Research on Improved Access Control Model Based on T-RBAC

Yong Wang¹, Yueping Yang¹, Binxu Wang, Qingwen Ran² and Xiaoming Ju*²

¹ Ningbo Power Supply Company Information Communication Branch, State Grid Corporation of China, Zhejiang, 315000, China
² School Computer Science and Software Engineering, East China Normal University, Shanghai, 200062, China
*Corresponding author’s e-mail: xmju@sei.ecnu.edu.cn

Abstract. The T-RBAC model has a wide range of applications in various fields, but has certain disadvantages. Aiming at the shortcomings of T-RBAC model in task assignment and authority management, an improved model is proposed. The new model simplifies the task classification of T-RBAC model. Considering the task context, task time limit, and task status attributes, the role manager, task manager, and authority manager are introduced, and the grant of authority is closely linked with the three managers, which enhances the dynamic management of rights. The constraints are strengthened and expanded to ensure dynamic separation of duties during task execution. The model has finer-grained rights management, better meets the principle of separation of duties and least privilege, and has higher security.

1. Introduction
Access control is a way to allow or restrict the access rights and scope of the subject to the object through some means. By controlling access to key resources, it can prevent the intrusion of illegal users or the damage caused by the misoperation of legitimate users. Common models currently have role-based access control, task-based access control, and task- and role-based access control.

In 1996, Ravi [1] proposed the Role-based access control (RBAC), which introduces the concept of roles, using roles as a bridge between rights and users, assigning permissions to roles first, and then assigning roles to users, thereby completing the assignment User rights, this way separates users from permissions, enhances security, and is more flexible and simplified. However, the RBAC model does not strictly distinguish between authorization and decentralization, and there is an illusion of least authority, so it is impossible to dynamically manage rights. In 1996, Sandhu [2] proposed the Task-based access control (TBAC), the model introduces the concept of tasks, task-centric, dynamic authorization, and enhanced security, but there is still no clear separation between tasks and roles in TBAC neutralization. Problems such as role hierarchy levels are not supported, so they are not widely used.

While the literature [3] combines the advantages of the above two models, In 2003, Park proposed a task- and role-based access control model T-RBAC. T-RBAC changes the three-layer access control model to four layers, adding between roles and permissions. The task is to assign the access rights to the task, and then assign the task to the role. The task is the bridge between the role and the permission, and the two are connected. This simplifies the assignment and management of tasks while preserving the advantages of dynamic management of permissions in TBAC. But the model has some improvements in fine-grained permissions control. Literature [3] introduces context to control the execution
environment of the task, but does not consider the task's own state. The literature [4] improves the security and management dynamics by classifying roles and permissions from the perspective of control. In [5], from the role refinement, a group and role-based access control model is proposed, which divides the roles into groups, thereby improving the dimension of authority control. However, it does not consider the fine-grained tasks, the context attributes of tasks, the inflexibility of dynamic authorization, the inability to supervise permissions, the allocation of permissions, and permanent ownership.

Aiming at the above problems, this paper proposes an improved model based on the T-RBAC model. The new model simplifies the task classification of the original model, and divides the task into two types: atomic task and separable task. Task context attributes, task life cycle, etc., at the same time to strengthen the management of the task life cycle and fine-grained control of the permissions, proposed role managers, task managers, rights managers, in addition to introduce licenses to ensure minimum authority and authority Separation and other principles of security control, through the issuance of licenses to ensure that our selected users meet the various principles of the model requirements.

2. Related information
The T-RBAC model adds the "task" item to the RBAC model and turns the three-layer model of RBAC into a four-layer model. Unlike RBAC, which does not separate tasks from roles, TBAC does not show roles. T-RBAC puts tasks and roles in equal importance. The two are independent and related. It is a combination of TBAC and RBAC. The RBAC model is shown in Figure 1.

In the T-RBAC model, rights are assigned to roles through tasks. Tasks are privileged. Tasks have different permissions depending on the requirements of task execution and permission constraints, and the physical environment of the task. Permissions vary with the execution of the task, which enables dynamic allocation of permissions and on-demand allocation. A role has permission only when it performs a task, and the rest of the time is in a sleep state. The T-RBAC model is shown in Figure 2.

3. T-RBAC improved model design
(1) Task classification is simplified. The T-RBAC model divides tasks into four categories and expresses various types of tasks in detail, but it is too cumbersome and difficult to implement. Therefore, this paper simplifies task classification and only divides into atomic tasks and separable tasks. For a task that can be completed by one person at a time, the task that can be divided is a task that requires multiple people to complete multiple times, so that the division is more clear than the original model.

(2) Increase the role manager (RM). Dynamic management and monitoring of roles enables the role to be supervised after activation. It records the behavior and authority status of all roles that complete the entire task, expands the role constraints, and ensures separation of duties.

(3) Propose a task manager (TM). Manage and monitor tasks, monitor task status, dynamically adjust and allocate tasks, and make a predictive calculation for the time required to complete the task, ensuring that the time is completed, the task status and task context are integrated. Attributes, task constraints, etc. control and allocate resources required by tasks.

(4) Propose a permissions manager (PM). Distributing and reclaiming permissions prevents the role's permissions from being "activated once and used permanently".

Figure 3 shows the improved model.
3.1. Definition of model components

In addition to the five definitions of the original model, the new model has added some new concepts, which are defined as follows:

Roles Manager (RM) : The role manager is the administrator assigned when the user activates the role. It is used to dynamically manage the role, is responsible for all actions of the role, has the authority to issue licenses, and has the option to complete the next node work in the separable tasks. The user is given the authority to license his or her own node, and the role manager is responsible for the winner of his or her choice.

When the first role of a task is activated, the system configures a role administrator for it, passing the role constraints and model security control principles to the role administrator. The role manager monitors the entire process during the entire task execution. Manage each role, record the behavior and status of each role in the process of completing the task, and be responsible for the role. The life cycle of the role administrator is the same as the life cycle of the task. When the task is executed, the role manager is released. Through the role manager, it is possible to monitor and track the behavior of the role, and to check the constraints of the role.

Tasks Manager (TM) : The task manager is an object of the dynamic management task generated by the system when the user instantiates the task. It mainly checks the constraints of the task and the resources required by the task, and calculates the time required for the task to complete. A dynamic management and supervision of task status.

When the task is instantiated, the system generates a task manager object, passes the task's context attributes, task constraints, and task-required resources to the task manager, and establishes communication between the role manager and the task manager. During the running of the task, the status of the task is recorded, and the task is dynamically mobilized according to the task constraints, the task context, the task state, and the resources required by the task, and the permission needs to be transmitted to the authority manager. Calculate the time required for the task to complete normally, ensure that the task can be completed normally within the event, and handle the abnormally interrupted tasks accordingly.

Permissions Manager (PM) : The rights manager is an object assigned by the system when the user activates the role. The main task is to authorize and authorize the rights recovery when the role performs the task.

When a role is assigned to a user, the rights manager needs to check the user's conditions, role constraints, licenses, etc. to determine whether to assign the task to the user and to reclaim the permissions when the task is completed. The rights manager is responsible for the assignment and collection of rights.

3.2. Model formal description

The improved T-RBAC = \{U, R, T, P, RM, TM, PM\}, U is the user set, R is the role set, T is the task set, P is the permission set, and RM is the role manager set. TM is the task manager set, and PM is the rights manager set.
User-Role Assignment (URA): \( URA \subseteq U \times R \), user role assignment, indicating a many-to-many mapping relationship between the user set and the role set. The role set assigned by user \( u \) is \( UA(u) = \{a \in A \mid (u, a) \in URA\} \).

Role-Task Assignment (RTA): \( RTA \subseteq R \times T \), role task assignment, representing a many-to-many mapping relationship between a character set and a task set, and a task set with multiple roles assigned by role \( r \) \( RA(r) = \{r \in R \mid (r, t) \in RTA\} \).

Task-Permission Assignment (TPA): \( TPA \subseteq T \times P \), task rights assignment, indicating a many-to-many mapping relationship between task set and permission set, and permission set assigned by task \( t \) \( PA(t) = \{p \in P \mid (t, p) \in TPA\} \).

Role Hierarchy (RH): \( RH \subseteq R \times R \), which is a partial order relationship on a character set, also called a role inheritance relationship.

3.3. Model Constraint Rules
This model introduces role managers, task managers, authority managers, etc., and needs to expand the constraints. The constraints are checked and managed by the role manager, task manager, and authority manager. This model constraint mainly includes role constraints and task constraints. The details are as follows.

3.3.1. Role Constraints
A role constraint is a constraint rule that should be followed when a task is assigned to a role, or when a role is assigned to a user, or when the user activates a role. It mainly includes role cardinality constraints, role environment constraints, and role time constraints.

Role cardinality constraints: constraints on the number of roles, constraints on the user corresponding to the role, constraints on the number of tasks corresponding to the role, constraints on the number of active and active roles, etc., a role cannot be assigned simultaneously multiple tasks.

Role environment constraint: refers to the condition check that must be satisfied by the user granted by the role before the authorization of rights and the check of the role activation environment. For example, whether the user has a license, can complete the task on time, and whether the environment required by the role is ready.

Role time constraint: that is, the time limit of the character activity, that is, the role survival time, including the generation time and the recovery time. During this time period, all the activities of the character are valid, and vice versa, the role is inactive. Figure 4 shows role state transition.

3.3.2. Task Constraints
The task constraint, that is, the constraint on the task, can only be executed if the corresponding constraint is met. Task constraints include task cardinality constraints, task context constraints, task state constraints, and separation of duties constraints.

Task cardinality constraints: Constraints on the number of tasks, mainly the constraints on the number of roles assigned to the task.

Task context constraints: The task context mainly refers to the working environment of the task, and the task context constraint refers to the working environment that must be satisfied during the execution.

Figure 4. role state transition diagram
of the task, ensuring that the task is completed by a specific person with a specific resource or method within a specific time period. If the resource required by the task cannot be satisfied, the task hangs. If the resource scheduling succeeds during the execution of the task, the task continues to be executed. Otherwise, the task is terminated and the corresponding authority is recycled.

Separation of duties: The model requires that mutually exclusive rights cannot be assigned to the same entity, such as cashiers and accountants cannot be held by the same person. Since the task in this model is a bridge between roles and permissions, the separation of authority and authority can be achieved well in terms of task nodes. Task node constraints are divided into static constraints and dynamic constraints. Static constraints stipulate that mutually exclusive tasks cannot be assigned to the same role or user. Dynamic constraints stipulate that the same role or user is prohibited from simultaneously activating and running two or more task nodes. Dynamic constraints are checked during the execution of the task node.

3.4. Task Classification and Attributes

The TRBAC model divides the task into four types, which can describe different types of tasks in detail, but it is cumbersome to implement. This model simplifies task classification and only divides tasks into atomic tasks and separable tasks. An atomic task is a task that requires only a single person to perform a single task, and refers to a simple single task. Such as distributing materials, sending mail, etc. The detachable task refers to a task that requires multiple people to perform multiple nodes, that is, a task that requires multiple people to cooperate. We divide the separable task into multiple nodes, each node assigns a role to complete. When the role completes the task of this node, it selects the user who completes the next node and gives the license of its own node. The authority manager checks the task completion of the latter node, and after the verification succeeds, assigns rights to the task and executes the task. The task classification of the new model is simpler than the original model, the operation is more clear, and the user is separated from the authority, which increases the security.

3.5. Task Properties

3.5.1. Atomic task properties

Unlike the TRBAC original model, which has been owned by users since the task permissions were granted, this model introduces task context attributes, task status, and rights managers, who dynamically manage tasks to avoid lifetime ownership. The situation avoids the smallest security risks.

The state of an atomic task includes a sleep state, an active state, a running state, and an ending state. When the task is scheduled, it enters a sleep state, and when it is called, it transitions from the sleep state to the active state. When the task satisfies the task constraint, the role satisfies the role constraint, and the task context attribute is satisfied, that is, the physical environment is satisfied, and the required resources are satisfied and then enter the running state. If the system pauses, the task status returns from the running state to the active state, and the relevant authority enters the freezing. When the system performs active recovery, it will enter the running state again, and the related permissions will be restored. Sleep state, activation state, and running state. If the three states fail to complete the task within the task time limit, they will jump from the current state to the end state. Secondly, when the task execution is completed, it will transition from the running state to the ending state. The atomic task state transition is shown in Figure 5.
3.5.2. Separable task attributes
In the same way as the atomic task, in order to achieve dynamic management of permissions and fine-grained control, sub-tasks can also have corresponding state transitions. The separable task states include a sleep state, an activation state, a running state, a suspended state, a failed state, and a successful state. The specific conversion diagram is shown in Figure 6.

Sleep State: When the task is scheduled to sleep.
Active State: When the task is called, there is a sleep state to enter the active state.
Running Status: The task in the active state meets the constraint condition and the context attribute. The task is activated and the execution is successful. When the status is entered, the permission administrator grants the relevant permission and the task time limit starts counting.
Suspended State: When the context is not satisfied during the execution of the task, or the system is forced to suspend, it enters the suspended state, and the related permissions are frozen. However, the task time limit continues to count down. If the task context is satisfied, or the system actively recovers, it returns to the running state again, and the related permissions are restored.
Failure Status: Enter this status when the task is not completed within the task time limit.
Success Status: Enter this status when task execution is complete.

4. Effect analysis
Through the above description of the improved T-RABC model, it can be known that the new model introduces the role manager, the task manager, the authority manager, and also simplifies the task classification. The task context is considered in consideration of the task context, and the task time limit and so on.

The simplified task classification makes the structure of the new model simpler than the original model, the classification is clearer, the implementation is easier, and the operation is simpler. The introduction of three managers makes it possible to have a more secure guarantee when assigning rights. At the same time, the dynamic granting and recycling of permissions is realized, and the task time limit is avoided to avoid the task locking state, which provides a more powerful new model. Rights management capabilities, in addition, the task execution requirements meet the task context attributes,
which provides a better environment and security for the execution of the task. The new model better satisfies the principle of separation of duties and least privilege of access control.

The concept of role manager, task manager, and authority manager is proposed, the security mechanism is added, and the authority can be dynamically adjusted. The execution of the task can be traced to the root cause. If the task execution is wrong, it can find out which step is wrong. Develop solutions for specific situations. Secondly, the introduction of the concept of the license reduces the task of the role manager. In addition, the task performer is more familiar with the task, knows which type of role and user the task needs, and can find a person more suitable for the next task node, and all managers it applies for check and check constraints. This separation of powers and positions has better security and also guarantees the principle of separation of duties.

After introducing three managers, more strict constraint rules were proposed, and the constraint rules were classified and expanded, and a more strict and complete definition was given. A strong constraint check was implemented. Supported the principle of separation of duties, and realized the dynamic adjustment of permissions, dynamic authorization and recycling of permissions.

5. Conclusion
Based on the TRBAC model, this paper proposes an improved model. The new model retains the role hierarchy and task-driven, role assignment characteristics of the original model. It simplifies the task classification, introduces the concepts of role manager, task manager, authority manager, license, etc., and considers the task context, task life cycle and other aspects, and strengthens the constraints. The new model has more granular permissions management, which can better guarantee the principle of least privilege and separation of duties, and has higher security.

Acknowledgments
This research is supplied by the State Grid Ningbo Power Supply Company Innovation Consulting Project-- Research on Intelligent Detection and Verification Technology for Power Information Network Security Hidden Trouble.

References:
[1] Ravi S. Sandhu, Edward J. Coyne. Role based access control models[J].IEEE Transaction on Computer, 1996, 29 (2) :38-47.
[2] Dao TH, Gao J, Qi HL, Li L. Research progress of access control model[J]. Computer Science. 2010(11)
[3] Feng J, Wang J. An Improved Model of Access Control Based on T-RBAC[J].Computer Engineering,2012,38(16):138-141.
[4] Cai T, Nie QB, Ouyang K, Zhou JL.RBAC model based on role extension[J]. Computer Application Research. 2016(03)
[5] Huang Y, Li KL. A task-role access control model for cloud computing[J].Application Research of Computers,2013,30(12):3735-3737.
[6] YU CHSH, Nie J. Workflow authority access control model based on group and role[J].Computer Applications,2011,31(03):778-780+783
[7] Chen Sh, Yuan Ch, Lou Sh, Zhang WY. Research and Application of Role Mutual Rejection in RBAC Model[J]. Computer Technology and Development. 2012(12)
[8] H. Zhang, J. Wang and J. Chang, "An access control model for multi-level security in multi-domain networking environments," 2017 9th International Conference on Modelling, Identification and Control (ICMIC), Kunming, 2017, pp. 809-814.
[9] N. Pan, L. Sun, Z. Zhu and L. He, "A temporal approximation-based role mining approach for TRBAC," 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, 2017, pp. 2366-2370.
[10] Zhang B, Zhang Y. Access Control Model Based on Attributes and Roles[J]. Computer Engineering and Design,2012,33(10):3807-3811.