I.J.Modern Education and Computer Science, **2011, 5, 47-53**
Published Online August 2011 in MECS (http://www.mecs-press.org/)

H-RBAC: A Hierarchical Access Control Model for SaaS Systems

Dancheng Li
Software College of Northeastern University, Shenyang, China
Email: ldc@mail.neu.edu.cn

Cheng Liu and Binsheng Liu
Software College of Northeastern University, Shenyang, China
Email: {lectery, lbs.neu}@gmail.com

***Abstract*—SaaS is a new way to deploy software as a hosted service and accessed over the Internet which means the customers don't need to maintain the software code and data on their own servers. So it's more important for SaaS systems to take security issues into account. Access control is a security mechanism that enables an authority to access to certain restricted areas and resources according to the permissions assigned to a user. Several access models have been proposed to realize the access control of single instance systems. However, most of the existing models couldn't address the following SaaS system problems: (1) role name conflicts (2) cross-level management (3) the isomerism of tenants' access control (4) temporal delegation constraints. This paper describes a hierarchical RBAC model called H-RBAC solves all the four problems of SaaS systems mentioned above. This model addresses the SaaS system access control in both system level and tenant level. It combines the advantages of RBDM and ARBAC97 model and introduces temporal constraints to SaaS access control model. In addition, a practical approach to implement the access control module for SaaS systems based on H-RBAC model is also proposed in this paper.**

***Index Terms*—H-RBAC, access control, SaaS, RBAC, hierarchical model, multi-tenant**

I. Introduction

SaaS is a new way to deploy software as a hosted service and accessed over the Internet [1], by which users can rent web-based software from the service providers to manage business activities instead of purchasing and maintaining software by themselves. At the same time, this new method brings new challenges to data security, consistency and integrity in SaaS systems. In order to make users be assured of the safety of important or confidential data, permission management and access control are particularly important in the process of SaaS system development.

Access control as an important part of security services is an essential measure to ensure the security of information system [2]. It is a defensive method to prevent unauthorized resource use, and make sure the system is used safely. Access control determines what the user can do, and what types of resources can be used, it can effectively prevent the invasion of illegal users and the unauthorized access to system resources of legal users.

There are three most widely recognized access control models: Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role Based Access Control (RBAC) [3]. MAC and RBAC are both non-discretionary. DAC is very flexible, but security is not strong, authorization management is complex. MAC is a more stringent access control method, but it is inflexible and not suitable for the large-scale application, because it carries into execution complexly [4]. RBAC is an access control strategy which is between DAC and MAC, it has a strong ability to represent the semantic meaning of the relationship between users [5]. It is not only able to express the "responsibility" in complex social organizations, but also to reduce the complexity of access control management. Now it has been widely accepted as an alternative to traditional discretionary and mandatory access controls [6, 7].

II. Related Researches

RBAC is an enabling technology for managing and enforcing security in large-scale and enterprise-wide systems. It was first proposed by David Ferraiolo and Rick Kuhn in 1992 [8]. RBAC model introduces the "role" concept between "user" and "permission", every user is related to one or more roles, one role is related to one or more permission, and the roles can be created or deleted according to needs. After that, professor Ravi Sandhu of George Mason University put forward the most famous RBAC96 model with his colleagues; they added the role hierarchy and assign constraints to RBAC model, the RBAC96 model can be divided to RBAC0, RBAC1, RBAC2, RBAC3 the four conceptual models [9]. RBAC0 is the base model which consists of users (U), roles (R), permissions (P), and sessions (S). This RBAC0 supports the least-privilege principle. A user belonging to several roles can invoke any subset of them that enables tasks to be accomplished in a session. Thus, a user who is a member of a powerful role can normally keep this role deactivated and explicitly activate it when needed. RBAC1 introduces role hierarchies (RH) to the base RBAC model. Role hierarchies are invariably discussed along with roles in the literature [10] and they are

commonly implemented in systems that provide roles. RBAC2 is unchanged from RBAC0 except for requiring that there be constraints to determine the acceptability of various components of RBAC0. Only acceptable values will be permitted. RBAC3 combines both RBAC1 and RBAC2 and provides role hierarchies and constraints [9]. But in modern large enterprise wide systems, there may be a large number of roles and many users. The relationships among the roles, permissions and users change continuously, so the previous centralized RBAC models have several drawbacks to do access control in this situation. In 1997, Sandho and Bhamidipati raised the ARBAC97 model which consists of URA97 (User-Role Assignment '97), PRA97 (Permission-Role Assignment '97), and RRA97 (Role-Role Assignment '97). They based on the basic idea that using RBAC to manage RBAC and further to provide administrative convenience and scalability, especially in decentralizing administrative authority, responsibility, and chores [11]. After that, they have extended the ARBAC97 model to ARBAC99 where they separate users/permissions into mobile and immobile users/permissions [12], and later to ARBAC02, where they use an organization structure to define user-role assignment and role-permission assignment [13]. ARBAC02 contains the main features of ARBAC97, and add the concept "organization" to improve many imperfect aspects in ARBAC97.

In RBAC, permissions are associated with roles, and roles are assigned to users. With the increment of roles and permissions, we need to reassign some roles from one user to another in short term or long term to make sure the business activities performed normally. And also permissions can be revoked from roles as needed. All the roles delegation actions were performed via the administrator's temporary user-role configuration before. But with the development of business, the number of roles in a company is increasing all the time. If we use the old method to achieve the delegation work, the system administrator will face with so heavy burden that some of the delegations will not be accomplished in time. And this practice will affect the efficiency of enterprises directly. So we need a mechanism to realize the user-role reassignments in enterprise systems. Based on this theory, Barka and Sandhu proposed the RBDM0 model [14] which introduced the "delegation" to the traditional RBAC model. In RBDM0, a user has the ability to authorize the assigned roles to another user to help him perform the works. But the RBDM0 model is so simple that it's hard to be used in the real enterprise applications directly. In 2005, they did some improvement to the previous model and put forward the RBDM1 model [15]. In RBDM1, hierarchical roles were added to the basic model, like that happened in RBAC1. This change makes the model more practical in actual application environment. But there are still many problems that the RBDM models didn't take into account, such as the temporal constraints to the roles delegation, multistep delegation and the delegated roles revocation mechanism. In 2000, Longhua Zhang's team proposed the RDM2000 model based on the basic RBDM models. This model solves the problems of the hierarchical delegation and multistep delegation effectively. But RDM2000 still doesn't bring the temporal constraints to access control model.

Researchers and vendors have proposed many enhancements of RBAC models in the past decades. Chen Nanping and his colleagues proposed a RBAC model with www extends, they added role proxy layer between users and roles to implement role assign dynamically, and improve the efficiency of network transmission [16]. Xia Luning and his team proposed the N-RBAC, a hierarchical namespace-based RBAC model. They used namespace to organize roles and resources in order to simplify the complexity of the role hierarchy structure [17]. Ma Lilin and Li Hong did some research on admission control, operation control, data access control of SaaS systems, but they didn't do deeply research on access control and didn't provide a feasible implementation of access control module [18].

The features of multi-tenant, configurability and security make SaaS systems so different with the traditional systems. If we apply existed access control models to SaaS systems directly, the following problems will appear:

A. Role Name Conflicts

In SaaS applications, there are always a large number of tenants using the system services at the same time. Each tenant usually needs very large number of roles, which means that there are many nodes in the roles hierarchy structure. However, these nodes cannot have duplicate names, so we have to take measures to modify role names to avoid naming conflicts, such as adding a prefix to the role names. In this way, the roles inheritance become more complex, and the roles' names will be longer.

B. Cross-level Management

According to the regulations of ARBAC97, the system administrators inherit the permissions of all the following administrators in different tenants. This means that system administrators can do some changes within each tenant's permission scope and can ignore the "can_assign" constraint in URA97. But in fact, a tenant is usually an independent company or a department of a big business. That means that each tenant must be an autonomous unit. The tenant's resources such as permissions, roles, users etc. must be managed by the system administrator of the tenant instead of the SaaS system administrator which is usually a staff of the SaaS service provider.

C. The Isomerism of Tenants' Access Control.

SaaS System is a multi-tenant online rental system, although in theory the tenants should belong to a same field, but from the practical point of view, each tenant's development is uneven, and thus there're many differences in access control among the various tenants. The differences are mainly as follows:

1) The Different Control Scope of Permissions

Since the tenants of different sizes have distinct access control needs, the control scopes of permissions among all the tenants are not the same. Some tenants would like to restrict all the resources, while some only want to do access control on certain permissions. Even some tenants do not need the access control mechanism which means all the members in the tenant can access all the functions and data. All the above exist objectively in the real world.

2) The Heterogeneous Relations of Roles

Since the departments and jobs are different within different tenants, the roles and role-permission relations in different tenant are different. If the traditional RBAC model is used in SaaS systems, all roles will be defined in global scope, they are visible to any tenants. This will not only bring the naming conflicts problems, but also it's inconvenient to manage the roles within a tenant.

3) The Heterogeneous Constraints of Permission Assignments

Each tenant in SaaS system is a highly autonomous entity, so the constraints of role assignments in different tenants are different. If we define the constraints of role assignments in the global scope, this will bring the assignment constraints conflicts, which may bring confusion in the roles relationship management.

D. Temporal Role Delegations

Sandhu has formally defined the role-based delegation model based on hierarchical relationship between the roles involved to realize the roles delegation between users. According to the enterprise actual needs, the access control model should include the constraints or rules of roles delegation and revocation. These limitations may include whether to allow the original user revoke the permissions from delegated user directly or get back the roles after the authorized users' use. If we allow the directly revocation, the mandatory operations may bring system data loss or the system may face the data consistency problem. Because some functions may be in operating state when the revocation performing. In addition, time constraint is also an indispensable part of role delegation model. After a user delegate some roles to another user, the original user should also define the permissions' valid period to prevent the abuse of authority.

The temporal constraint based delegation in SaaS systems includes two levels: the system level and the tenant level. The system level time constraints are the constraints that SaaS services providers define to limit the valid period of SaaS services according to how much the tenants pay for them, such as the system expired time constraints. The tenant level time constraints are the limitations that the tenant system administrator formulates to assign available permissions to system users reasonably after the tenant gets the use authorization of SaaS services and further to ensure the system security. However, all the existing models don't meet all the requirements of SaaS systems' temporal roles delegation.

In SaaS systems, the services every tenant has belong to different instances of system though, but all these services are deployed at one time. Taking into account that this article is focused on level-3 SaaS systems, therefore, when designing the access control model, the purpose is very clear that all the users and tenants' information must be stored in the same database server, so there are high demands in the clarity of the access control model. In addition, taking into account the upgrade and maintenance, the access control model the systems use should have some scalability. The models mentioned above are all unable to fully meet the three conditions that SaaS systems access control model must satisfy.

Therefore, this article proposed a hierarchical access control model named with H-RBAC for SaaS systems after summarizing advantages and disadvantages of the existing access control models. This model solves the access control problems from both system and tenant perspective.

III. H-RBAC MODEL

This paper proposes a RBAC model based on hierarchy structure, defines the management scope from both system and tenant point. In a SaaS system, each tenant has its own users with the features distinct from others', so that each tenant should have its own access control policy and administrative scope. Therefore, SaaS systems need not only provide access control for tenants, but also provide users' access control within every tenant. On the system level, the target objects of access control model to address are the tenants, not the detailed users. So, from the whole system perspective, the access control in SaaS systems should contain several sub-access controls, that means system access control include tenant access control, tenant access control is based on the system access control.

Above all, we propose the H-RBAC model, it improves tenant-based RBAC model, the static model as shown in Fig. 1. Furthermore, this paper realizes the access control in SaaS systems from two aspects: tenant-level access control and system-level access control. The tenant-level access control uses the organization-based access control model, and the system-level access control uses the administrative role-based access control model based on ARBAC97 which provide the mechanism to distinguish administration roles from general roles. In addition, we take the roles delegation into account so the H-RBAC model combines the advantages of ARBAC97 model and RBDM and we also do several improvements based on them.

A. H-RBAC Basic Structure

We can see the hierarchical structure of H-RBAC model from Fig. 1 clearly. The top level elements are all belong to the SaaS system provider. These elements are used to implement the tenant authorization. While the bottom elements are belong to every tenant, and used to realize the access control in a tenant. The H-RBAC model contains the following elements:

- System User: it represents all the individuals that use the SaaS system directly, such as the system administrators in SaaS services provider companies and the clients (tenants) that rent the system services from the services provider.

● AURC: Administrative User-Role Constraint, it defines the rules that apply to the user-roles assignments. Since in the actual SaaS system services provider company, there must be a large number of administrative tasks to be performed, the top level administrator may divide the tasks to many other administrators. Each administrator can only use the permissions that assigned to him. The AURC ensure the administrator doesn't have the rights that not be assigned and ensure one administrator will not get the exclusive permissions.

● APRC: Administrative Permission-Role Constraint, it is used to separate duties, to ensure different roles own different permissions.

● TTC: Tenant Time Constraint, it represents all the time related rules that SaaS system providers define and to be applied to the tenants, such as the tenant authorization activation time constraints, the tenant authorization valid time period constraints and so on.

● User: all the individuals that use the system services directly, including both the persons and other entities, such as the personal computers, agents, networks etc.

● Role: a job function within the organization that describes the authority and responsibility conferred on a user assigned to the role.

● Permission: a description of the operation types of authorized interactions a subject can have with one or more objects.

● RE: short for resource, anything used or consumed while performing a function in the system. The resources can be divided into several categories, such as time, information, objects, and processors etc.

● Role Hierarchy: a partial order relationship established among roles.

● User Group: a set of users that are in the same department in the tenant company or have the similar duties. In the system, the user group can be regarded as an access control unit. A major difference between groups and roles is that groups are typically treated as a collection of users but not as a collection of permissions. A role, serving as an intermediary, is both a collection of users and a collection of permissions.

● URTC: User-Role Time Constraint, the limitations that used to define the assigned roles valid activation time and the roles available time period for a user.

● URDC: User-Role Dynamic Constraint, it is a kind of dynamic duties separation constraints, which is used to avoid assigning overmuch permissions to one user. The URDC is only act on the permissions activated in the current session to ensure the exclusive roles that defined in the URDC will not be assigned to one user. It's established when a user logs in the system, and if the user logs out, the constraints will become invalid.

● General Permission: a description of the type of authorized interactions a subject can have with an object in the system. The permissions include the resources and operations. The resources include all resource entities that the system functions need, such as data tables, properties etc. The operations contain all the actions of accessing the system resources, such as database queries, update, delete and modification and so on.

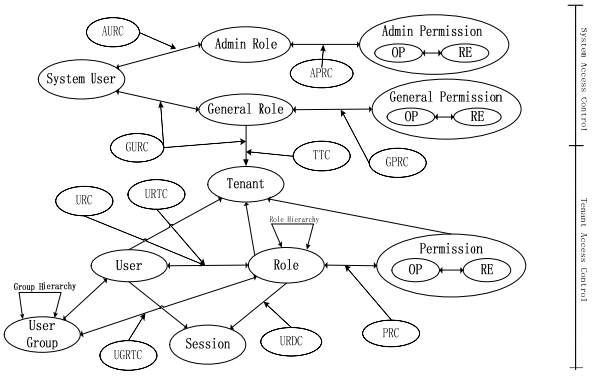

Figure 1. H-RBAC model.

● General Role: in SaaS systems, it represents the release suits that the service provider provide for renting, such as the standard edition, advanced edition, starter edition etc.

● Admin Permission: express the permission to operate the model itself, such as adding a new role, deleting a user etc.

● Admin Role: it corresponds to the jobs in the system whose responsibility is managing the access control model itself. Such as the permission to modify the set of users, roles, permissions and modify the user assignments or permission assignment relations are all included in the administrative jobs.

● GPRC: in real life, certain functions cannot be assigned to the same role, that is the separation of duties, which is aim to divide different skills and different interests to different kinds of people in order to prevent or reduce the chance of fraud and cut down the loss made by mistakes. General Permission-Role Constraints define some rules to avoid a role has some exclusive permissions.

● GURC: General User-Role Constraint, it defines the authorize rules between the roles and users.

● Admin Constraint: the "admin" means the management of access control model，including the user-role assign management, permission-role assign management etc. It ensures the separation of exclusive administrative duties.

● Tenant: the client of SaaS systems, tenants pay on demand to SaaS services providers. Each tenant requires SaaS system to ensure a high degree isolation of data and configuration to ensure the security and privacy also requires the customization of user interface, business logic and data structures etc. In practice, each tenant is the form of enterprise, so each tenant can have many users.

● Session: a session is a mapping from a user to multiple roles, a session is established when the user activates some or all authorized roles. What the user can do is within the tasks set activated during this session.

● URC: User-Role Constraint, it defines the authorization rules between the users and roles in a tenant, to avoid assign the exclusive roles to the same user.

● PRC: Permission-Role Constraint, it defines the assign rules between the permissions and roles within a tenant, to avoid the exclusive permissions authorized to the same role.

● UGRTC: User Group-Role Time Constraint, it is the limitations or rules of user group-role assignment. That

defines the available time or period that the roles act on a user group.

The concepts not mentioned above are exactly the same as the corresponding ones in classical ARBAC97 model. As mentioned above, the differences between H-RBAC model and ARBAC97 model are as follows: the user resources are not defined in the global scope, but in the different tenant namespaces, in this way, we can effectively solve the roles naming conflicts problem. Meanwhile, in a tenant, the administrator can define and distribute all the roles and permissions within the tenant on demands, which solves the inconsistent permission needs problems among different tenants. In the H-RBAC model, each tenant is an autonomous entity. Therefore inheritance exists between different roles, and implementation of inheritance is similar to that in ARBAC97. In ARBAC97, the inheritance of roles for administrator is aiming to facilitate the distributed management of permissions, however, in H-RBAC, the distributed management is implemented via the way of tenant autonomy, and the permissions in one tenant will not spread to other tenants. So in the system permission management level of H-RBAC, the system administrators will neither inherit the permissions of tenant administrators anymore nor be allowed to interfere with the permission assignments within the tenant scope. That means the tenants are regarded as "black box" at the system level, and the permissions in a tenant are managed by the tenant administrator.

B. Time Constraints in H-RBAC

In the actual environment the time constraints mainly include two aspects: the delegation starting time constraints and the delegation duration constraints.

The basic delegation model consists of three parts: (S, O, R). S means subject user who initiate the delegation action. O means object user who is to accept the delegation role. R means the delegation role in this action.

Whether a role is valid at some time is related to the state of the role. In the H-RBAC model, we define that every role has three states based on the using condition. Three states are assigned, activated and disabled as shown in Fig 2.

We define two levels time constraints in the SaaS system delegation. They are the system level constraints and tenant level constraints.

The system level time constraints' target objects are the tenants. These constraints focus on controlling the system available duration. Because if a company want to use SaaS services, it must pay some money to the SaaS system providers to get a fixed services available duration. So the SaaS system must provide a mechanism to stop the services being used once the services are expired. And this mechanism can be implemented by the system level time constraints in H-RBAC. Before we introduce the time constraints, the tenant authorization in SaaS system is in the form of (t, r). The t represents a tenant, r means the role assigned to the tenant. In H-RBAC, we add the valid time duration to it, the form of temporal tenant authorization is (t, r, d). The d is a time interval [ts, te], in which the ts means the start time of the authorization effect and the te means the authorization expired time. The new delegation expression means assign the role r to tenant t, and the delegation relation is valid only during the time period of d. If the current time past te, the SaaS system will withdraw the assigned role r from t.

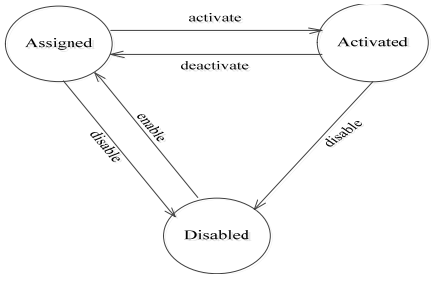

Figure 2. Role state transition.

After a role assigned to a user, the user can use the assigned rights to perform some business functions. But the user may need others' help in practice and these helps always relate to some confidential data or functions. So we define the tenant level time constraints to put some limit to the delegations. These constraints concentrate on providing the temporal limitations to all the users and user groups in the tenant scope. According to the actual needs, we divide the tenant level time constraints into three types:

1) Activation time constraint

It defines the valid roles' activation time period, which means the assigned roles should be only activated in a specified duration. For example, in some companies the stuffs can only use the system during the worktime, or some peculiar functions can be only activated in a specified time period.

2) Available duration constraint:

This constraint limit certain assigned roles of a user can be only activated for a fixed duration every time. Its goal is to protect the important or confidential operations from being embezzled because of the too long activation.

3) Available times constraint:

The available times constraint contains both the available duration constraint and the number limit of uses. It ensures that certain roles can be only activated fixed times during the specified duration.

IV. ACCESS CONTROL MODULE ARCHITECTURE

This paper applies H-RBAC model to the access control module of the community health services system based on SaaS, which provides some basic functions for small or medium sized community health organizations, including registration management, medical record management, outpatient clinic, pharmacy management etc. The system follows the SaaS patterns, provides its services in the way of single instance and multi-tenant structure. In view of the system involves a number of business units, roles assignments are complex, there are too many kinds of constraints and a wide range of other factors, we use the H-RBAC model to achieve the roles

management, roles allocation, dynamic constraint access control, dynamic permissions distribution and other access control functions. Concrete realization of the access control module is shown in Fig. 3.

The access control module is composed of the following components: ACS (Access Control Server), AFS (Access Filter Server), UDCS (User Dynamic Constraints Server), PMC (Permission Management Center), AUC (Authentication Center) etc. Following are the detailed description of each section.

A. Authentication Center

The authentication center is a function to authenticate each user that attempts to access to the system services. Only the users passed the authentication can send further requests to the system.

B. Access Filter Server

Access Filter Server is equivalent to a control switch, it filter access requests by using the filter configuration and the capability list generated by ACS. If a user has the specific permission, the AFS will forward the request to the application server, if doesn't have, the AFS will intercept the request and return the failure messages.

C. Access Control Server

Access Control Server forwards the requests to the tenant-level access control module, it implements the Access(S,O,OP) function which determine whether this session S has the operation OP on the object O. It use the capability list (CL) as the control method, attached the CL to the current session as an attribute.

D. User Dynamic Constraints Server

In the process of generating the capability list, the access control module must ensure the capability set does not include the exclusive abilities. But exclusive relationships are defined in the role-user and the role-permission constraints. The UDCS generates the permission set which cannot be assigned to the specific user based on the user information submitted by the ACS and the related constraints.

E. Temporal Constraints Server

It is used to handle all the operations related to the time constraints. The TCS get the role set or permission set after the UDCS dispose. Then it gets the related time constraints from the Access Control Database according to the identification included in the previous step's result. The TCS realizes the process of time validation for every role to be assigned to the user. It ensures that the expired roles or permissions will not be authorized to any user.

F. System Management Server

The SMS (System Management Server) provide an entrance of SaaS system management for the system top level administrators. These administrators usually belong to the SaaS service providers and are responsible for all the top level system management works, such as the tenant authorization, system parameters settings and system functions management etc.

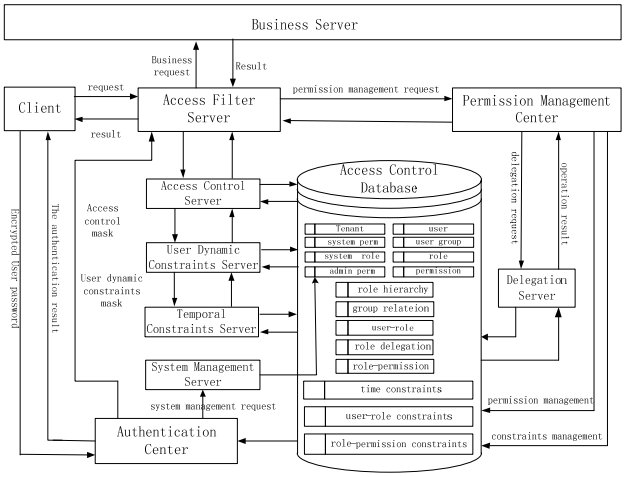

Figure 3. The structure of access control module

G. Delegation Server

The DS (Delegation Server) is used to handle all the roles delegation requests in the system. According to the user's assignment, the DS transfer the input from the UI page to formal expressions that can be easily stored and queried in the database.

H. Permission Management Center

Permission Management Center provides the permission management operations for system administrators, including system-level management and tenant-level management. The system-level management provides the functions from the view of system administrators it provides some management operations based on tenants, such as tenant management, service suit management, system permission management etc. While the tenant-level management achieves the management functions within a tenant, it involves the user management, permission management, constraint management etc.

The client initiated authentication request to access control module, then the module will send the user information to the Authentication Center, after the success of authentication, the center will returns the successful authentication access control mask, the user dynamic constraint code and other information and save them to the session related to this user. The Access Filter Server encapsulates the user's information and the authentication result into another data structure and sends it to the Access Control Server which can generate the permission constraints according to the request parameter, and then the ACS generates the capability list of this user based on the constraints and user role information. At last, the AFS validates the business requests according to the capability list and forward the valid requests to the application server.

V. CONCLUSION AND FUTURE WORK

This paper proposed a hierarchical access control model for SaaS systems named with H-RBAC. And we raised a practical implementation of access control module for SaaS systems based on the H-RBAC model. First we introduced the basic concepts about SaaS and access control methods. Followed by related researches on the RBAC-based access control, this paper analyzed the

advantages and disadvantages of existing RBAC models. Then we raised the H-RBAC model which solves the access control problem in SaaS systems. Finally, we put forward a practical way of access control module implementation for SaaS systems. Practice shows that the access control based on H-RBAC model is practical and it has several advantages**:** (1) flexible structure, conducive to hierarchical responsibility segments and authority management. (2) self-government within tenants, without overemphasizing the role hierarchies of multiple levels. (3) intuitive permission assignments, easy to understand and use for SaaS system developers. (4) good scalability, supporting the needs of different tenants of heterogeneous access control.

However, the method proposed in this paper has some imperfections, so we need to do more in-depth research in the future. For example, we'd better take the system security and data consistency into account when designing models. Because when the roles revocation proceeding we must guarantee the services are available and will not be interrupted. Another direction is promoting the efficiency of the access control. Since the permission related operations are performed frequently in SaaS systems, we can provide an optimization method based on H-RBAC model to ensure the system efficiency.

ACKNOWLEDGMENT

This Work was supported by Natural Science Foundation of Liaoning Province. (No.20092006)

REFERENCES

[1] Frederick Chong, Gianpaolo Carraro, Architecture Strategies for Catching the Long Tail, http://msdn.microsoft.com/enus/architecture/aa479069.aspx, 2006, 4.

[2] Messaoud Benantar, Access Control Systems: Security, Identity, Management and Trust Models, Springer US, 2009, 12.

[3] Bo Lang, Ian Foster, Frank Siebenlist, Rachana Ananthakrishnan, A Flexible Attribute Based Access Control Method for Grid Computing, Journal of Grid Computing, vol.7, pp.169-180.

[4] Jiang Yueqiu, Jiao Yan, Research and Implementation of Access control Model of Military Information System, Acta Armamentarii, 2009, 4, pp.431-437.

[5] Feng Demin, Wang Xiaoming, Zhao Zongtao, An Expanded Role-Based Access Control Model, COMPUTER ENGINEERING AND APPLICATIONS, 2003.

[6] Liu Peishun, He Dake, Application of RBAC in the Railway Passenger Ticket Network Security System, JOURNAL OF THE CHINA RAILWAY SOCIETY, 2004.

[7] J. Bacon, K. Moody, Toward open, secure, widely distributed services, Communications of the ACM - Adaptive middleware, vol. 45, 2002.

[8] David Ferraiolo and Richard Kuhn, Role-Based Access Controls, Reprinted from15th National Computer Security Conference, 1992, pp.554-563.

[9] R. S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman, Role-Based Access Control Models, IEEE Computer, IEEE Press, 1996, pp.38-47.

[10] S.H. von Solms and I. van der Menve, The Management of Computer Security Profiles Using a Role-Oriented Approach, Computers & Security, *v*ol. 13, No. 8, 1994, pp. 673-680.

[11] R. Sandhu, V. Bhamidipati, and Q. Munawer, The ARBAC97 Model for Role-Based Administration of Roles, ACM Transactions on Information and System Security (TISSEC), vol. 2, 1999, pp. 105-135.

[12] R. Sandhu and Q. Munawer. The ARBAC99 Model for Administration of Roles, In Proceedings of 15th Computer Security Applications Conference, 1999, 2, pp. 229.

[13] S. Oh, R. Sandhu, A model for role administration using organization structure, Proceedings of the 7th ACM symposium on Access control models and technologies, Monterey, 2002.

[14] E. Barka and R. Sandhu. A role-based delegation model and some extensions. In 23rd National Information Systems Security Conference, Baltimore, MD, October 2000.

[15] Barka, R. Sandhu, Role-based delegation model/hierarchical roles (RBDM1), Computer Security Applications Conference, 2004, pp.396 – 404.

[16] Chen Nanping, Chen Chuanbo, Implementing role based access control in WWW environment, Journal of Huazhong University of science and technology, 2002.

[17] Xia Luning, Jing Jiwu. An Administrative Model for Role-Based Access Control Using Hierarchical Namespace. Journal of computer research and development. 2007.

[18] Ma Lilin, Li Hong, A permission model of SaaS system based on RBAC, Computer application and software, 2010.

Dancheng Li was born in Shenyang, Liaoning province in 1963, earned M.S. degree in the field of computer software in 1990 from Shenyang Institute of Computing Technology, Chinese Academy of Sciences.

She is now an associate professor and postgraduate supervisor in Software College of Northeastern University, China (NEU). Before joining NEU, she was an associate research fellow in Shenyang Institute of Automation, Chinese Academy of sciences for about 3 years. Her main research directions include IT service management and information system engineering.

Cheng Liu, born in Shenyang, Liaoning province in 1988, earned B.S degree in the field of software engineering in 2010 from Northeastern University, China. Now he is a postgraduate student major in computer software and theory in Northeastern University, China.

Binsheng Liu was born in China in 1987. He received the bachelor degree in software engineering in Northeastern University, China in 2010. He is recently a postgraduate student in Northeastern University, China.

