ISSN: 2350-0328

International Journal of Advanced Research in Science, Engineering and Technology

Vol. 4, Issue 10 , October 2017

Developing Software Module Based on TT-RBAC Model of Access Control

Kadirov Mirhusan Mirpulatovich, Tulyaganov Zoxidjon Yakubdjanovich

Assistant professor, Department of Information Technologies, Tashkent State Technical University, Tashkent, Uzbekistan

ABSTRACT: The article is devoted for managing and controlling users' permissions, defining users' rights, also developing new profitable role-based access control (RBAC) model in collaborative systems. The basis for the protection of the processed information from unauthorized access is the implementation of a delimitating policy for access to file objects.

KEYWORDS: TT-RBAC system, object-oriented technology, user, role, access control, role-based access control, collaborative systems, model of access control, attribute, software module.

I. INTRODUCTION

The rapid growth of heterogeneity and scale of modern corporate networks leads to an excessive increase in the vulnerability of not only external but also internal network services.

The larger the scale of the network, the more difficult it is for the administrator to provide reliable network protection, providing adequate response to all possible attempts to break into the computer system. It should be taken into account that threats to information security can be associated not only with unauthorized access to workstations, servers or communication lines. Attacks can be subjected to specialized devices that perform functions within the network routing of the message flow. An attacker can redirect the flow of messages to perform further unauthorized actions on it.

The solution of the task of protecting information from unauthorized access in any information system is based on the implementation of control and delineation of the subjects' access rights to the protected resources, primarily to file objects, since they are intended for storing the processed data. At the same time, users who are identified by accounts are the subjects of access in the demarcation policy. Rules for access of subjects to objects are usually set in the form of an access matrix (the matrix representation is expedient from the point of view of the possibility of transposing a matrix that allows to represent two ways of defining a differentiation policy - subjects to objects or, conversely, to objects of subjects).

II. ENFORCING OF TT-RBAC MODEL IN COLLABORATIVE SYSTEM

TT-RBAC system, which is researched, is developed with object-oriented technology. We define eight classes that represent user, role, permission, object, operation, team, task and session, respectively.

All of them inherit from a superclass named TTRObject that defines the common variables and methods needed by its subclasses. These classes are called TT-RBAC core classes. The major variables and methods defined for these classes and their inheritance relations are shown in Figure 1.

TTRObject is the superclass of all other TT-RBAC core classes. In TTRObject there define three variables; they are id, type and state that are used to hold an entity's identifier, type and state, respectively. Entity type could be user, role and so on[1].

An entity could be in active state or inactive state. Entity state decides if an entity can be used for access control. There are various factors, such as context, that can affect entity state.

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 4, Issue 10 , October 2017

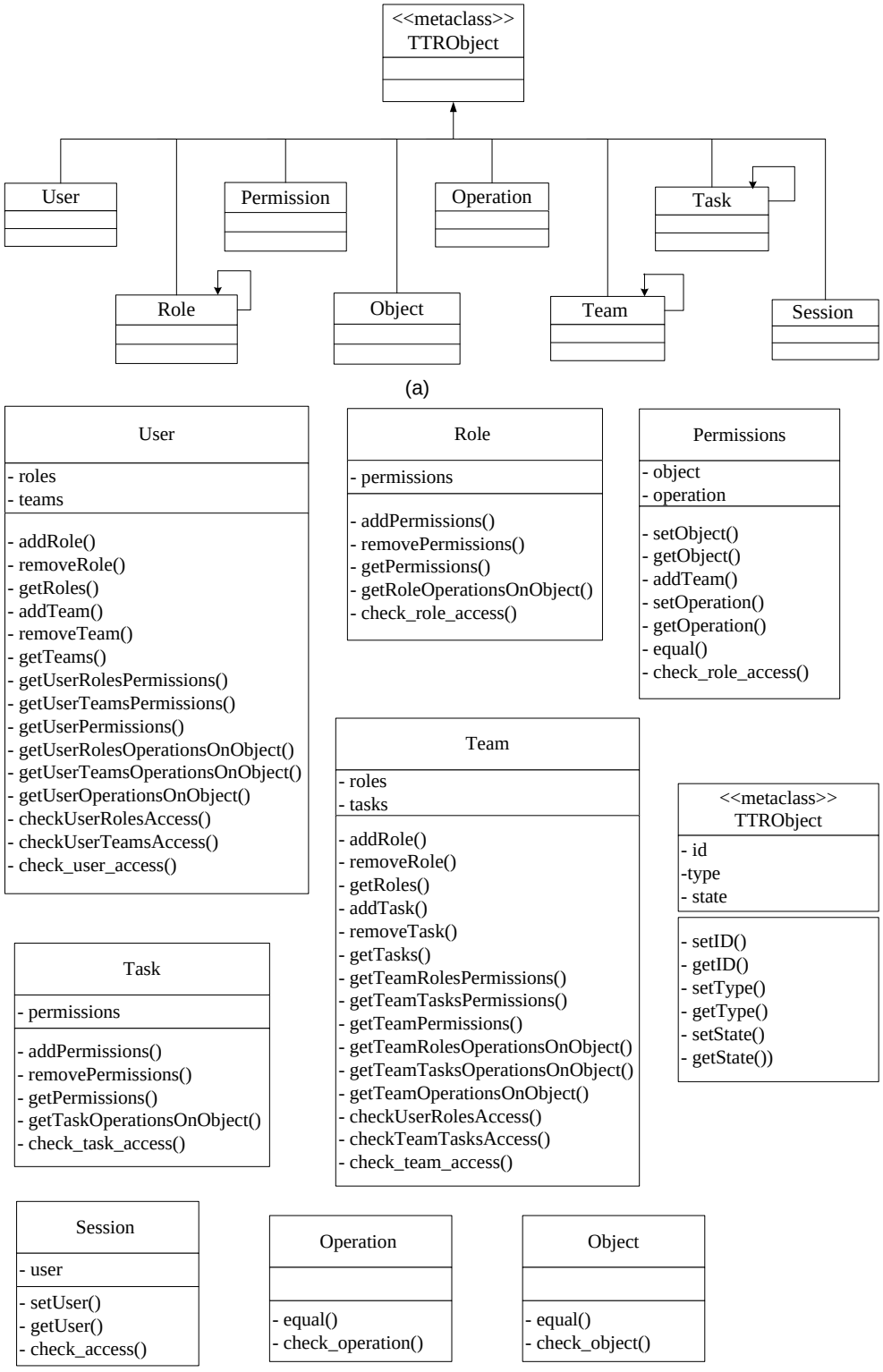

Figure 1: TT-RBAC core classes: (a) inheritances; (b) definitions

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 4, Issue 10 , October 2017

User is the class used to describe users. The variables roles and teams are used to hold Role objects and Team objects, respectively. The methods checkUserRolesAccess, checkUserTeamsAccess and check_user_accessare used for checking user access.

Role is the class used to describe roles. The variable permissions is used to hold permission objects. The method check_role_access is used for checking role access.

Permission is the class used to describe permissions. The variables object and operation are used to hold object and operation objects, respectively. The method equal is used for comparing with an input permission object. The method check_permission is used for checking permission access.

Object is the class used to describe resources. The method equal is used for comparing with an input object name. The method check_object does the similar function to equal, except it also considers object's state.

Operation is the class used to describe operations. The method equal is used for comparing with an input operation name. The method check_operation does the similar function to equal, except it also considers object's state.

Team is the class used to describe teams. The variables roles and tasks are used to hold the role objects and task objects, respectively. The methods checkTeamRolesAccess, checkTeamTasksAccess and check_team_accessare used for checking team access.

Task is the class used to describe tasks. The variable permissions is used to hold the permission objects. The method check_task_access is used for checking task access.

Session is the class used to describe sessions at runtime. The variable user is used to hold a user object. The method check_access is used for checking session access.

From the above introduction, we know these classes are not independent. Their relations are shown in Figure 2.

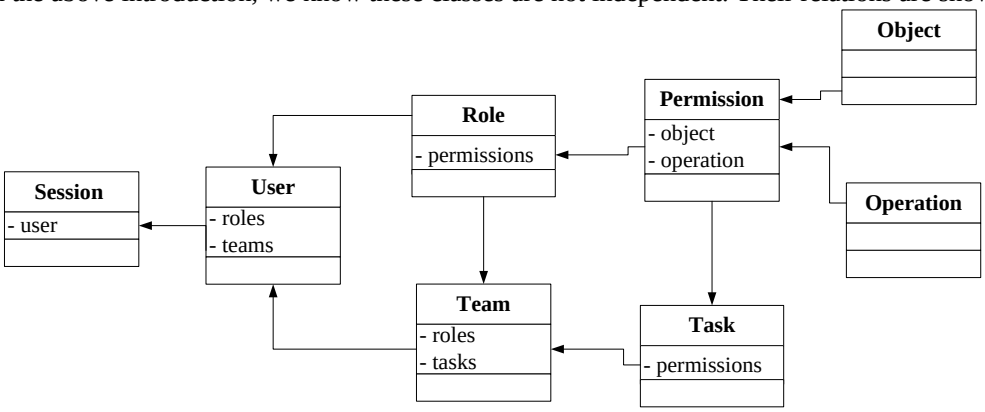

Figure 2: TT-RBAC core class relations

III. CONTEXT CONSTRAINTS OF TT-RBAC

In TT-RBAC, every entity's state can be active or inactive. One entity can be used for access control only when it is in active state. For example, the permissions held by a role are available to a user only when this role is in active state. One entity's state change may affect the permissions that are available to other entities. For example, if an object is deactivated, then all the permissions that relate to this object are deactivated. This effect will be propagated among all the entities that relate to these permissions. One entity's state change may also affect other entities' state. For example, all the employees are assigned to role Staff, and the role Staff is the prerequisite role to all other roles. If the role Staff is deactivated in a session, then all the roles depending on it will also be deactivated. In TT-RBAC different kinds of entities have different effective scopes to other entities. The entity effective scopes are shown in Figure 3. The arrows indicate the entities' effective directions. Their meanings are described as follows[2]:

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 4, Issue 10 , October 2017

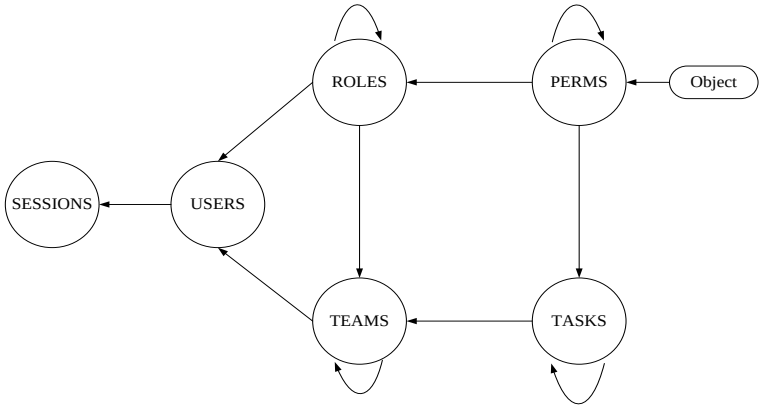

Figure 3: Effective scopes of TT-RBAC entities

- if an object is deactivated, then all the permissions that relate to it will be deactivated;
- if a permission is deactivated, then all the roles and tasks that relate to it will be affected. This permission will be withdrawn from them. This permission's state change may also affect other permissions that depend on it;
- if a role is deactivated, then all the users and teams that relate to it will be affected. This role will be withdrawn from them. This role's state change may also affect other roles that depend on it;
- if a task is deactivated, then all the teams that relate to it will be affected. This task will be withdrawn from them. This task's state change *may* also affect other tasks that depend on it;
- if a team is deactivated, all the users who relate to it will be affected. This team will be withdrawn from them. This team's state change may also affect other teams that depend on it;
- if a user is deactivated, all the sessions that belong to him/her will be affected. These sessions will be terminated.

A user obtains privileges through the active roles in sessions. A role can be activated as a session-role or a session-team-role. If a role is activated as a session-role, then the user will get all the permissions assigned to this role. If a role is activated as a session-team-role, then the permissions that this user can get are also decided by which tasks have been activated in the team. It means that the permissions of a session-team-role are filtered by the permissions of session-team-tasks. In a session the permissions available to a user are the union of all permissions gotten through session-roles and session-teams. In this section we investigate the TT-RBAC evaluation process.

In TT-RBAC systems authorization requests can be formatted as (user, object, operation). It states that the request user wants to perform the request operation on the request object. The request user information should include all the active session-roles and active session-teams in which also include the active session-team-roles and active session-team-tasks. The structure of TT-RBAC authorization request is shown in Figure 4. TT-RBAC authorization requests are evaluated by TT-RBAC decision makers [3].

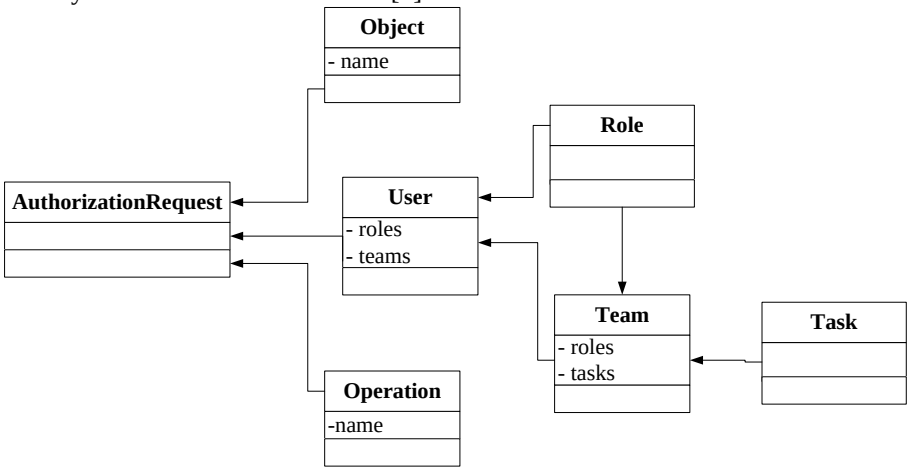

Figure 4: Structure of TT-RBAC authorization request

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 4, Issue 10 , October 2017

In order to improve the performance of TT-RBAC evaluation, it was developed a novel TT-RBAC evaluation mechanism, in which a decision maker is specially created for each authorization request. This decision maker is then used to evaluate the requested permission formed by the request object and request operation. In fact, this decision maker is a user object that is created according to the request user. So the decision maker is also called user decision maker. The role, team, team-role and team-task objects contained in a request user are not ground. For example, the role objects in a request user do not contain permission objects. On the contrary, all the objects inside a decision maker are ground. For example, all the role objects in a decision maker contain all the required permission objects, and these permission objects are also ground. After all the entities relating to a decision maker are ground, this user object is ready for making access control decisions [4]. The structure of TT-RBAC decision maker is shown in Figure 5.

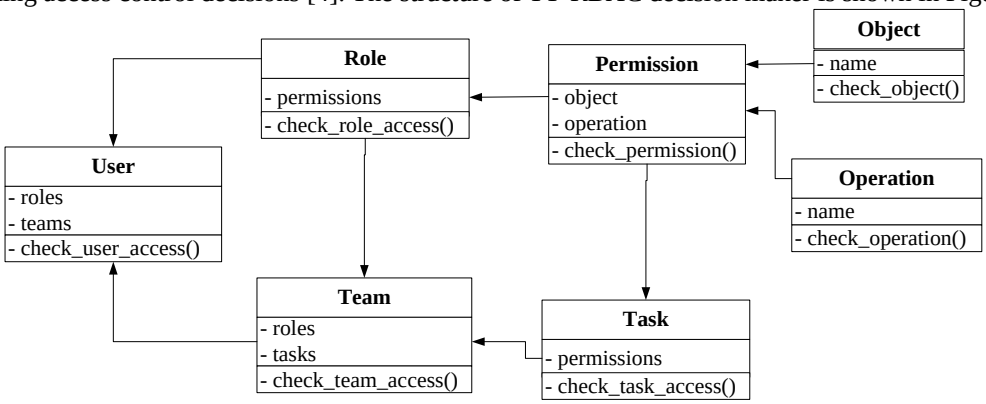

Figure 5: Structure of TT-RBAC decision maker

In order to accelerate the speed of creating user decision makers, some or all the role, team, task, permission, object and operation objects can be initialized and saved in some object containers when a TT-RBAC evaluation system starts up.

When an authorization request arrives, the evaluation system creates a special user decision maker according to the request user object, and then checks if this user can perform the required permission (ob, op) that is formed by the request object (ob) and request operation (op). The whole authorization check process is shown in Figure 6 and described as follows:

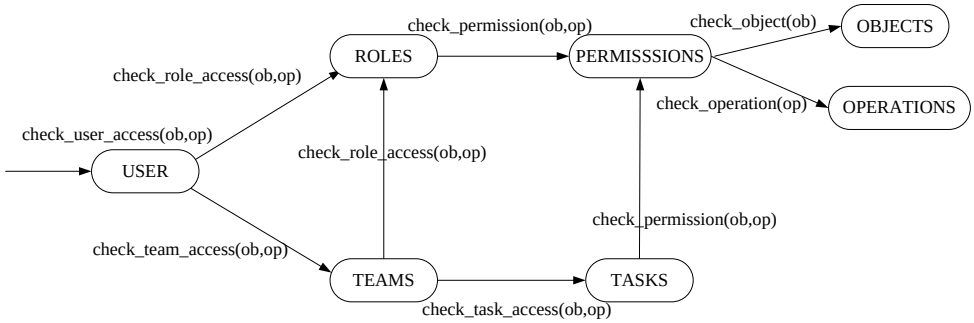

Figure 6: TT-RBAC authorization check process

- check_user_access is a method defined in User class. It checks if a request is permitted by a user object (user decision maker). This method first checks the state of the user object. Only when it is in active state, the check process continues to do the role permission check and team permission check, otherwise the check process stops and returns the value “false”. The role permission check is through invoking the method check_role_access defined in role objects. The team permission check is through invoking the method check_team_access defined in team objects. If there is any role or team permits this permission, the check process stops and returns the value “true”;
- check_role_access is a method defined in Role class. It checks if a request is permitted by a role object. This method first checks the state of the role object. Only when it is in active state, the check process continues to do the role permission check, otherwise the check process stops and returns the value “false”. The role

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 4, Issue 10 , October 2017

permission check is through invoking the method check_permission defined in permission objects. If the request is permitted by the role, it returns the value “true”;

- check_team_access is a method defined in Team class. It checks if a request is permitted by both team-roles and team-tasks. This method first checks the state of the team object. Only when it is in active state, the check process continues to do the team-role and team-task permission checks, otherwise the check process stops and returns the value “false”. The team-task permission check is through invoking the method check_task_access defined in task objects. If the request is permitted by both the team-roles and team-tasks, it returns the value “true”;
- check_task_access is a method defined in Task class. It checks if a request is permitted by a task. The check process is similar to check_role_access;
- check_permission is a method defined in Permission class. It checks if a request is permitted by a permission object. This method first checks the state of the permission object. Only when it is in active state, the check process continues to do the permission check, otherwise the check process stops and returns the value “false”. The permission check is through invoking the methodcheck_object defined in (permission) object, and the method check_operation defined in operation object. Only when both of them return the value “true”, this method returns the value “true”, otherwise returns the value “false”;
- check_object is a method defined in Object class. It checks if the input request object is equal to a (permission) object. This method first checks the state of the object. Only when it is in active state, the check process continues to do the object comparison, otherwise the check process stops and returns the value “false”. The object comparison is through comparing two objects’ names. If their names are equal, this method returns the value “true”, otherwise returns the value “false”;

check_operation is a method defined in Operation class. It checks if the input request operation is equal to an Operation object. The check process is similar to the check process of check_object.

IV. DEVELOPING SOFTWARE MODULE OF TT-RBAC

To install the program, you need to select the setup.exe application shown in Figure 7.

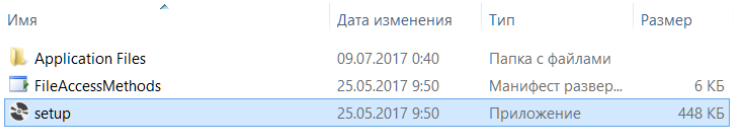

Figure 7. Selecting a file

After downloading the application, the following window will be displayed:

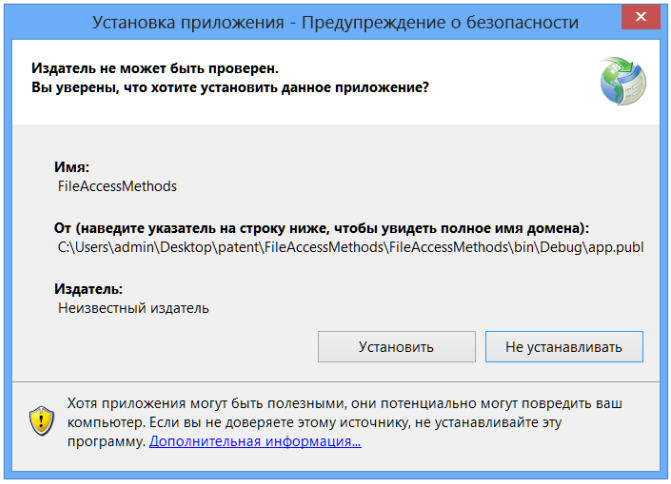

Figure 8. Installing the application

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 4, Issue 10 , October 2017

You need to click on the "Install" button and the program will start.The program consists of two parts. The first one is defining user rights by choosing users and the second one based on current user. Defining user rights by choosing users is below:

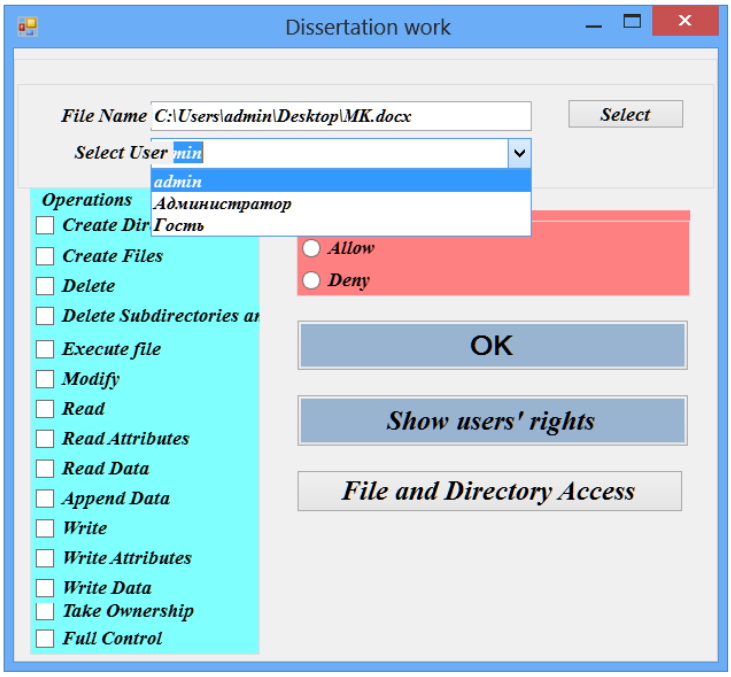

Figure 9: Defining user rights by choosing users

In this part of program defining users rights has been given by choosing users through file which has been shown [5]. Also It can be checked user rights that had been defined through the current file. In the below, the process of defining user rights through file which has been shown:

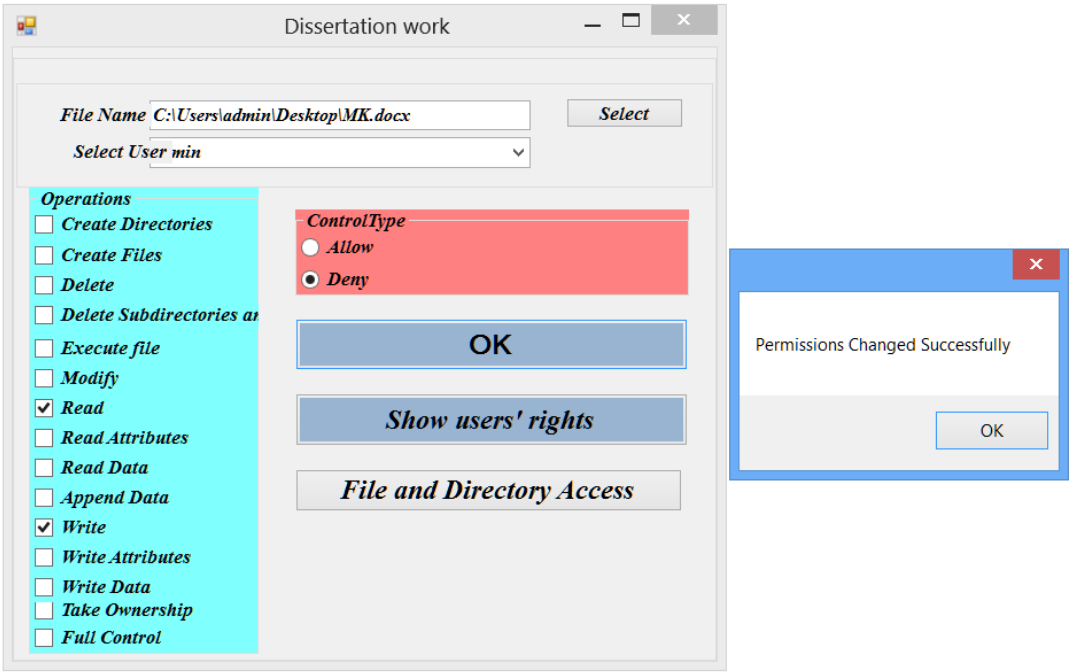

Figure 10: The process of defining user rights through file

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 4, Issue 10 , October 2017

In the below, the checking process of defining user rights that had been remarked through the current file:

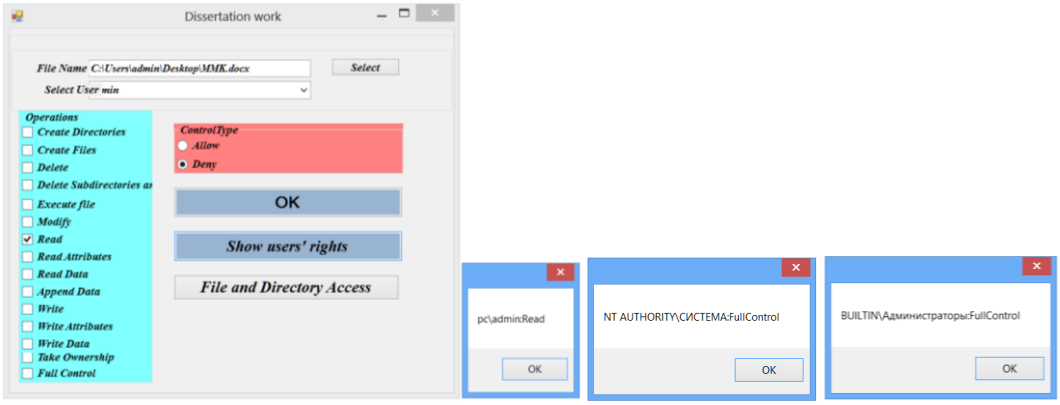

Figure 11: The checking process of defining user rights

The second part of the program consists of the two windows. In the first window, defining rights for file was located. In the second window, defining rights for folder was located. Access control for file is shown below:

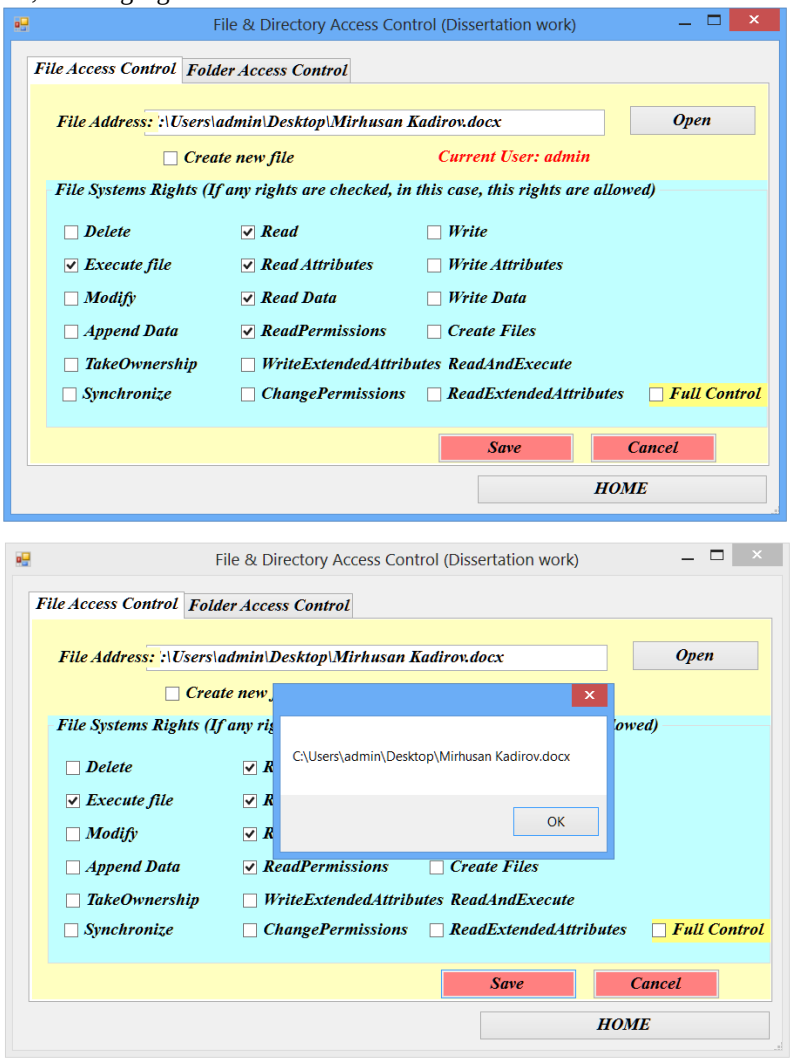

Figure 12: Access control for file

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 4, Issue 10 , October 2017

V. RESULT

After the program is distributed to the file "C: \ Users \ admin \ Desktop \ MMK.docx" using the program, the following error occurs when starting the MS Word document:

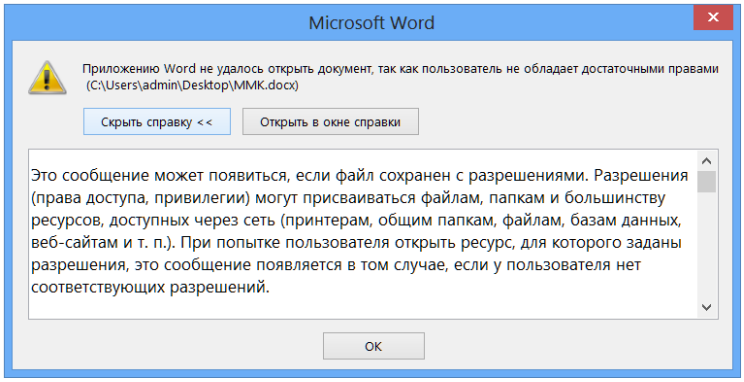

Figure 13. Error reference.

The program is designed to control the mark of the rights of users of the system, control permission for users of the system and control when the access rights are marked.

The program performs the following functions: marking the rights of users relative to the file, viewing a specific file when marking the rights of an existing user, marking the rights of users relative to the folder object.

Through the program, the organization of effective management of the rights of users of the system by marking, as well as preventing the use of data by users who do not have permission to access, is carried out.

The program introduced a mechanism for adding context constraints to any TT-RBAC objects. This access control model was motivated by the need to reduce time and raise the level for security administration.

VI. CONCLUSION

From this article it can be used as information resource that defining information, projecting users rights for resources, managing and controlling users.

A practical significance characterized by the fact that this requires access control models, policies, and enforcement mechanisms for collaboration resources. This article reviewed several research issues in this area. They are listed as follows:

- access control model that caters the requirements for access control in collaborative environments;
- authorization constraints that could be used for both expressing and enforcing authorization constraints;
- how to enhance and simplify the access control policies used in collaborative environments;
- policy and mechanism used for managing and enforcing multiple heterogeneous authorization policies in distributed authorization environments.

REFERENCES

[1] W.Tolone, G.Ahn, T.Pai, and S.Hong. Access control in collaborative systems. ACM Computing Surveys, 37(1):29–41, March 2009.
[2] W.Zhou, C.Meinel. Team and Task Based RBAC Access Control Model. In Proceedings of the 5th Latin American Network Operations and Management Symposium (LANOMS 2010), pp. 84-94, Petrópolis, Brazil, September 2010.
[3]Zhou W. Access control model and policies for collaborative environments :дис. – University of Potsdam, 2008.
[4] Rajaboevich G. S., Mirpulatovich K. M., Yakubdjanovich T. Z. The Methodology of the Ways for Increasing the Efficiency of Intrusion Detection Systems //International Journal of Engineering Innovations and Research. – 2016. – Т. 5. – №. 5. – С. 296.
[5] Kadirov M.M. Access control model and policies for collaborative environments. International Journal of Advanced Research in Science, Engineering and Technology, Vol. 4, Issue 7, July 2017, p. 4223-4229.

ISSN: 2350-0328

International Journal of Advanced Research in Science, Engineering and Technology

Vol. 4, Issue 10 , October 2017

AUTHOR'S BIOGRAPHY

Kadirov Mirhusan Mirpulatovich Assistant professor. Has more than 78 published scientific works in the form of articles, journals, theses and tutorials. Currently works at the department of "Information technologies" in Tashkent State Technical University.

Tulyaganov Zoxidjon Yakubdjanovich Assistant professor. Has more than 10 published scientific works in the form of articles, journals, theses and tutorials. Currently works at the department of "Information technologies" in Tashkent State Technical University.