Study on Centralized Authorization Model Supporting Multiple Access Control Models

Shaomin Zhang  
Department of Computer, North China Electric Power University, Baoding, China  
e-mail: zhangshaomin@126.com

Haiyan Zhang  
Department of Computer, North China Electric Power University, Baoding, China  
e-mail: zhhy860121@126.com

Baoyi Wang  
Department of Computer, North China Electric Power University, Baoding, China  
e-mail: wangbaoyi@126.com

Abstract—With the rapid development of the process of information of large-scale enterprises and organizations, information integration has become a hot research issue at present. Existing systems usually adopt access control model, such as ACL (access control list), MAC (mandatory access control) or RBAC (role-based access control), which results in the diversity of access control models in system integration. This paper designs a centralized authorization model supporting multiple access control, which authorizes permissions for multiple heterogeneous information systems on one point, and it makes permission management easier and more flexible in distributed environment. The process of decision-making is analyzed and methods of solving conflicts that results from decision-making are proposed at the same time. A concrete sample is cited so as to verify the feasibility of this model.

Keywords—access control; centralized authorization; decision-making arithmetic

I. INTRODUCTION

With the further development of information process, there exists many ‘information islands’ in enterprises and the government, information integration has become the serious problem nowadays, secure access control is a problem that cannot be ignored. Nowadays there are many researches on access control in distributed environment[1-5]. Paper [1] studies the problem of role-spliting during role-mapping of inter-domain based on RBAC. Paper[2,3] design and implement practical RBAC access control middleware, which enhances the access control module to the height of RBAC middleware, and takes centralized access control in distributed environment. Paper [4] analyzes a typical secure access control technology based on middleware, which takes the user from system access list into the role in RBAC by increasing related conversion and management module, and achieve RBAC model easily. Paper [5] proposes a role-based and context-based access control model (RCBAC) in accordance with the needs of the request of unified resource access and the problems in RBAC model. In the paper [6], a permission centric hybrid access control model are proposed, the paper points out that there are different users and information resources in large organizations or information systems, different situations need different access control models, so different access control models are used in a system at the same time.

Nowadays researches on distributed access control mainly focus on new access control models or improving existing access control models, and then unified authorization is achieved on this base. When these models are applied to the integration of existing systems, because of the diversity of system access control models, it needs to carry out large amendments to original systems, and can not meet diverse needs of the access control model as well.

The paper presents a centralized authorization model to support multiple access control models, which achieves centralized authorization of multiple systems under the circumstances of not changing access control models of original systems.

II. CORE MODEL OF CENTRALIZED AUTHORIZATION

Different access control models may be used by those systems developed in different periods because of historical reasons, which result in the diversity of access control models. The centralized authorization model supporting multiple access control models presented in this paper achieves the centralized management to users and permissions, i.e., allocates user’ access permission to system resources rationally at one point, and implements access different resources of different systems further.

A. The Basic Definition

Definition 1. System-Domain (Domain): The entity is responsible for recording system-domain divided, every domain represents an application system. Existing and new application systems are defined as different domains, including domain-ID, domain-name, domain-description and the access control model the domain used (ac_model).

The definitions of ‘User’, ‘Role’, ‘Permission’ and ‘Object’ are identical to the definitions of the corresponding entities in RBAC[6]. The relationships among those entities include: URA[6], PRA[6], RH[6], ACL[6], UDA, ODA, PDA, RDA.

Definition 2. UDA: UDA ⊆ Users×Domains, this is a many-to-many mapping user-to-domain assignment relation.
When the access control model (ac_model) of the system used is MAC, because the same user in different systems has different sensitivity levels, we set the property u_level in UDA, which figures the user' sensitivity level in the relevant domain.

Definition 3. ODA: ODA ⊆ Objects×Domains, this is a many-to-many mapping object-to-domain assignment relation. When the access control model (ac_model) of the system used is MAC, because the same user in different systems has different sensitivity levels, we set the property o_level in ODA, which figures the object' sensitivity level in the relevant domain.

Definition 4. PDA: PDA ⊆ Permissions×Domains, this is a many-to-many mapping permission-to-domain assignment relation.

Definition 5. RDA: RDA ⊆ Roles×Domains, this is a many-to-one mapping role-to-domain assignment relation. In other words, a role can only belongs to one system-domain.

Relational functions which can be used in query operations are defined as follows:

\[ \text{assigned_users: } (d: \text{Domains}) \rightarrow 2^{\text{Users}} \]
\[ \text{assigned_domains: } (u: \text{Users}) \rightarrow 2^{\text{Domains}} \]
\[ \text{assigned_permissions: } (d: \text{Domains}) \rightarrow 2^{\text{Permissions}} \]
\[ \text{assigned_roles: } (d: \text{Domains}) \rightarrow 2^{\text{Roles}} \]

B. The Authorization Process

The authorization process of the centralized authorization model is described as follows:

Step 1: Add the user Useri, and assign the corresponding set of system-domains for the user Useri, then we grant permissions for Useri in every domain the user belongs to, and a user can belong to multiple system-domains.

Step 2: Get domains assigned to Useri through assigned_domains: (User), then get the first domain, and then mark it as Domainj.

Step 3: Grant permissions for the user Useri in the system-domain Domainj.

Switch (ac_model of Domainj)

Case RBAC:

IF role assignment satisfies ‘Separation of Duty’, ‘Role Cardinality Constraints’ and other constraints

THEN assign the role Rolek for the user Useri in URA.

Case ACL: assign permission Permissionn for Useri in ACLIST.

Case MAC: set up u_level of Useri in UDA, that is, the sensitivity level of Useri in Domainj.

Step 4: IF authorization for the user Useri in all system-domains it belongs to is completed

THEN user authorization is completed.

ELSE search the next system-domain of Useri, and mark it as Domainm, then go to Step 3.

In order to adapt to the distributed heterogeneous environment of application systems, the information of centralized authorization are stored in the form of XML documents, at the same time, the security of XML documents is of great significance, we can use classic encryption algorithm to encrypt the XML data, but also encrypt the communication protocol, we achieve the transmission security of XML data considering network security protocol, and we also can use other safety-assisted strategy, such as digital signature to documents, the validity period of XML data.

C. Application in Information Integration

Achieving centralized authorization in the process of information integration can reduce the burden of the administrator, and ensure system’s security. For existing information systems integrated, their user and permission information is mostly stored in different types of relational databases, having distributed heterogeneous characteristics. Realizing centralized authorization in existing systems, it is important to do a synchronization job of the user information and the permission information between centralized authorization module and original systems, which is shown in Figure 1.

The authorization information of various systems is stored in Centralized Authorization Platform(CAP). Functions of these parts are introduced as follows:

\[ \text{Centralized Authorization Platform} \]
\[ \text{UUMAP} \]
\[ \text{Mediator} \]
\[ \text{XML} \]
\[ \text{ACL} \]
\[ \text{MAC} \]
\[ \text{RBAC} \]
\[ \text{XML-RDB} \]
\[ \text{RDB} \]

Figure 1. Diagram of centralized authorization in information integration

UUMAP(User Map): The mapping relationship of the user between CAP and application systems, which achieves synchronization of the user information.

Mediator: The mapping relationship between the XML Schema of centralized authorization and the XML Schema
of local authorization module, which indicates parsing information of centralized authorization into authorization information of local systems in the form of XML.

XML-RDB module: It implements bi-directional mapping between the xml document and the relational database, making the local authorization information stored in the form of XML synchronized to the right management module of the original system.

Steps of authorizing for various systems through the CAP are explained by the following example, the ac_model of the domain d is RBAC, d represents the application system A, add the role r to u in the domain d: ①add the role r to u in the domain d at CAP; ②search user-name in A that is corresponding with the user u through UUMAP, naming it as uA; ③user and permission information will be parsed to the authorization information of the local system through mediator, it will be stored as the authorization information for uA in the form of XML; ④Synchronize it to the permission module of the system A through XML-RDB.

III. DESIGN OF CONTROLLER

Access controller permits or denies users’ access request based on the authorization data. The core of access controller is the access control decision-making algorithm.

A. Design of Decision-making Algorithm

The decision-making algorithm for centralized authorization model which supports multiple access control models is as follows.

Input of algorithm: Request (user, object, action).

Output of algorithm: Result ∈ {Permit, Deny}.

Then decision-making algorithm is as follows:

If (user ≠ User) return Deny;

If (object, action) is a valid permission take the permission_id of (object, action) from Permission into permission;

Else return Deny;

//get related domains and their access control model take the domain_id from PDA into domain;

take the ac_model of domain from Domain into ac_model;

//case of ACL
If (ac_model=ACL){
    If (user, permission) ∈ ACL list
        return Permit;
    Else
        return Deny;
}

//case of MAC
If (ac_model=MAC){
    //get the sensitivity level of user and object in this domain take u_level from UDA where domain_id= domain and object_id= object;
    If (action=Read)
        If (user, u_level > object, o_level)
            return Permit;
        Else
            return Deny;
    If (action=Write)
        If (user, u_level <= object, o_level)
            return Permit;
        Else
            return Deny;
}

//case of RBAC
If (ac_model=RBAC)
    take the activated role for user from URA set role;
    take the child roles of role from RRA into set assigned_roles;
    add role into set assigned_roles;
    while (assigned_roles ≠ ∅ )
        take a next role from assigned_roles;
        If ((role, permission) ∈ PRA
            return Permit;
        }
    return Deny;

B. Conflict Solution

The diversity of access control models and the distribution of information further exacerbates the complexity of decision-making process in distributed environment. For example, multiple systems have the problem of cross-authority, because of the differences of definition in different systems, there will have different corresponding decision-making results in different systems, while there has one final decision-making result for a user request, so the ADF needs to carry out conflict solution, in order to make a legal access to resources continue, the definition of conflict solution strategies are as follows:

(1)Deny Priority: This means that for multiple decision-making results of a user request to the resource, if one of those results is ‘Deny’, then the final decision-making result is ‘Deny’;

(2)Permit Priority: It means that for multiple decision-making results of a user request to the resource, if one of those results is ‘Permit’, then the final decision-making result is ‘Permit’;

(3)Application Priority: For the resource request of a user, the first decision-making result will be seen as the final result.
IV. CASE STUDY

We show how access controller work below by giving an example. The authorization data is built following entities and relationships proposed in section 2.1, which includes User, Object, Role, Permission, Domain, URA, PRA, ACL, UDA, ODA, PDA and RDA. Sample authorization data is shown in Figure 2.

<table>
<thead>
<tr>
<th>USER</th>
<th>ROLE</th>
<th>OBJECT</th>
<th>PERMISSION</th>
<th>DOMAIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wang</td>
<td>r1</td>
<td>File1</td>
<td>p1</td>
<td>d1</td>
</tr>
<tr>
<td>Li</td>
<td>r2</td>
<td>File2</td>
<td>p2</td>
<td>d2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PRA</th>
</tr>
</thead>
<tbody>
<tr>
<td>role_id</td>
</tr>
<tr>
<td>r1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_id</td>
</tr>
<tr>
<td>Wang</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ODA</th>
</tr>
</thead>
<tbody>
<tr>
<td>object_id</td>
</tr>
<tr>
<td>File1</td>
</tr>
</tbody>
</table>

Figure 2: Authorization data for sample environment

User requests may be permitted or denied by access controller depending on access control policies of every domain. Several cases of requests will be analyzed as follows.

Case 1: Request (Wang, File1, Read). User ‘Wang’ requests reading of object ‘File1’, permission p1 that contains (File1, Read) belongs to domain ‘d1’, ac_model of d1 is RBAC, user ‘Wang’ belongs to role ‘r1’, meanwhile, ‘r1’ contains permission p1. So access controller will permit this request.

Case 2: Request (Li, File3, Read). User ‘Li’ requests reading of object ‘File3’, permission p3 that contains (File3, Read) belongs to domain ‘d3’, ac_model of d3 isACL and ACLIST does not contain permission (Li, p3). So access controller will deny this request.

Case 3: Request (Wang, File2, Write). User ‘Wang’ requests writing of object ‘File2’, permission p2 that contains (File1, Read) belongs to domain ‘d1’ and ‘d2’, ac_model of d1 is RBAC, and the ac_model of d2 is MAC. User ‘Wang’ belongs to role ‘r1’, ‘r1’ contains permission p2, so access controller permits this request in domain ‘d1’; the ‘u_level’ of user ‘Wang’ in domain ‘d2’ is 80, the o_level of object ‘File2’ in domain ‘d2’ is 70, u_level>o_level, so access controller denies this request in domain ‘d2’. This decision-making leads to conflict results. We use different conflict solutions, ①‘Deny Priority’: the last result is denying this access request; ②‘Permit Priority’: the last result is permitting this access request; ③‘First Application Priority’: because decision-making result of d1 is permission the last result is permitting this access request.

V. PERFORMANCE ANALYSIS

(1) The model supports multiple access control models.

(2) The model can be used to solve the problems of heterogeneity under the distributed environment system, which achieves centralized authorization of heterogeneous systems, at the same time, The model can minimize changes of existing systems and retain the independence of the original system; using the reusability of permission management module, a newly developed system is no longer need develop an independent permission management module, thereby reducing development costs, what is more, the model makes permission management much easier.

(3) The model does not change the access control model of original systems. For new systems, we can select the appropriate access control models in authorization management platform in accordance with requirement.

VI. CONCLUSION

In this paper, a centralized authorization model supporting multiple access control models is offered, the model is independent of specific applications, it is able to support a wide range of applications as long as we only need to carry out as few changes as possible. We adopt the access control with distributed centralized management, permission information is stored in XML documents, so the security of XML is important, which is the topic we need to study in future.

REFERENCES


[6] American National Standardor Information Technology–Role Based Access Control,
