**Dongliang JIAO[1,2], Lianzhong LIU[1*], Shilong MA[2]**

Key Laboratory of Beijing Network Technology, Beihang University (1)
State Key Laboratory of Software Development Environment, Beihang University (2)

# Research on a Fine-Grained Overriding Mechanism Based on Delegation

***Abstract**. Discretionary Overriding of Access Control is a flexible solution that gives the subject of the access control policy the ability to override the denied access. However, the definition of emergency situations is difficult to express in the mechanism, which may render it inefficient in such situations. In the present work, a fine-grained overriding mechanism based on delegation in presented. In the proposed mechanism, the permissions of the Overriding Ability Subject are delegated from the Overriding Permission Subject, so that users with high-level roles can determine whether or not there exists an emergency situation and whether or not to allow overriding.*

***Streszczenie.** System DOAC umożliwia dostęp zewnętrzny w sytuacjach wyjątkowych. W pracy zaproponowano subtelny system udzielania zgody na dostęp oraz możliwość uznania przez upoważnionego użytkownika, że sytuacja jest wyjątkowa. (**Badania ograniczenia dostępu typu overriding bazującego na delegacji upoważnienia**)*

**Keywords:** Overriding, Delegation, Subject Logic, Belief, Access Control.
**Słowa kluczowe: kontrola dostępu, overriding.**

## 1. Introduction

Role-Based Access Control (RBAC) is a well known and widely used model for expressing access control policies [1-2]. However, the model does not consider some special cases in actual application, thereby necessitating its expansion. Delegation is one of the most important expansions required to solve issues with role backup, collaboration, and decentralization of rights [3-5]. Many studies have been conducted on the delegation of authority. An easily overlooked problem by researchers is that all the approaches published thus far require that some authorisations are created in advance, that is, either an explicit permission or a delegation right that allows the permission to be created when it is needed. None of the approaches reported address the issue that it is not always possible to plan ahead, and that the authority empowered to create an authorisation may not be available at the time access is needed, rendering authorisation missing [6].

Rissanen et al. [6] suggested a flexible solution called Discretionary Overriding of Access Control (DOAC). This solution gives the subject of the access control policy the ability to override the denied access, and it requires that the overriding process be audited and that a notification be sent to a managing authority. DOAC differs from other solutions in that it has the notion of Authority Resolution, which is an automatic procedure that, given information about an override and an access control policy, determines who is in a position to audit and approve the override in a retroactive manner. The discretionary override mechanism increases the flexibility of the access control model to handle hard-to-specify or unanticipated situations.

However, the defects of DOAC mechanism are obvious. Rissanen et al. [6] only discussed the necessity of establishing the mechanism and the feasibility to achieve this Mechanism; they did not discuss the leak of privilege that may result from this approach. At least three problems must be solved: First, the subjects set who can perform overriding, that is, which type of role or user can perform overriding in an emergency situation, must be determined. Second, which permissions set can be overridden, that is, which permissions can be executed in an emergency, must be identified. The sensitivity of different permissions usually varies such that not all permissions may be overridden. Third, the occurrence of overriding in non-emergency situations must be prevented and that overriding takes place only in case of emergency, although an emergency situation cannot be precisely defined, must be ensured.

To solve these problems, an Overriding Model based on Delegation according to the FCDAM is proposed [7]. In this mechanism, the permissions of OAS (Overriding Ability Subject) are delegated from OPS (Overriding Permission Subject), so whether or not an emergency situation exists and whether or not to allow overriding can be determined by thigh-level Roles, which have a higher trust value. It is a fundamental solution to the defect of the overriding mechanism that cannot define emergency situations. It is possible to implement fine-grained overriding by introducing a trust mechanism, which can compensate greatly for the shortcomings of existing research.

## 2. Releted Work

The idea of being able to override denied access is not new, but not many scholars have paid attention to it so far. Here we list some of the typical work in this area.

Li N et al. [8] introduced the notion of resiliency policies in the context of access control systems. Such policies require an access control system to be resilient to the absence of users. An example resiliency policy requires that upon removal of any *s* users, there should still exist *d* disjoint sets of users such that the users in each set together possess certain permissions of interest. Such a policy ensures that even when emergency situations cause some users to be absent, there still exist independent teams of users that have the permissions necessary for carrying out critical tasks.

Povey [9] recognized the problem of legitimate demand for access in unanticipated situations. His main focus was on guaranteeing system integrity by means of transactions that can be rolled back.

Rissanen et al. [10] extended a particular access control framework, the Privilege Calculus, which featured the possibility of overriding denied access for increased flexibility in hard-to-define or unanticipated situations.

Rissanen et al. [6] suggested DOAC. This approach differed from other solutions in that it has the notion of Authority Resolution, which is an automatic procedure that, given information about an override and an access control policy, determines who is in a position to audit and approves the override in a retroactive manner.

Alqatawna et al. [11] introduced a discretionary overriding mechanism in XACML. The approach featured XACML obligations and allowed the definition of a general obligation-combining mechanism.

## 3. Fine-Grained Overriding Mechanism based on Delegation

### 3.1 Preliminary Knowledge

For the convenience of establishing a delegation-overriding mechanism in the next section, the basic structure, two basic functions, and authority constraint relations of the RBAC model are given from definitions 1 to 3.

**Definition 1 (Core RBAC Model).** Core RBAC recognizes five administrative elements: (1) users, (2) roles, and (3) permissions, where permissions are composed of (4) operations applied to (5) objects.

- *UA* ⊆ *USERS* × *ROLES,* a many-to-many mapping between users and roles (user-to-role assignment relation).
- *PA* ⊆ *PRMS* × *ROLES*, a many-to-many mapping between permissions and roles (role-permission assignment relation).
- *SUBJECTS*, the set of subjects.
- *subject_user*(*s*: *SUBJECTS*) → *USERS*, the mapping of subject s onto the subject's associated user.
- *subject_roles*(*s*:*SUBJECTS*) → $2^{ROLES}$, the mapping of subject *s* onto a set of roles. Formally: *subject_roles*($s_i$) ⊆ {*r* ∈ *ROLES*|(*subject_user*($s_i$,*r*) ∈ *UA*}

**Definition 2 (Return Role of User Function, Return Permission of User Function).**

*Assigned_roles*(*u:USERS*)→$2^{ROLES}$, the mapping of role *r* onto a set of users. Formally: *Assigned_roles*(*u*)= {*r*∈ *ROLES* | (*u*,*r*) ∈*URA*};

*Assigned_Permissions*(*r:ROLES*)→$2^{PERMS}$, the mapping of role *r* onto a set of permissions. Formally: *Assigned_Permission*s(*r*)= {*P*∈*PERMS* | (*p*,*r*) ∈*PRA*}.

**Definition 3 (Authorization Restraints).**

There are two basic kinds of authorization restraint in the RBAC model: SoD (Static SoD, Dynamic SoD) and the least privilege principle. Their specific definitions can be found in Ref. [1].

### 3.2 Overriding Model-based Delegation

Traditional mechanisms of access control only care about the authority obtained from roles of a user; they do not discuss the situation that if the subject is not available, who will execute the permissions? Delegation authority mechanisms and the idea of discretionary overriding of access control provide a very effective solution for this problem. Delegation authority has seen some mature implementations, however, discretionary overriding does not include a discussion of the scope of subjects and permissions that execute overriding. Thus, the mechanism has two shortcomings: (1) Urgent situations still cannot be defined. Overriding operations must be licensed by third-party authorities, but these third-party authorities are often unavailable in the automated management system. (2) In this mechanism, the subjects that can execute overriding and permissions that may be overridden are unlimited. This scenario could result in permissions leaking.

For the problems above, the concepts of OAS and overriding execution subject are introduced. The delegated authority mechanism and the subject that has overriding ability are introduced. The subject with overriding ability can delegate overriding permissions to the subject that can execute overriding permissions, so as to achieve the effect of "Separation of Duty" in some sense. The mechanism also solves the problem where discretionary overriding mechanisms cannot define urgent situations. In order to facilitate the discussion, we first give the formal definition of overriding.

**Definition 4 (Permission Overriding).** Let *L* be a permissions system, $\exists u, u' \in USERS$, and *assigned_permissions(assigned_roles(u)) ≠ assigned_permissions* (*assigned_roles*( $u'$ )), if user *u* can override some permissions, $P_1,P_2,\ldots P_i$ of user $u'$ is expressed as:

$u' \hookleftarrow u$ ($P_1,P_2,\ldots P_i$), {$P_1,P_2,\ldots P_i$} ⊆ *assigned_permissions* (*assigned_roles*( $u'$ )),and{$P_1,P_2,\ldots P_i$} ⊄ *ssigned_permissions* (*assigned_roles*(*u*));

where symbol ↵ indicates overriding. $P_1,P_2,\ldots P_i$ is the permission of user $u'$ but not user $u$ .

**Definition 5 (Overriding Ability Subject and Overriding Permission Subject)**. OAS is the subject who can execute the permissions overridden, while OPS is the subject who has the right to override the permissions of another subject. In addition, we set the rule that permissions of overriding permission subject can be executed by delegating to the overriding ability subject.

The separation of rights and ability of overriding requires that the occurrence of an Override must be involved in at least two subjects. First of all, the overriding permission subject delegates its overriding permissions to the overriding ability subject, and then the overriding ability subject executes it. Thus, the difficulty of defining urgent situations is ingeniously solved, and the risk of permission leaks is reduced.

For the convenience of description, we let $S_d$ to indicate OPS, let $S_e$ indicates OAS, and indicates the subjects whose permission is overridden. To prevent the phenomenon of self-authorization in this mechanism, we define the following constraints:

**Constraint 1**: $S_d \neq S_e$, that is, a subject cannot have both overriding permission and overriding execution capability.

Based on the analysis above and constraint 1, we redefine the definition of overriding model based delegation, that is, if we let $\leftarrow$ indicates delegation authorization, then overriding model based delegation would be presented as :

$S_o \hookleftarrow (S_d \leftarrow S_e)(p_1, p_2 \ldots p_i)$,

which {$P_1,P_2,\ldots P_i$} ⊆ assigned_permissions($S_o$).

Through analysis of realistic scenarios, we think that the OAS used to include the subjects that have permissions similar to the subject being overridden, but the subjects who have higher or lower levels have little possibility (for example, when the manager is unavailable, a vice manager or manager assistant can sign, but not the Chairman of the Board or an employee). However, subjects with higher levels have higher trust values, for instance, most people prefer the director of surgery rather than a houseman or practice nurse to treat patients. From the analysis above, in order to improve the OMBD mechanism, the concept of Role Level is introduced.

**Definition 6 (Role Level).** $RH \subseteq R \times R$ , defines the partial orders on the roles, which indicates the relationships of hierarchy of roles. Denoted as ⩾ , $r_1 \geqslant r_2$ indicates that $r_1$ has all the permissions of $r_2$. $RH$ has the features of reflexivity and transitivity.

**Definition 7 (Same Order User, Higher Level Role; Lower Level Role).** Let $u, u' \in Users$, $r_e, r', r'', \in Roles$, if *assigned_roles*(*u*)= *assigned_roles*(*u'*) = $r_e$, then the relationship of $u$ and $u'$ is considered as the same order, indicated as $u_{equel} : u \approx u'$ ; if $r' \geqslant r''$, then $r'$ is called a

higher level role of $r''$ and $r''$ is a lower level role of $r'$, signed respectively as $R_{hign}$ and $R_{low}$.

**Definition 8 (Role Relationship Function).** *fRcompare*(*r*,*r'*)→{$R_{hign}$,$R_{low}$,△}, which $R_{hign}$ denotes $r$ is a higher level role of $r'$; $R_{low}$ denotes $r$ is a lower level role of $r'$; and △ denotes that there is no hierarchy relationship between $r$ and $r'$.

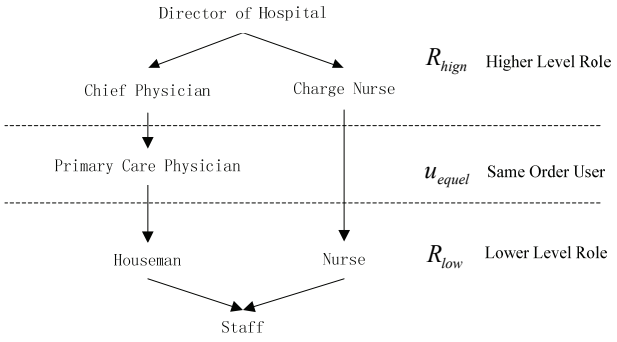


Figure 1 Role Hierarchy diagram in a health care system

Figure 1 is a role hierarchy diagram of a health care system. Which the symbol → denotes the inheritance relationship between roles, the beginning side is child-role, the ending side is parent-role, the user who has the same roles is the same-order user with the one whose permissions is overridden, indicated as $u_{equel}$: $u \approx u'$.

In the Figure 1, each column includes Director of Hospital→Chief Physician→Primary Care Physician→ Houseman→Staff and Director of Hospital→Charge Nurse →Nurse→Staff forms the hierarchical relationships of the role. This relationship is transitive, but there is no relationship between two chains.

### 3.3 Overriding Model based Delegation and Trust Value

Role is a collection of permissions in RBAC model. The sensitivity of a role's permissions is usually different, but the traditional trust management in RBAC ignores the differences between them. This facilitates the authorization management, but is not conducive to the implementation of constraints in the delegation authority of RBAC, increasing the risk of permission leak. Zhai et al. [7] presented a fine-grained controllable delegation authorization model (FCDAM) suitable for open environments that integrates the merits of both RBAC and role-based trust management and can effectively control the propagation of permissions of different sensitivity levels in roles. An approach for assigning trustworthiness thresholds to permissions in local access control policy is discussed, thereby avoiding the defect that RBAC cannot distinguish sensitivity between permissions.

We introduce a trust value and trust threshold similar with FCDAM for the permissions of this system. In FCDAM, however, in order to determine trust threshold of Permission which role obtained through inheritance, with the manner of specifying attenuation coefficient for trust threshold which inheritance from the parent role. Although this avoids specifying trust thresholds for inherited permissions, it makes the trust thresholds of permissions different for different roles, and the higher role's level is the lower permissions' trust threshold. This could cause conflicts in a trust-based authorization management system. Therefore, we believe that the trust threshold is a relative value that indicates the relative sensitivity of a permission to other permissions in the permission set. Thus, the trust threshold should be a constant and have no changes in the role inheritance. The following shows the definition of trust value, trust threshold, and related functions:

**Definition 9 (Trust Value *TV*, Trust Threshold *TT*).** *TV*=*TT*=[0.0, 1.0], indicates the range of trust value or trust threshold, numerically distributed between 0 and 1, where 0 means do not trust at all and 1 means full trust.

**Definition 10 (Return User Trust Value Function, Return Permission Trust Threshold Function)**.

*assigned_UTV*(*u*:*USERS*)→[0.0,1.0], indicates that the trust value assigned to the user *u*. Formally: *assigned_UTV*(*u*)=*TV*.

*assigned_PTT*(*p*:*PERMS*)→[0.0,1.0], indicates that the trust threshold assigned to the permission *p*. Formally: *assigned_PTT*(*p*)=*TT*.

Overriding-based Delegation and trust value is achieved by OPS who have higher level roles and OAS together, so we should consider the trust values of OAS and OPS comprehensively. Therefore, we provide another basic reference additional constraint:

**Constraint 2**: The weighted average trust value of OPS and OAS need to be greater than the trust threshold of permissions to be executed. Formally:

$$\sum_{i=1}^{2} q_i \times assigned\_UTV(u_i) \geq assigned\_PTT(p)$$

, which $\sum_{i=1}^{2} q_i = 1$. Generally, we take $q_1$=$q_2$=0.5, that is arithmetic mean value of OPS and OAS.

Example 1: Table 1 gives an example of a medical information system's role level and permission trust threshold. Assuming that there are seven kinds of roles listed in Table 1 in the medical information system: $R_{dh}$ (Director of Hospital), $R_{cp}$ (Chief Physician), $R_{pcp}$ (Primary Care Physician), $R_p$ (Physician), $R_h$ (Houseman), $R_n$ (Nurse), $R_s$ (Staff); there are five kinds of access modes listed in the table for patient information: $P_{pf}$, $P_{ph}$, $P_{ah}$, $P_{fh}$, $P_{mi}$, the trust threshold of various permissions varies corresponding to information sensitivity. Let $\exists$ *u, u1, u2, u3, u4, u5, u6* ∈ *USERS*, *assigned_roles*($u_1$)=$R_{dh}$; *assigned_roles*(*u*)=$R_{pcp}$; *assigned_roles*($u_2$)=$R_{cp}$; *assigned_roles*($u_3$)=$R_p$; *assigned_roles* ($u_4$)=$R_h$ ; *assigned_roles*($u_5$)=$R_n$; *assigned_roles*($u_6$)=$R_s$.

By default, users who have the role $R_{pcp}$ have all rights that can access the information of patients, formally: *assigned_permissions*($R_{pcp}$) = {$P_{pf}$, $P_{ph}$, $P_{ah}$, $P_{fh}$, $P_{mi}$}, and the doctors of other roles have no rights to access patients' information. We believe that trust value and trust threshold are used to indicate a permission's sensitivity level relative to all permissions in the system, so it can be assigned in advance.

It should be noted that the default value of the role listed in Table 1 does not mean that the role has trust value itself. It is used to initialize the user who is assigned to this role; the user inherits the default value. Trust value is constant here, but it will change based on the user's behavior of overriding. Specific evaluation mechanisms will be discussed in detail in future papers.

Now let us suppose an emergency situation. For example, patient *A*, who is assigned to primary care physician *u*, has a condition that suddenly deteriorates and physician *u*'s treatment is needed urgently. At this time, however, *u* is unavailable, and he or she did not delegate his or her permissions to other people. Let *u',u1, u2, u3, u4, u5, u6* is available now, which *u'*: *assigned_roles*(*u'*) = $R_{pcp}$, then need them to execute overriding function based on delegation. Depending on the different circumstances, there are two scenarios for discussion:

Table 1 Example of role level and trust threshold in a medical information system

| Role | Role Class | Default Trust Value | Permissions | Threshold Value |
|---|---|---|---|---|
| Director of Hospital | $R_{high}$ | 0.85 | Patient File | 0.75 |
| Chief Physician | $R_{high}$ | 0.75 | Patient History | 0.60 |
| Primary Care Physician | / | 0.70 | Allergic History | 0.50 |
| Physician | $R_{low}$ | 0.65 | Family History | 0.65 |
| Houseman | $R_{low}$ | 0.60 | Medical Insurance | 0.85 |
| Nurse | $R_{low}$ | 0.50 | | |
| Staff | $R_{low}$ | 0.30 | | |

Scenario 1: The first user may executive overriding based on delegation is $u_2$ and $u'$, the reason is $fRcompare(assigned\_roles(u_2),R_{pcp})=R_{hign}$, $u \approx u'$ . This is consistent with the actual situation, because the first user to be notified and informed is often the nearest user. Because of $assigned\_UTV(u_2)$= 0.75, $assigned\_UTV(u')$= 0.70, $u'$ obtains the trust value through delegation $delegated\_UTV(u')$ =0.725, then $u'$ can visit all the information of the patient besides of Patient File and Medical Insurance.

Scenario 2: As in Scenario 1 Trust Value user $u'$ obtains is 0.725. Assuming the situation where a user needs to visit the patient's patient file, but the trust value needs to be greater than 0.75 to access the file. According to constraint 2, $u_2$ and $u'$ are not qualified to execute overriding, and the operation must be performed by $u_1$ and $u'$ or $u_1$ and $u_2$.

In the initial state, the trust value of user is inherited from roles, but the value will be changed according to the evaluation from a subject whose permissions are overridden. The change will become evidence for the next overriding based on delegation, and the subject who has low trust value will not be eligible to execute overriding. Thus, the security of the system is strengthened, preventing the occurrence of overriding with low reliability.

## 4. Author Artwork Risk Analysis of OMBD

The main risk of Overriding Model-based Delegation is to have to allow the overriding authorized rules that are not strict to meet the needs of authorization in case of the emergency that cannot be defined. It should be noted that not all overriding is detrimental. Usability is the goal to design. The goal of usability requires that this mechanism is necessary in a permission system. The need to guard against is how to avoid misjudgment necessity of overriding and the possibility of overriding mechanism abuse because that punishment mechanisms to the subject who leading rights leak are inadequate. For example, it is necessary that u3 can override some permissions of u in an access control system based on automatic control in Example 1. However, if the subjective evaluation system is imperfect, there are obvious errors after several executions of overriding by u3, the user may still have enough trust value to execute next overriding, which may lead to leakage of rights.

There are two main reasons for the overriding mechanism proposed in this paper: First, OPS and OAS may misjudge the overriding need and execute overriding when it should not have happened thereby leading to a leak of rights. Second, the subjective evaluation system is imperfect, such that malicious users can exercise privileges that are illegal. For different reasons causing the abuse, we should take different preventive strategies. For the former, when Overriding is triggered, the system should notice the subject whose permission is overridden in various ways [6], and it is necessary that the subject who evaluates whether or not overriding is reasonable and timely, thereby preventing the recurrence of illegal override. For the latter, system administrators should carefully check whether or not the mechanism of subjective evaluation is reasonable, and should increase the necessary restrictions for the system.

## 5. Conclusion

It is both preventability of privileges abuse and availability of permissions that to be the demand and difficulty of design and implementation in the overriding mechanism. The current study is still dependent on the subject's own moral constraints, so it cannot guide the actual development of the system with the overriding mechanism and the authorized operation of the administrator. It lacks prevention mechanisms for permission leaks. Therefore, we present a fine-grained overriding mechanism based on delegation. The permissions of OAS are delegated from OPS, so whether or not in an emergency situation and whether or not to allow overriding can be determined by high-level Roles that have a higher trust value. This mechanism is a fundamental solution to the defect that most overriding mechanisms cannot define emergency situations, and it is possible to implement fine-grained overriding by introducing of a trust mechanism, which compensates greatly for the shortcomings of existing research.

## REFERENCES


[1] Sandhu R S., Lattice-based access control models. *Computer*. 26 (1993) No.11, 9-19.

[2] Sandhu R S, Coyne E J, Feinstein H L, et al., Role-based access control method, *IEEE Computer*. 29 (1996) No.2, 38-47.

[3] Zhang X, Oh S, Sandhu R. PBDM: a flexible delegation model in RBAC, *In: Proceedings of the eighth ACM symposium on Access control models and technologies*. ACM, 2003. 149-157.

[4] Na S Y, Cheon S H., Role delegation in role-based access control, *Proceedings of the fifth ACM workshop on Role-based access control*, 2000, 39-44.

[5] Crampton J, Khambhammettu H., Delegation in role-based access control, *International Journal of Information Security*. 7(2008) No. 2, 123-136.

[6] Rissanen E, Firozabadi B, Sergot M., Towards a mechanism for discretionary overriding of access control, *Security Protocols*, 2006, 312-319.

[7] Zhai Zheng-De, Feng Deng-Guo, Xu Zhen., Fine-Grained Controllable Delegation Authorization Model Based on Trustworthiness, *Journal of Software*, 18(2007) No.8, 2002-2015.

[8] Li N, Wang Q, Tripunitara M., Resiliency Policies in Access Control, *ACM TRANSACTIONS ON INFORMATION AND SYSTEM SECURITY,* 12(2009) No.4, 20.

[9] Povey D., Optimistic security: a new access control paradigm, *Proceedings of the 1999 workshop on New security paradigms,* 2000, 40 – 45.

[10] Rissanen E, Firozabadi B, Sergot M., Discretionary overriding of access control in the privilege calculus, *Formal Aspects in Security and Trus*t, 173(2005), 219-232.

[11] Alqatawna J, Rissanen E, Sadighi B., Overriding of access control in XACML, *Eighth IEEE International Workshop on Policies for Distributed Systems and Networks Proceedings*. 2007, 87-95.



***Authors**: Dr.Jiao Dongliang, prof. Liu Lianzhong and prof. MA Shilong, E-mail: slma@nlsde.buaa.edu. School of Computer Science and Engineering, Beihang University, Beijing 100191, China. Corresponding author: Liu Lianzhong, E-mail: lianzhong-liu@163.com*