2016 7th International Conference on Cloud Computing and Big Data

Design and Implementation of A Role-Based Access Control for Categorized Resource in Smart Community Systems

Siping Shi[†‡] and Chengzhong Xu[†§]
[†]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[‡]Shenzhen College of Advanced Technology, University of Chinese Academy of Sciences
[§]Department of Electrical and Computer Engineering, Wayne State University, USA
{sp.shi, cz.xu}@siat.ac.cn

***Abstract*—With the progressive development of smart communities, security of smart community systems becomes an important issue. Role-Based Access control is a way to solve this problem. However, existing implementations of role-based access control are not fine-grained, and it takes no account of category information of the resources. As every resource in the model was authorized in the same way, such a model cannot meet the security requirement of smart community systems. In this paper, we proposed an improved role-based access control model for categorized resources, in combination with special requirements of smart community systems. We designed a role-based access control model for categorized resources, which integrates the community category information in the definition of roles so as to limit the number of roles. The new model was fully implemented in a community management system with 14500 users from 14 communities. We compared our system to Spring Security, an existing security open source framework and demonstrated the advantages of our access control model.**

I. Introduction

Smart community becomes a new concept of community management in recent years, as an innovative model of social management. A smart community system is an integrated platform which makes full use of the Internet of things, cloud computing, mobile Internet and other new generation of information technology. It provides a safe, comfortable, convenient, modern and intelligent living environment for residents in a community. With the increasing number of users who have access to the system, we expect the users can have access only to authorized resources of authorized communities. Traditional role-based access control model is too coarse-grained, it cannot meet this requirement. It is highly demanded that we should design a proper access control model for the smart community system.

Role-based access control is a widely used access control model in applications. However, there are limitations to imply the access control in the smart community system based on the predominant role-based mechanism. For example, one of the security requirements is that the user can have access only to the authorized resources of authorized communities. In a traditional way, if the number of communities is N, in order to imply this requirement, the number of roles we should create for the authorized resource is N. That is, every community in the system has a role related to the same resource. As the number of communities grows large, the management of the system role becomes more and more difficult, and causes a waste of storage space of the smart community system. So we should improve the role-based access control model to overcome this limitation and adapt the requirements of the smart community system.

In this paper we develop a new approach to apply the role-based access control for considering the categorized resource to meet the security requirements in smart community systems. Our design is a mixture of the role-based access control and the UNIXs access control mechanism [1] for reference in the authorization progress. In our design, we classify the system resources into three categories inspired by a task-role based access control model [6] widely used in enterprise environments. We also refer to the proposals for relationship-based access control model [2], [3], [4], [5] which is widely used by social networks. We believe that our model provide a more practical and applicable approach for role-based access control mechanism than existing proposals of role-based access control for the smart community system.

Our model introduces several contributions. The most significant one is the designing of the role-based access control for categorized resource. This improvement can also be used in scenarios which are similar to the smart community system. The time cost of authorization process in our designing is less than existing proposals for role-based access control model because of using bit operation in our authentication algorithm, which refers to the UNIX's access control mechanism. Our model can support more users and resources, and allow for a fine-grained authorization policy. It is powerful and scalable.

In the next section, we provide an overview of the related work of role-based access control model and task-role based access control model. In Section III, we describe the design and of our access control model, including basic ideas, some notations, framework of our designed model and authorization mechanism. In Section IV, we introduce the background of our study case and imply our model in the smart community system of the case and evaluate the implementation with a comparison of Spring Security. Then, we conclude the paper

978-1-5090-3555-7/16 $31.00 © 2016 IEEE
DOI 10.1109/CCBD.2016.45

with a summary of our contributions and ideas for future work.

II. Related Work

Role-Based access control (RBAC) [7] is widely used and has been the subject of extensive research in recent years [8], [9]. It assigns a user to one or more organizational roles. These roles are then authorized to perform certain actions on particular resources, represented by permissions, as shown in Figure 1. A role is primarily a behavioral concept: roles give an idea of what users can do in the application. However, the RBAC model is not as ”fine-grained” as its supporters claim. Some proposals are introduced to tackle these kinds of problem [10], [11]. In practical applications, there are two primary ways of actually implementing and performing access control: an implicit way, and an explicit way.

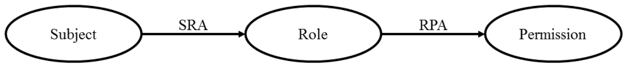

Fig. 1. Role-Based Access Control Model

In implicit access control, we actually do not know what a role is allowed to do; no one defined that behavioral statement anywhere in systems. This mechanism reduces the space and administrative complexity of access control, but it also greatly reduces the flexibility. Take for example the role ’Project Manager’, it is just a string name to developers.There is nothing else about it that a software program can inspect to find out what operation can be done if users assigned this role. If a user is allowed to view project reports, developers will typically code like Algorithm 1 and they write an if/else statement reflecting the assumption that users assigned the ’Project Manager’ role which can view project reports. Security access control like the above example is very brittle. That is, it is likely to break, fail, or cause inefficiencies with even slight changes to security requirements. Beyond that, the implicit access control also cannot support the ability to dynamically create or remove roles at runtime. To solve the above problem in implicit access control, the explicit access control was emerged.

Algorithm 1 Example Implicit Role-Based Access Control security check

```
1: if user.hasRole(”Project Manager”) then
2:     show the project report button;
3: else
4:     don’t show the button;
5: end if
```

In explicit access control, we are explicitly attributing a concrete behavioral statement about a specific resource instance with a user account. We protect resources as well as what actions a user could do to those resources. So the security protection is based on resource, when we break it down to this most primitive level and we can describe security policies in a much more fine-grained and change-resilient manner. For example, the above code block in Algorithm 1 can be done effectively with resource-based semantics, as coded like Algorithm 2. This example is much more explicit as to what access is being controlled. We are basically checking if the current user is permitted to ’view’ the ’project Report’ with ID ’12345’. The code is now based on what is being protected, not who might have an ability. We can see that the explicit way is apparently simple and perhaps more obvious conceptually. The explicit way is better and we design our model based on this.

Algorithm 2 Example Explicit Role-Based Access Control security check

```
1: if user.isPermitted(”projectReport:view:12345”) then
2:     show the project report button;
3: else
4:     don’t show the button;
5: end if
```

Task-Role based access control (T-RBAC) is a proper access control model for enterprise environments through the integration of the role-based access control and activity based access control (ABAC) models [12], [13]. This model proposes classifications for job functions. There are two central ideas in the T-RBAC. One is the classification of enterprise tasks (job functions) according to their characteristics. The other is to use intermediate tasks between access rights and roles instead of assigning access rights to roles. It solves the limitations of ABAC model and RBAC model.

However, neither RBAC model nor the T-RBAC model can meet the requirements in smart community systems separately. We combined these two models together and proposed a new model based on RBAC for categorized resources. Inspired by the classification of tasks in T-RBAC, we classify the resources in our model, and define two authorization policies for authorizing different categories of resources as referred to the UNIX Access Control model. Our new approach of implying role-based access control can completely fulfil the requirements of access control in smart community systems. We imply our model in smart community systems to prove this in Section IV.

III. Design of the Role-based Access Control for Categorized Resource

In this section, we introduce the design of role-based access control for categorized resource mentioned above in detail. First we introduce our basic ideas and present the notations used in the model. Then we describe the framework of our design. We introduce the authorization mechanism at last.

A. Basic Ideas

In a smart community system, some resources are related to community and only the user who has authority of the requested resource and the requested community can be allowed to access. Some resources are irrelevant to the community, if a user has the authority of requested resource, he/she can have access to the resource; the other resources are private and user can choose whether to share the resources with others in the smart community system or not.

According to the security requirements of the smart community system, we divided the resources into three categories: system resources, community related resources and private resources. Each of the categorized resource is associated with a resource category ID called CID, we assume the CID of system resources is 0, the CID of community related resources is 1, and the CID of private resources is 2. When we assign the resource to a role, the CID is assigned to the role together, so there are just three categories of roles in our system. Then we assign the role to the user, according to the CID of the role, we can determine whether we should assign the community Id to the user at the same time. If the CID is 0, we just need to relate role to the user without the community Id; if the CID is 1, we should relate the role and community Id to user together; if the CID is 2, we not only relate the role to this user, but also the Id of public or private should be related to user. It should be pointed out that one role can only related with one category of resources, but one user can be assigned multiple category of roles.

In our design, we just need to classify the resources and manage the relation information between user and role, the size of roles is manageable, and we don't need to manage the unpredictable size of roles' information as the number of communities is unpredictable in the traditional way. Our design reduces the storage space of the smart community system. It is scalable and powerful.

B. Notations

- Subject: That is the user of the smart community system. Basically, it is community administrator or anyone communicating with the system.
- Community: A certain area within the city, with a large residential independent living environment, and is equipped with a complete set of service facilities. A property management company may serves several communities.
- Role: A Role is a named entity that typically represents a set of behaviors or responsibilities. Those behaviors translate to things you can or can't do with a software application. Roles are typically assigned to user accounts, so by association, users can 'do' the things attributed to various roles.
- Resource: Whatever in system the user can access, such as JSP pages, the certain data and a functional module are all the resources. Users can only access the authorized resources.
- Permission: The atom authorized unit in security policy. It represents the authority that user has to operate a resource. For instance, the operation such as view, add, modify or delete user's data is permission of the user's data. Permission determines whether the operation on resource is allowed, it cannot determine who can perform this operation. So we should grant the permission to the user, that is defining which actions (permissions) on a resource are allowed to user.

C. Framework of the Access Control Model

Our model is an improved model based on the role-based access control for smart community systems. Fig. 2 shows a brief model of our access control. The difference between our design and the RBAC in Figure 1 is that we classify the resources into three categories. We also relate the community to the process of assigning role to subject in our model, so the role can be parameterized and fine-grained.

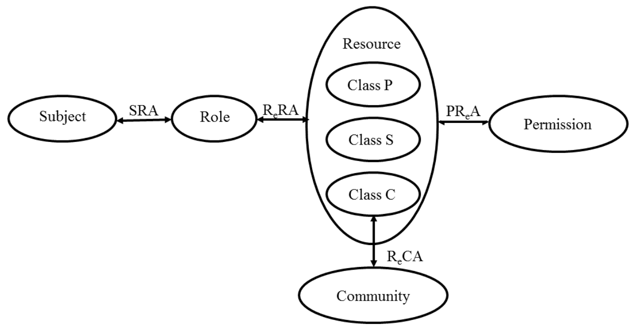

Fig. 2. The Model of Role-Based Access Control for Categorized Resource

We assume the existence of a set of users (subjects) U, a set of communities C, a set of roles R and a set of permissions P.

Property 1. Classification of resources. All of resources are classified into 3 classes as follows:

- Class C : community related resources
- Class S : system related resources
- Class P : private resources

R_C, a set of resources in Class C, R_S, a set of resources in Class S, R_P, a set of resources in Class P, a set of resource R_e:

$R_e = R_C \cup R_S \cup R_P$ and $R_C \cap R_S = \emptyset$, $R_S \cap R_P = \emptyset$, $R_C \cap R_P = \emptyset$

Property 2. Resource-Role Assignment. A Role has resources one or more. A Resource can be assigned to many roles. The resources must be in the same class.

$$R_eRA \subseteq R \times R_e$$

Property 3. Permission-Resource Assignment. A Resource has permissions to executing itself. Permission is defined as a collection of actions.

$$PR_eA \subseteq P \times R_e$$

Property 4. Role-Community-User Assignment. If the role assigned to a user which the related resources belong to Class C, the information of communities should be assigned to the user at the same time.

$$RCUA \subseteq R \times C \times U$$

There are two central ideas in our model. One is the classification of resources according to their characteristics. The other is to use intermediate resources between permissions and roles instead of assigning permissions to roles directly. These make the authorization more fine-grained and overcome the limitation of considering without the resources' classification in RBAC.

D. Authorization Mechanism

Authorization is the process of determining access rights to resources in an application. In other words, determining "who has access to what". Authorization is used to answer security questions like, "is the user allowed to edit accounts", "is this user allowed to view this web page", "does this user have access to this button?" These are all decisions determining what a user has access to and therefore all represent authorization checks.

As we showed in Figure3,checking an access request has two steps: first we must compute the principals for which the requester is authorized, second we must determine whether the principals are authorized for the requested action. Finally we decide to allow or deny the request based on those authorizations.

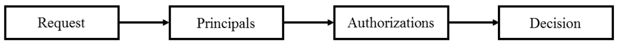

Fig. 3. Authorization Process

It should be pointed out that in step two we need to define the principal matching policy [14], [15] which used to determine whether the request is allowed. As we classify the resources into different categories in our model, the different request that the requested resource is in different calsses may need to be evaluated by different principal matching policy. According to this, we proposed two principal matching policies: All-match policy and First-match policy.

1) *All-match policy*: let P be a set of enabled principals, let AP be the principal which the requester authorized, and the request is denied where if there exists one principal in P that could not be matched from AP.
2) *First-match policy*: let P be a set of enabled principals, let AP be the principal which the requester authorized, and the request is allowed where if one-of (P) matched from AP.

In UNIX access control model, actions can be done on resources are represented by bits. We assume the existence of a set of generic actions $A = \{a,d,u,v\}$ and $B \subseteq A$, where $b_a b_d b_u b_v \in \{0,1\}^4$ is the characteristic set of B. For example, the set $\{a,u\}$ is represented by bits like 1010. This means each group of four bits corresponds to the permissions associated with each principal for that resource. In this way, the cost of manage the authorization state is reduced, it can also improve the speed of principal matching process. So we adopted this mechanism in our model.

IV. Case Study

In this section, we introduce the background of our study case. Then we imply the access control in this case with our design, including the basic structure and data model. We evaluate our implementation and compare it to the majority implementation of role-based access control in applications at last.

A. Background

The Particular case study that we present in this paper was carried out with a major community service company in China with 1600 employees. The company's main business is to provide comprehensive community services for multiple communities. In order to make community management convenient and provide a comfortable environment for residents, the service company uses a smart community system to support its business.

There are three applications in this smart community system, one web application and one mobile application for employees of the community service company, the other is a mobile application for residents lived in the 14 communities. Residents can get community's information and submit service order through the mobile application, and the employees can publish community notices and manage residents' orders through the application. All of these applications are supported by the powerful smart community system.

Security design of the smart community system is necessary. As both residents and employees access to the system, access control is difficult in it. We conclude three security requirements below:

- Users can have access only to authorized resources of authorized communities.
- A user can have multiple roles and a role can be assigned to multiple users. The associations between users and roles are required to be dynamic updating.
- A role can have multiple resources and a resource can be assigned to multiple roles. The associations between roles and resources are required to be dynamic updating.

According to the requirements concluded above, we found that we can imply this system with our design completely. So we imply it in the following section.

B. Implementation

We analyzed the smart community system, and decided to divide the access control into two parts. One of them is in charge of assigning roles to users and defining the permission between resources and roles, we call this part as authorization control. The other is in charge of verifying user identity and the authority they have been granted, we call this part as authentication control. The access control management is independent of the whole smart community system, structure of the access control system is illustrated in Figure 4.

In this structure, the database processes to maintain the basic information such as user information, user authority, role information, resource authority as well as provides an operation connection with each of the system's functions. The

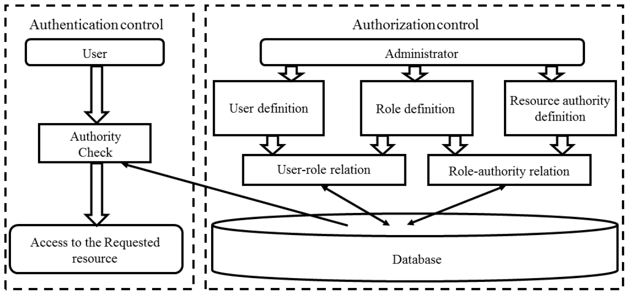

Fig. 4. Structure of the access control system

authentication control works to decide whether the user has access to the requested resources.

we designed the database structure as it shows in Figure 5. In our design, the users may be assigned multiple roles and this makes it necessary to set up lists to record how the users matched the roles. Similarly, the roles may be assigned multiple resource authorities and we also need to set up the lists to record how the roles associated with resource authorities.

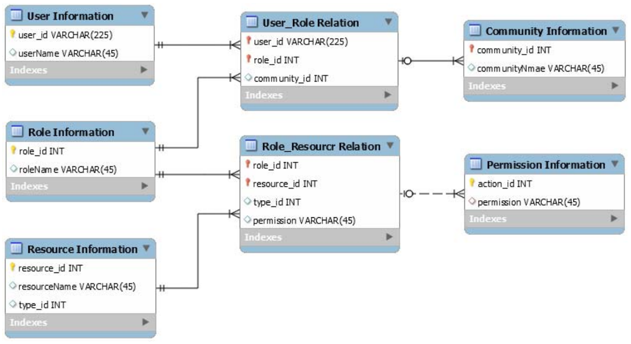

Fig. 5. Database structure of the access control system

In Figure 5, we can imply the permission-resource assignment and the resource-role assignment based on table Role-Resource Relation and imply the role-user assignment based on table User-Role Relation. When we create a role, information of resources and corresponding permissions which assigned to the role will be stored in table Role-Resource Relation. Similarly, when we assign a role to a user, if resources assigned to the role is related to community, information of role and corresponding communities which assigned to the user will be stored in table User-Role Relation.

We imply our system in java and the development is based on the typical java open source framework named Spring. We use Spring MVC to develop the web application and Hibernate framework is used in data persistence layer.The main function modules of this access control system is user role management module, user management module and administrator management module. We code 5 APIs for role management module, 7 APIs for user management module, 7 APIS for administrator management module and 4 APIs for authorization progress.

C. Evaluation

We applied the access control for the smart community system. So far the system provides support for 14500 active residents in 14 communities. The number of community related resources is 12, the number of system resources is 7, the number of private resources is 4. As we can see from Table I, the number of role need to be created in spring security is $183 = 12 \times 14 + 7 + 4 \times 2$, the number of role need to be created in our model is $23 = 12 + 7 + 4$. Obviously our model is better than traditional role-based access control, and the maintenance costs less in our design.

TABLE I

Model	C	R1	R2	R3	Roles
Traditional Model	14	12	7	4	183
Our Model	14	12	7	4	23

We compared our implementation with spring security which was a popular security management open source framework implied with RBAC framework in java. We found that our design is simple and has a major impact on development and production deployment practices. We conclude the advantages of our approach as below:

- Reduced Code Refactoring: Spring security basing code on what the application can do, we're basing our security on things that are core to the application itself and which change much less frequently, that is the resources that the application interacts with. Using this approach, software developers can modify security checks as they work on the application's functionality, not the other way around as traditional RBAC so often requires.
- Resources and Actions are intuitive: Representing what is being protected and how it might be acted upon is a more natural way of thinking about the problem. The Object Oriented programming paradigm and REST communication models fundamentally reflect this viewpoint and are very successful because of it.
- Externalized Security Policy Management: Because the source code only reflects resources and behaviors and not combinations of users, groups, and roles, that kind of association management can be externalized to a different part of code or to a dedicated tool or administrative console. This means developers don't need to spend their time making security policy changes, and instead a business analyst or even end-user can make security policy changes as necessary.
- Make Changes at Runtime: Because security code checks do not depend on knowing how behaviors are associated (i.e. how groups, roles and users might be related), we can change those associations in an application's security policy while the application is running. There is no need

to refactor code to enforce a new security policy change, as is always the case under traditional RBAC.

V. Conclusion and Future Work

We proposed an advanced Role-based access control model for categorized resource in smart community system. The improvement is that we assigned resource authority to role and assigned role to user, unlike the coarse-grained way that only assigned user an implicit role, our approach is more fine-grained. As the resources in different classes are authorized in different policy in smart community system, we classified the resources in our model inspired by the design of Task-Role access control model. We also referred the UNIX access control model to imply our presentation of authorization state. We believe our model provides significant advantages over existing models for role-based access control. Our model is generic, it can be applied in various system.

There are still many challenges for future work. With the developing of cloud computer and big data, people store, search and share resources on the cloud, the security of cloud is research hotspots in cloud computing. So research on access control in cloud is necessary. In recent years, researchers have proposed some access control models for cloud environment, but there still exists the improvement space. Similarly, resources on cloud are in different classes, and our model considered the classification of resources, it may be possible to apply our model to the cloud environment. So our future work is to research how to improve our role-based access control model for categorized resource to provide support for cloud access control.

Acknowledgments

This work was supported by the China National Basic Research Program (973 Program, No.2015CB352400), NSFC under grant U1401258, Science and Technology Planning Project of Guangdong Province (2015B010129011), the Research Program of Shenzhen (No.JSGG20150512145714248), and the CAS/SAFEA International Partnership Program for Creative Research Teams.

References

[1] J. Crampton, "Why we should take a second look at access control in unix," 2012.

[2] C. Yuan, J. Park, and R. Sandhu, "Relationship-based access control for online social networks: Beyond user-to-user relationships," in *Privacy, Security, Risk and Trust*, 2012, pp. 646–655.

[3] Y. Cheng, J. Park, and R. Sandhu, "A user-to-user relationship-based access control model for online social networks," *IEEE Transactions on Dependable & Secure Computing*, vol. 7371, pp. 1–1, 2015.

[4] S. Z. R. Rizvi, P. W. L. Fong, J. Crampton, and J. Sellwood, "Relationship-based access control for an open-source medical records system," in *The ACM Symposium*, 2015, pp. 113–124.

[5] P. W. L. Fong, "Relationship-based access control: protection model and policy language," in *ACM Conference on Data & Application Security & Privacy*, 2011, pp. 191–202.

[6] S. Oh and S. Park, "Task-role based access control (t-rbac): An improved access control model for enterprise environment," in *Database and Expert Systems Applications, International Conference, DEXA 2000, London, Uk, September 4-8, 2000, Proceedings*, 2000, pp. 264–273.

[7] M. Hammoutene, M. Petkovic, and C. V. Conrado, "Role-based access control," 2013.

[8] C. Bellettini, E. Bertino, and E. Ferrari, "Role based access control models," *Information Security Technical Report*, vol. 6, no. 2, pp. 21–29, 2001.

[9] N. Li, J. C. Mitchell, and W. H. Winsborough, "Design of a role-based trust-management framework," vol. 1, pp. 114–130, 2002.

[10] G. Mei and S. L. Osborn, "A design for parameterized roles." in *Research Directions in Data and Applications Security Xviii, Ifip Tc11/wg 11.3 Eighteenth Conference on Data and Applications Security, July 25-28, 2004, Sitges, Catalonia, Spain*, 2004, pp. 251–264.

[11] L. Giuri and P. Iglio, "Role templates for content-based access control." in *ACM Workshop on Role-Based Access Control*, 1997, pp. 153–159.

[12] T. Huang and T. Jia, "A role-activity based access control model for workflow system," *Network Security Technology & Application*, vol. 33, no. 10, pp. 2295–2300, 2011.

[13] S. Oh and S. Park, *An Integration Model of Role-Based Access Control and Activity-Based Access Control Using Task*. Springer US, 2002.

[14] J. Crampton and J. Sellwood, "Path conditions and principal matching: A new approach to access control," *Computer Science*, pp. 187–198, 2014.

[15] P. Mankiewicz, "On context in authorization policy," in *Proceedings of the eighth ACM symposium on Access control models and technologies*, 2003, pp. 80–89.