Applying Quorum Role in Network Management

Edemilson da Silva[1] Altair Olivo Santin[1] Edgard Jamhour[1] Carlos Maziero[1] Emir Toktar[2]
[1]Pontifical Catholic University of Paraná – Graduate Program in Computer Science
Curitiba – PR – Brazil

[2]University of Paris VI, LIP6 Computing Laboratory
Paris – France

{*edemilson, santin, jamhour, maziero*}*@ppgia.pucpr.br, emir.toktar@computer.org*

***Abstract* — This work presents a proposal for extending the Role-Based Access control (RBAC) model to support activities that demand runtime mutability in their authorization attributes. Such activities cannot be subdivided in a set of subtasks executed sequentially neither can be accomplished by a single role. The approach presented allows the creation of quorum roles, which can only be activated in a session with the endorsement of a quorum of other roles. A prototype illustrates the application of our proposal in a network management scenario. In the illustrative scenario, a previously defined set of roles, by endorsement, activates a quorum role to perform a management task without the participation of the network administrator role.**

***Index Terms*: RBAC Constraints, Usage Control, Quorum of Roles, Network Management**

I. INTRODUCTION

THE Role-Based Access Control (RBAC) model adopts the best features of classical models, discretionary and mandatory, in the sense of being flexible like the first one and centralized like the second one. However, the RBAC restriction model is based on separation of duties (SoD), supporting the minimum privilege principle [1]. Therefore, the model dynamics is significantly influenced by separation of duties. Other types of restrictions have been considered and discussed in publications in the access control domain, but they have not been included in the RBAC reference model [2].

The separation of duty restrictions, combined with the minimum privilege principle, apply well to scenarios where a task can be divided in several sequentially executed subtasks. In that model, however, it is not possible to specify policies for environments where it is necessary authorization (endorsement) of a minimum set of principals (quorum) to accomplish a task [3]. Tasks with this sort of requirements are usually related to actions in heterogeneous or multi-disciplinary environments. In this case, the principals must agree to cooperate for executing an activity where single individuals would not have the required authorization to accomplish it. The quorum, in this context, does not aim to implement an intrusion or fault tolerance strategy, but to define a minimum (significant) set of principals that, by using their roles, grant to a third party the required authorization to accomplish a task.

The classic access control mechanisms generally verify the authorization only in the beginning of a session (subsequently to the authentication process). However, whether the authorization polices are modified during a session, the authorization requirements may be violated. In RBAC, policies are verified when a role is activated, even if the role activation is accomplished during a session. On the other hand, when an access is denied, the access control mechanism does not indicate the role that should be active in order to provide the required access right.

The $UCON_{ABC}$ model supports the modification of attributes of a subject or an object (attribute mutability) within an already established session [4]. Therefore, whether an attribute is modified during a session, its effect is immediately taken into account. However, a mechanism that implements such feature requires continuously monitoring of subjects' and objects' attributes.

This work employs attribute mutability, as defined by $UCON_{ABC}$, in order to propose an extension to the RBAC constrained model. The extension allows the activation of multiple roles (quorum) within a single session, to endorse a user to execute a task that could not be performed by a single role.

An example scenario involving a company with a faulty Internet connection (i.e., a BGP router not working) and an unavailable network administrator shows the proposal viability. In such a scenario, the Internet access needs to be restored urgently, but neither the company staff nor the vendor technicians hold enough authorization to access the router.

The remaining of the paper is organized as follows: Section II describes the RBAC consensus model. Section III describes our proposal. Section IV presents an example of application. Section V discusses the implementation aspects of our proposal. Section VI discusses some related works. Finally, section VII presents our conclusions and points to future developments.

II. RBAC CONSENSUS MODEL

The concept of role-based access control (RBAC) was introduced in the 70's. A more formal definition of RBAC was specified by NIST (National Institute of Standards and Technology) and a working group led by Ravi Sandhu, which defined the *Unified NIST RBAC Model* [5]. Subsequent work

978-1-4244-3487-9/09/$25.00 © 2009 IEEE

defined the *NIST RBAC Consensus Model*, which is a standard based on four elements: *Core RBAC*, *Hierarchical RBAC*, *Static Separation of Duty Relations* and *Dynamic Separation of Duty Relations*. This modular organization allows vendors to partially implement RBAC features in their products (as defined in the RBAC Standard ANSI INCITS 359-2004).

The Core RBAC model element includes sets of five basic data elements called users (USER), roles (ROLES), objects (OBS), operations (OPS), and permissions (PRMS). The relationship between such elements is shown in Fig. 1.

The main idea behind the RBAC model is that permissions are assigned to roles instead of users. The User Assignment (UA) is a many-to-many relationship (i.e., a user can be assigned to one or more roles, and a role can be assigned to one or more users). An important assumption is that roles must be activated during a RBAC session, to allow a principal to use the rights assigned to her. A session is associated with a single user, and each user is associated with one or more sessions. The Permission Assignment (PA) is also a many-to-many relationship between permissions and roles. A permission is an approval to perform an operation (e.g., read, write, execute, etc.) on one or more RBAC protected objects (e.g., a file, a directory entry, a software application, etc.).

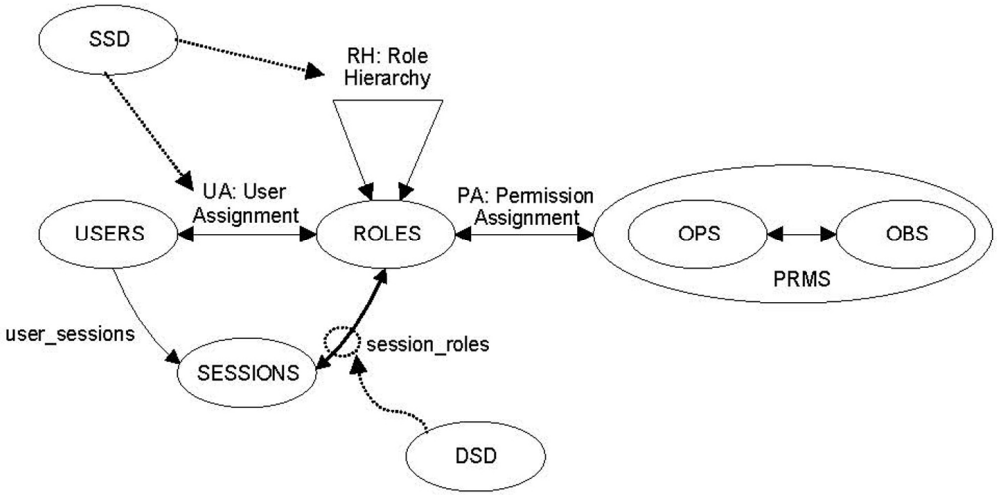

Fig. 1. RBAC NIST Consensus Model [5]

The NIST RBAC model adopts the separation of duty, which consists in dividing tasks that can lead to conflicts of interest into several subtasks – executed by different users [6]. This approach reduces the privileges assigned to individual users (minimum privilege principle). The separation of duty was discussed in details by Simon and Zurko [7]. In spite of the existence of several restrictions proposed by researches in access control area, the NIST RBAC model supports only dynamic and static SoD [8]; these restrictions are also presented in other models [4] [9].

The *Static Separation of Duty* (SSD) model introduces static constraints to User Assignment (UA) relationship, by excluding the possibility of users being assigned to conflicting roles. The RBAC model defines SSD based on two arguments: a role set that includes two or more roles and an associated cardinality (some roles can have a limited number of associated users [10]). For example, for constraining a user to assume the role 'r1' or 'r2', one must define a set {r1, r2} with cardinality 1 (i.e., the user can exclusive assume one role from the set).

The *Dynamic Separation of Duty* (DSD) model introduces constraints on roles that users can activate within a session. The strategy for imposing constraints on role activation is similar to SSD approach, using a set of roles and an associated cardinality bigger than one. It should be noticed that SSD imposes general constraints on which roles a user can assume, while DSD imposes constraints on which roles a user can simultaneously activate in a session.

Fig. 2 illustrates the dynamic for establishing an RBAC session, taken into account the constraints. A user initiates a session (event S1.0) by activating the RBAC controller through the user interface. At event S1.2, the roles assigned to the user are retrieved and, at event S1.4, the sub-roles are retrieved. The roles a user can activate in a session are shown at event S1.7. The activation restrictions are retrieved at event S2.2. RBAC controller verifies whether a user possesses authorization for activating the selected roles, by taking into account the session restrictions. If all steps are successfully executed, the selected roles are activated and the session is initiated for the user (event S2.7).

Recently, Park and Sandhu [4] have introduced the $UCON_{ABC}$ usage control model. $UCON_{ABC}$ is based in 3 kinds of statements: *Authorizations (A), oBligations (B), and Conditions (C)*. Authorizations consider attributes of principals and requirements of objects in access control decisions. Obligations evaluate required actions that must be accomplished by principals in the sense of usage control. Conditions impose restrictions based on environmental (system) variables or (location) attributes.

In order to define the models that compose $UCON_{ABC}$, Park and Sandhu considered three circumstances to apply the restrictions: before (*pre*), *ongoing* (*on*) and after (*post*) an

event. An event can be a session establishment, an object access, an object usage, a policy changing, etc. It should be noticed that *post* restrictions applied to an event e_i will affect only its subsequent events $e_{j>i}$, not e_i itself. In this paper, only the *pre-oBligation* (preB) and *on-oBligation* (onB) restrictions are relevant. The obligation restrictions must be accomplished in session initialization (*preB*) and during the session (*onB*).

Attributes are dynamically modified as the result of actions performed by principals, i.e., depending on actions of principals within the session [11]. Privilege attributes (A) can be modified during or after a session. The next section presents the proposal for extending the NIST RBAC model in order to support the *preB* and *onB* constraints defined by the $UCON_{ABC}$ model.

III. RBAC EXTENSION PROPOSAL

This work considers scenarios in which minimum privilege approach (i.e., dividing a task in subtasks sequentially executed) is not feasible. Instead, it is assumed that a predefined quorum of roles temporarily authorizes a principal to execute a task that herself would not be successful. In this case, the attributes of a principal (third party) are modified in order to grant the authorization assigned by the quorum.

As explained in section 2, the NIST RBAC model does not apply to scenarios in which principal's attributes can be dynamically modified (during the session). Therefore, we propose extending the NIST RBAC model for supporting *obligations* constraints from $UCON_{ABC}$, for quorum-based access control. The proposed extension defines two types of roles: *simple* and *quorum*.

Simple roles are defined as in RBAC original model, in which a principal can activate any role assigned to her that satisfies separation of duty constraints. The quorum roles can be activated only with the authorization of a quorum composed by a pre-defined set of simple roles, i.e., a quorum role cannot be activated by a simple role alone.

Principals that compose a quorum activate their roles only to endorse the quorum role activation and not to use the privileges assigned to their simple roles. The participation on quorum role activation does not forbid using the associated role in a normal user session. In other words, any user participating in a quorum role activation for a given session can also activate that single role in her own session, when needed.

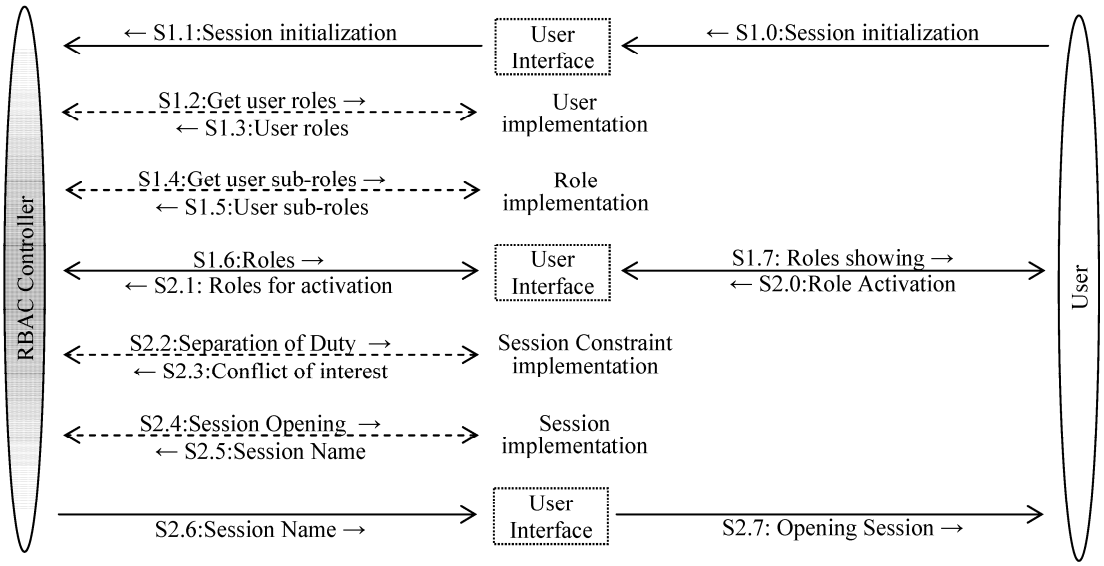

Fig. 2. Events for activating a RBAC session

The constraints for activating a quorum role in a session have been divided into *preB* and *onB* restrictions. Quorum role imposes conditions and pre-obligation constraints. Conditions define environmental constraints such as date/time, source/destination addresses, etc. Pre-obligation constraints define which simple roles are required to activate a quorum role establishing a session. When a principal is already in a session, the pre-obligations become *on-oBligations* (pointing out the simple roles required for activating a quorum role in a session).

The minimum privilege, SSD, and the association of a quorum role to principals are conditions in the $UCON_{ABC}$ model, while the required quorum of principals and DSD denote obligations. Minimum privilege and separation of duty adopted by RBAC are not modified by our proposal. However, if a principal does not hold enough privileges for executing a task, it may receive additional privileges on her session for doing it, provided she was previously associated to the respective quorum role. In general, the more critical the task is (i.e., requiring more responsibility or authority), the greater should be the quorum of roles.

A bank office scenario illustrates a typical example of quorum authorization, in which a cashier can process checks up to U$5,000.00, without requiring a supervisor approval. Any check above this limit requires the authorization (endorsement) of a supervisor. Fig. 3 shows how the proposed extension would be applied to this example scenario.

In the case presented in Fig. 3, the cashier supervisor (Bob) activates his role in cashier's session (Alice) in order to authorize the payment of a check corresponding to values greater than U$5,000.00.

Fig. 4 illustrates the proposal dynamics for establishing a RBAC session, considering the proposed extensions. The user initiates a session (event S1.0) by using a user interface, which activates the RBAC controller.

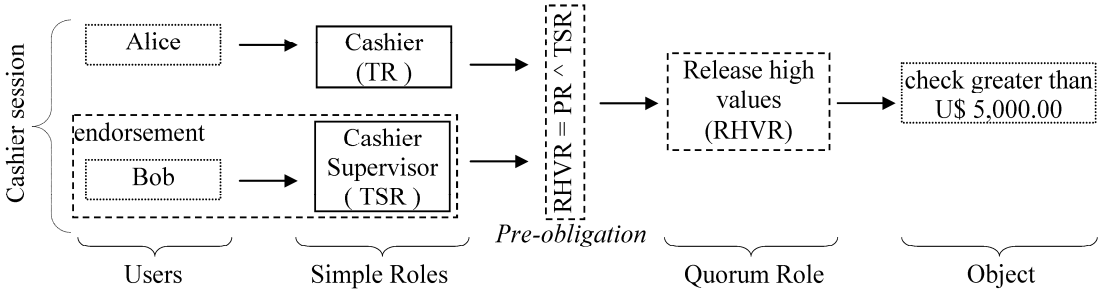

Fig. 3. Activating a quorum role

After retrieving the roles assigned to the user, the RBAC controller shows the set of roles available for activation (event S1.7). At event S2.4, the controller verifies if a simple or a quorum role is to be activated. For a simple role, after verifying the conditions for its activation, the session is open (event S2.5). Otherwise, in the activation of a quorum role, it requires to validate a quorum of principals (event S2.5.6). Whether all events preceding event S2.5.7 are successfully executed, the session is open. The user is then notified about the success or failure of session opening in event S2.7.

Since this proposal supports a scenario where attributes can be dynamically modified (mutability of attributes), i.e., at any time an attribute relating to a user or object can be modified, the obligations must be continuously verified (event S3.2), as they correspond to on-obligation containments in $UCON_{ABC}$.

The quorum role activation can be limited to a time interval. In such a case, when the time expires, the RBAC controller deactivates the quorum role and resets the permissions according to the user status before quorum role activation. Time constraints are defined by conditions in our proposed model.

The next section describes a simple scenario showing the usage of our proposed extensions. It should be noticed that this scenario can be adapted to other domains without modifying the procedures presented here.

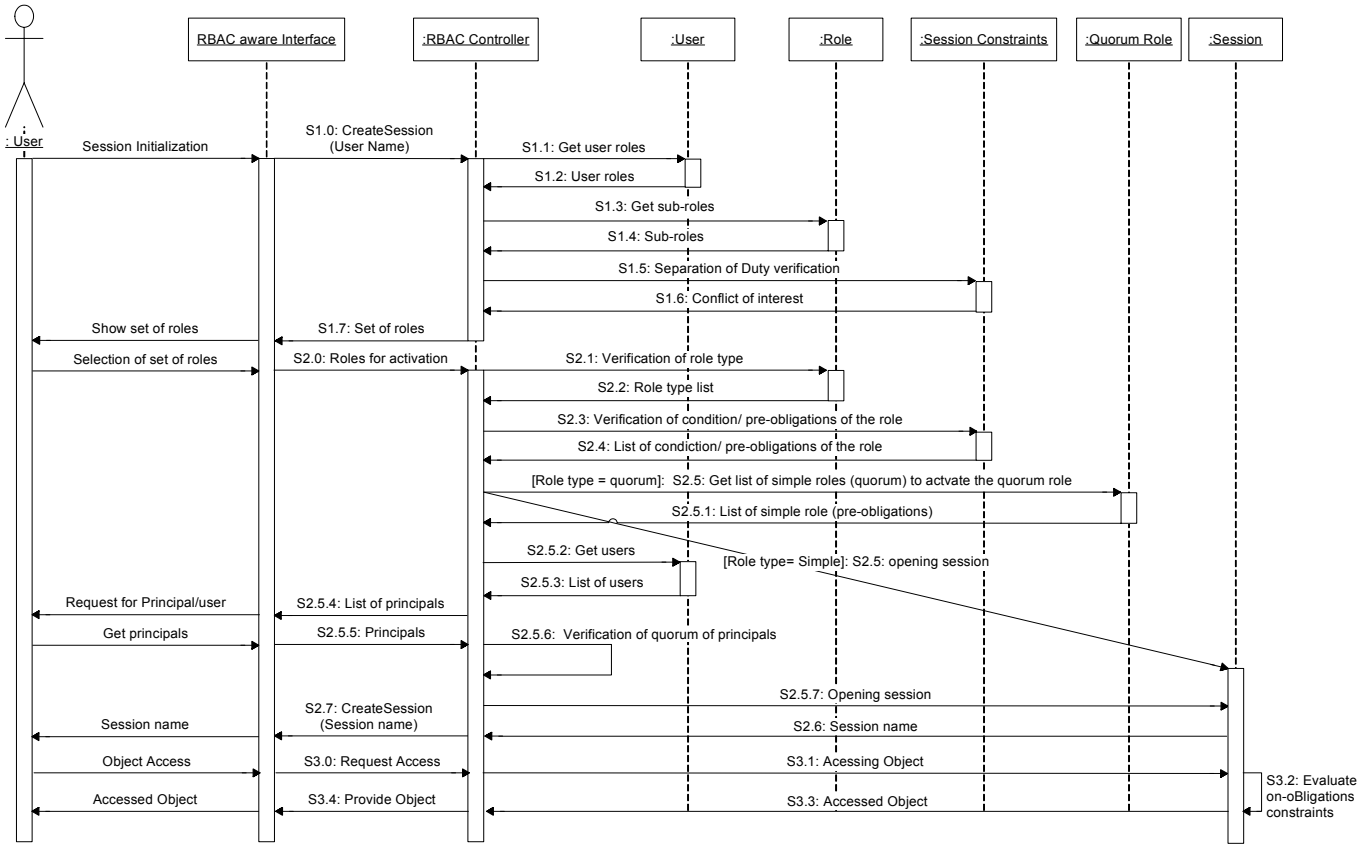

Fig. 4. An example of session establishment using the proposed RBAC extension

IV. SCENARIO: NETWORK MANAGEMENT

Network and system management are critical tasks for most organizations. In large organizations, these functions are performed by distinct roles. In this proposed scenario, we consider the following simple roles: R1 (guest), R2 (system operator), R3 (system administrator) and R4 (network administrator).

Table I illustrates the main functions assigned to each role. Table I also defines two quorum roles especially deployed, QR1 and QR2. The description of the main incumbency of each role gives a rough idea of network management policy.

Considering Fig. 5, let us suppose that the network administrator (Eve) is temporarily unavailable and a problem happened on a critical BGP router, preventing access to Internet. In order to restore access as soon as possible, another operator with the required knowledge about BGP must be temporarily authorized to accomplish the router maintenance.

Several solutions could be considered in order to avoid or minimize this difficult situation. Among the possible solutions, one could be to authorize an external technician (e.g., from equipment vendor) to do the maintenance. This solution can lead to another problem: how to create a temporary administrator account to an external principal without violating the organization policy?

We assume that the chosen solution is to authorize the network technician designated by the equipment vendor to do the maintenance. In order to get access to the BGP router, the technician should receive the authentication credentials (e.g., password) for a guest user, for example. By default, a guest user does not hold any privileges to access the BGP router. The quorum role QR1 associated to the guest user can provide such privileges, but it requires endorsement of role R2 to be activated.

Table I. Some roles of an organization

Role	Description of the main responsibility of each role
R1	Guest: very limited management access privileges to system
R2	System Operator: limited management access privileges to system
R3	System Administrator: unlimited management access privileges to system
R4	Network Administrator: unlimited management access privileges, applied only to network devices in system
QR1	Unlimited management access privileges to system and limited management access privileges to network devices
QR2	Unlimited management access privileges to network devices

When Alice activates the system operator role (R2) in the already established guest session, the RBAC controller automatically activates the QR1 role. By activating the QR1 role, the guest user receives limited network privileges allowing her to verify the current configuration and to run basic tests on the BGP router. However, she still does not get enough administration privileges to modify the router configuration (i.e., it is not able to perform an *enable* command on the BGP router).

Suppose that, after verifying the current status of the BGP router, the technician concludes that the router configuration should be modified. In that case, QR2 is required to be activated. The on-obligation for activating QR2 defines that it requires the endorsement of both system operator and system administrator. By the activation of both roles R2 and R3 in guest user session, the privileges of QR2 quorum role that allows modifying the router configuration will be granted, enabling the maintenance task.

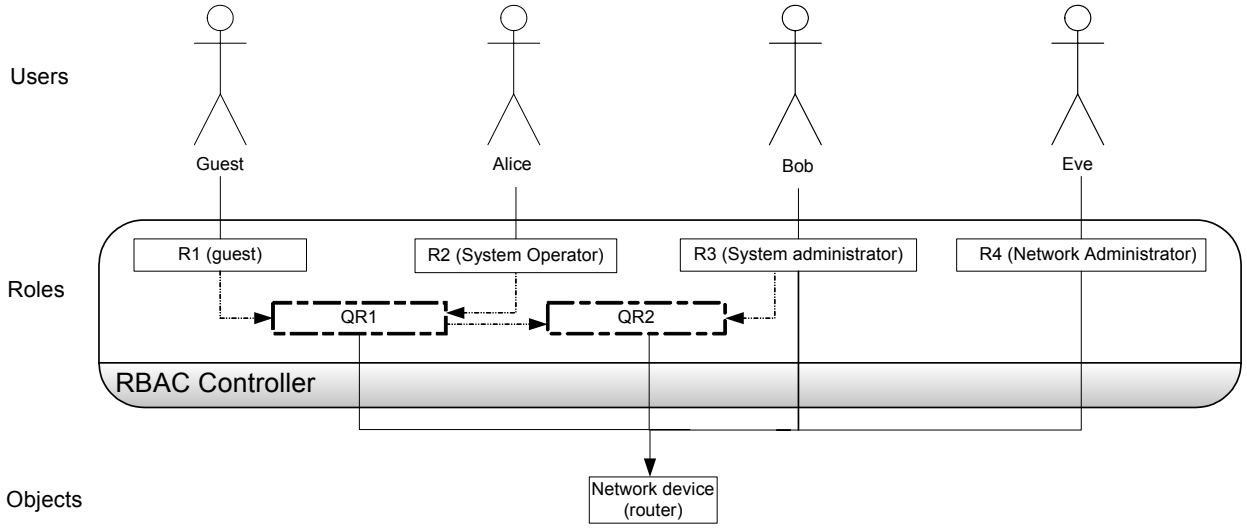

Fig. 5. Overview of quorum roles activation

V. IMPLEMENTATION ISSUES

A prototype was implemented by extending the RBAC/Web implementation from NIST [12]. Such implementation corresponds to an Intranet in which RBAC is used as the authorization scheme for controlling access to web pages of an HTTP server. In RBAC/Web implementation, users correspond to server's user accounts. HTTP transactions that can be performed by users (through their roles) on HTML pages represent the RBAC permissions. As a proof of concept, the prototype implements the scenario described in section IV (Fig. 5). Access control is implemented through an API (implemented by NIST in C and Perl).

In order to support quorum roles, an attribute called *Access Level* (AL) has been added to each RBAC role. This attribute stores an integer value for each role in the system. Integer values allow representing a set of simple roles required to activate a quorum role using a single attribute field.

In order to avoid misinterpretation of AL values, to each simple role is assigned an integer value resulting from the expression 2^n, with n = 0,1,2,3 etc; n denote the number of single roles in the system. For quorum roles, the AL results from the expression:

$$AL = \sum_{x=0}^{(n-1)} k.2^x$$

where k may be: $k = 0$ (default) or k = 1 (if a simple role with AL = 2^x is required for the endorsement).

Considering in the scenario example (Fig. 5) that roles receive the following ALs: R1 = 64, R2 = 32, R3 = 16, therefore, the quorum role QR1 = 96 (resulting from R1 and R2 ALs) and QR2 = 112 (resulting from R1, R2 and R3 ALs).

The prototype has been implemented by adapting the RBAC/Web classes. It should be noticed that to support our proposal some new methods have been introduced.

A private method *writeAccessLevel* was included in *Role* class. This method is executed at the moment of role creation by *addRole* method. It stores the role and its AL into an *AL_role* file to record their relationship. When excluding a role (by using the *DeleteRole* method) the AL is set inactive instead of being deleted; the AL is preserved by method *rmAccessLevel* for auditing purposes and it is never reused by the system.

The method *CreateSession* from *Session* class invokes private methods: *create_login_choices* (creates a set of roles that a user can activate – which are not constrained by dynamic separation of duty) and *writeARS* (store the roles chosen by user in the file *user_name.active_roles*). The method *checkAccessLevel* is invoked before the method *writeARS* – this method verifies the presence of quorum roles among the set of roles the user is trying to activate.

When a quorum role is activated, the *writeAccessLog* method stores additional information in a log file, such as activation date/time, quorum role activated and principals/roles that endorsed the activation. All the actions related to the activation of quorum roles generate additional log entries. All roles activated in session are also recorded in the log file.

In the prototype, an HTML page is offered to perform *login* into the BGP router management system. Initially, when the *guest* user performs *login*, she can only *browse* some router information. In order to get access to router configuration and perform some simple tests, the RBAC controller requests the system operator *login* (i.e., the endorsement role R2) to activate the quorum role QR_1. The same procedure is repeated when the *guest* user tries to modify router configurations. In such a case, the *login* of both, system operator and system administrator are required.

If the *login* and password are successfully checked, the *guest* user receives privileges to perform the requested operation, but the system operator and the system administrator do not receive the same privileges; they only endorse the activation of QR2 role and continue their work as simple users in their respective sessions.

The validity time of the quorum role endorsement's activation is an optional constraint offered by the endorsement screen of the implemented prototype.

VI. RELATED WORKS

This section discusses some related works that propose alternatives to RBAC NIST constraint model and $UCON_{ABC}$ implementation for collaborative computing.

The framework proposed in [13] is based on restrictions that are not part of the RBAC core model, but are taken into account for the access control decision. In such a case, the restrictions are called "context constraints" and impose dynamic restrictions to the authorization in each application context (e.g. ubiquitous and pervasive computing). The proposals present by [13] offer facilities for expressing the separation of duty in a way not addressed by the original model. However, the proposed usage of *SoD* follows the original RBAC behaviour, by separating a task into subtasks, organized in a way that they impose the participation of distinct principals for executing each subtask.

The work [14] applies $UCON_{ABC}$ to Virtual Organizations in the context of grid security infrastructure. In that proposal, obligations are not addressed; only authorizations and conditions are considered. It applies the PCIM outsourcing model (RFC3460), based in Policy Enforcement Point (PEP) and Policy Decision Point (PDP) entities to implement $UCON_{ABC}$. The continuous evaluations of attributes' mutability are considered by the PDP, which updates subject/object status according the current policies. The prototype is implemented as an Apache WebDAV module (http://webdav.org/mod_dav/). The proposal presented in [14] indeed implements classical access control in a dynamic environment considering continuity and mutability of attributes.

In our proposal, the endorsement to activate a role is not evaluated according to mutable access control attributes, to avoid possible role conflicts due to runtime rights updates, for example. Such ongoing updates can lead to role conflicts after the RBAC DSD evaluation. Instead, our proposal applies continuity and mutability to RBAC in a sense of roles'

usage in a quorum of principals. Also, the attributes' mutability should allow the RBAC controller activating or deactivating automatically roles, not changing authorization attributes, as aforementioned.

VII. CONCLUSIONS

This work presented extensions to the consensus NIST RBAC constraints model. We presented an approach capable of accommodating the requirements of an environment in which separation of duty cannot be implemented by separating a task into subtasks executed by distinct roles. In our proposal, a quorum of distinct roles is required to endorse the execution of a task that a single role could not perform alone. The RBAC NIST standard model has no support for this kind of constraint.

The RBAC NIST model does not support the mutability of attributes of subjects. However, by using a quorum role is possible to get a behaviour similar to that proposed by $UCON_{ABC}$ model. Since there is no implementation considering $UCON_{ABC}$ obligations, the proposal could be applied to evaluate issues regarding mutability and continuity in real systems.

There are no identifiable security risks related to a potential unauthorized activation of a quorum role, as the activation and control of role is implemented by the RBAC controller, not by a user.

In the access control mechanisms there is no hierarchy among access level (AL); any combination of single roles can activate a quorum role – this is required to support system scalability. As the combination of ALs generates a new AL, a quorum role must be explicitly assigned to a user, avoiding misinterpretations related to the combination of attributes used for creating policies, for example.

The prototype shows that the proposed strategy can be easily implemented by a few modifications on the RBAC NIST framework. In order to illustrate the applicability of proposed approach, a typical network maintenance scenario was evaluated. In such a scenario, a guest user was temporarily authorized by employees of the corporation to perform a maintenance task, without violating the corporative security policy and without modifying role assignments.

According to quorum role strategy, by using obligations, a user receives information about which roles are required to activate for performing a task she currently has not enough privileges to execute. This approach is innovative against traditional access control mechanisms, which in such a case only deny the access and do not inform the user about what she can do.

Presently, the prototype authentication mechanism is based on login/password. This mechanism should be enhanced to a more flexible scheme, maybe based on credentials, for example.

REFERENCES

[1] Saltzer, J.H., Schroeder, M. D., "The Protection of information in computer systems," In *proceedings of IEEE*, vol. 63, no. 9, 1975, pp. 1278-1308.

[2] Ferraiolo, D., Sandhu, R., Gavrila, S., Kuhn,D.R., Chandramouli, R., "A Proposed Standard for Role Based Access Control," *ACM Transactions on Information and System Security*, vol. 4, no. 3 , 2001.

[5] Sandhu, R., Ferraiolo, D., Kuhn, R., "The NIST Model for Role-Based Access Control: Towards A Unified Standard," In *Proceedings of ACM Workshop on Role-Based Access Control*, Berlin, Germany, 2000.

[3] Shoup, V., "Practical threshold signatures," In *Proceedings of Eurocrypt*, 2000.

[4] Park, J., Sandhu, R., "The $UCON_{ABC}$ usage control model," *ACM Transactions on Information and System Security*, Vol. 7, Issue 1, 2004.

[6] Brewer, D., Nash, M. "The Chinese wall security policy," In *Proceedings of the Symposium on Security and Privacy,* IEEE Press, 1989.

[7] Simon, R., Zurko, M. E., "Separation of Duty in Role-based Environments," In *Proceedings of the 10th Computer Security Foundations Workshop*, 1997.

[8] Ferraiolo, D., Kuhn, R., "Role-Based Access Control," In *Proceedings of NIST - NCSC National Computer Security Conference*, 1992.

[9] Nyachama, M., Osborn, S., "The Role Graph Model and Conflict of Interest," *ACM TISSEC*, vol. 2, no. 1, 1999.

[10] Sandhu, R., "Role-Based Access Control," In *Advances in Computers*, Academic Press, v. 46, 1998.

[11] Jaehong, P. Xinwen, Z. Sandhu, R., "Attribute Mutability in usage control," In *proceeding of DBSec*, pp. 15-19, 2004.

[12] Ferraiolo, D., Barkley, J., Kuhn, R., "A Role Based Access Control Model and Reference Implementation within a Corporate Intranet," In *Proceedings of NIST*, 1999.

[13] Strembeck, M., Neumann, G., "An Approach to Engineer and Enforce Context Constrain in RBAC Enviroments," *ACM transaction on Information and System Security*, Vol. 7, no. 3, 2004

[14] Zhang, X., Nakae, M. Covington, M., Sandhu, R. A., **"U**sage-based authorization framework for collaborative computing systems," In *Proceedings of the eleventh ACM symposium on Access control models and technologies*, 2006, pp. 180 – 189.