situation, the adjacent thresholds of
permission $R_2$ do not change. From the fifth experiment, the trust threshold for permission becomes lower and the risk of joint fraud has increased. Nonetheless, we can see from the experiment that, up to the fifth experiment, the object has resisted 10 subjects from committing joint fraud for 30 times. These ten subjects are five pretend-to-be honest subjects and five malicious subjects who deceive the object in turns. In practical applications, since each trust threshold for permission can dynamically adjust along with the change of access behavior and the scenario trust value of a subject also changes based on the environment, our STBDAC model can defend joint fraud to a large extent.

6. Conclusions

According to the dynamic nature of trust and by mimicking the temporal and spatial characteristics in the security of society, in this paper, we proposed the concept of scenario trust that includes four factors: access time, place, history behavior and risk control strategy. We also applied the fuzzy clustering method and information entropy theory to design a weight allocation algorithm for the factors of scenario trust to compute the scenario trust values. We then introduced the notion of trust threshold for fine-grained permission management in access control and described the rules to map scenario trust values to access permissions. After presenting a method for dynamic adjustment of trust thresholds for permissions, we finally proposed a dynamic access control model based on scenario trust and described its key data structure and data flow. Simulation results show that the proposed model has high efficiency and can also defend joint fraud. Moreover, the model exhibits good scalability and can meet the requirement of dynamic access control in open network environments.

References

An Access Control Method Based on Scenario Trust


