International Journal of Computer (IJC)
ISSN 2307-4523 (Print & Online)

http://ijcjournal.org/

Modelling a Policy Role Based Access Control Mechanism for Task Delegation in a Nomadic Environment

Osaremwinda Omorogiuwa[a*], Stella Chinyere Chiemeke[b]

[a]*Department of Computer Science & Information Technology Igbinedion University, PMB 001, Okada, Edo State, Nigeria*

[b]*Department of Computer Science, University of Benin, Benin City, Nigeria*

[a]*Email: ask4osas@iuokada.edu.ng*

[b]*Email: schiemeke@uniben.edu*

Abstract

Nomadic environments are governed by standard principles and lay down rules that should be followed to enable it meet set aim and objectives. More recently, nomadic environments have virtually employed the use of Role Based Access Control (RBAC) Mechanisms to proffer access control solutions to role assignments which ordinarily would have be accomplished manually. In modelling systems for users in a nomadic environment, most RBAC mechanisms does not effectively consider the security lapses related to human to human task delegation. To avert this lapses, during system modelling and design, there is the need for software developers to consciously put into consideration the inclusion of organizational policy rules guiding role assignment and task delegation in a secured manner. Failure to do so, may create usability and security issues resulting from a delegatee abusing his privileges in performing other tasks of the delegator. This paper is therefore aimed at using mathematical and algorithmic methods to model a policy based approach in implementing the Role Based Access Control mechanism for users in a nomadic environment. With this approach, task delegation can be implemented in a usable and secured manner.

Keywords: Nomadic Environment; Policy Based RBAC; Algorithms; Tasks; Delegation; Mathematical Modelling.

---

* Corresponding author.

1. Introduction

In this current trend of Information Technology (IT) and a dynamic shift in IT service delivery, system users are essentially nomads. Users want to be able to get access to their resources and render services to their organization where ever they are with minimal limitations. A distinguished scholar in [1] argues that Information world must no longer be seen through the traditional client/server eyes where wired computers are exchanging data packets with fixed servers in a Local Area Network. This is because computing devices have become portable, integrated and are in the possession of almost every users of an organization. It is important to note that the next generation networks is considered to be a "user-centric" in the technological world [2]. .As a result, current IT services are leveraged on cloud computing, ubiquitous computing and nomadic computing platforms.

The concept "user-centric" implies developing applications that is highly platform independent and usable to users irrespective of their portable devices or the network environment they find themselves. At design time considering "users satisfaction" is becoming core in software requirement specification. Software development is no longer only geared towards efficiency and functionality, but also towards portability and usability. Concerted effort is therefore required that while trying to make a system more usable, the security level is not compromised and vice versa.

Explicitly [3] defined nomads as the users who are mobile and have electronic appliances (such as PocketPc, Palmtop, Personal Digital Assistant (PDA) in their pockets to get access to the remote information spaces no matter which device they currently are working with and no matter where they are. Nomads want to be able to use their portable devices to gain access to organization applications while still carrying out other personal functionalities. Nomadic environment can be found in hospitals, banks, government and business organizations provided the ICT infrastructure in such environment allows for users to freely access and render services from any terminal irrespective of their location.

In nomadic environments, providing users with the system support needed to provide rich set of computing and communication capability as they move from one location to another is highly needed. Comparatively [4] observation conforms to [1] concept that user can use any access to services and any terminal in environments where the infrastructure exists to meet organizational needs.

2. Some Related Literatures

Different literatures have explained nomadic environment from different perspectives, Reference [5] explained a nomadic environment to be a well-connected communication infrastructure that provides users with services they require. In such environment, users (known as Nomads) move from one location to another to carry out services. Reference [6] described "a nomadic environment as one in which users carrying wirelessly connected devices to enter places and use local services associated with those places. The services may be implemented and provided locally by the appliances in the place". There is consistency in both [5,6] concepts of a nomadic environment. There is however a fairly different perceptions coming from [7] who argued that following the

technological intention of the initiators of nomadic computing, anyone who accesses his or her computing environment from different locations is a computing nomad or simply a nomad.

Nomadic environment does not only exist on a Local Area Network, but may also exist on other networks. The emphasis on nomadic computing is user-centric, it is centered on system support that will enable a user (nomad) to carry out specific task or gain access to resources from any location within any given network platform.

Implementing role assignment and task delegation in a nomadic environment, Reference [8] suggested the use of a Role Based Access Control mechanism where the rights an employee has are determined by its position in the enterprise. The decision to grant or deny access is delegated to the server, which maintains the mapping between agents and roles, and a database listing the privileges of each role. Also, Reference [9] proposed a concrete design of a mechanism that supports policies for regulating access to information via corporate Intranet. They argued that in order for corporate webs to reach their full potential, access control mechanisms that can express regulations and practices governing businesses are needed and showed that current web technologies provide only limited support for this purpose.

3. Entities, Roles and Groups in a Nomadic environment

Users Role Assignment in nomadic environment provide a superficial resemblance to already established concept of user groups, Entities, which is widely used for access control purposes, especially in association with Role Based Access Control systems. However, it is imperative to discuss these three components.

(i) The main aim of having GROUPS in nomadic environment is to collect users according to their responsibilities. Figure 1 depicts this group (user, roles and the entities contained). Groups are classification of users according to their responsibilities (roles). In contrast, roles are created to collect unique sets of permissions, each being the set of permissions necessary to carry out some associated (assigned) duty responsibilities [10].
(ii) A user after being assigned to a group is a member of this group at all times and in all circumstances. However, users can be assigned different roles in a group.
(iii) Entities are sets of task(s) contained in a role. A user must belong to role in an organization before he can be assigned any job functions.

In any organization, the division of work produces many roles with differing responsibilities and job functions. Although the final set of roles depends on the structure of the organization. An organization consists of persons belonging to various administratively or physically divided staff groups, such as a department, a division, or a project team that are referred to as structural units. Various job positions and specific activities arising in structural units of an organization are represented by user roles. A user role is associated with users that fill a job position or perform a specific activity and permission that describes this position or activity. The task(s) performed by users are referred to as Entities.

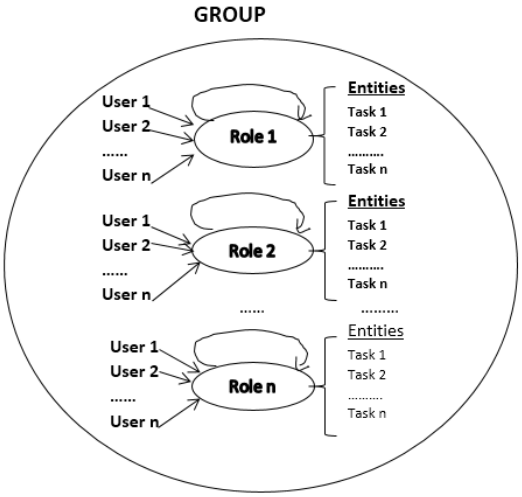

Figure 1: A Schematic diagram illustrating the relationship between Groups, Roles and Entities.

4. Research Focus

This paper focuses on modelling activities that will allow user to user task delegation in a nomadic environment. As such concerted effort to model organizational policy set and rules that will enhance secured task delegation in a nomadic environment will be represented.

First, a role (R) is a job functions or job title within the organization that represents authority and responsibility conferred on members of the role. A role can be seen as a collection of objects (entities) to accomplish a job function. These objects consist of the unique tasks that can be performed by job title. To represent this mathematically, if R defines a role then

$$R = \{task_1, task_2, task_3 \ldots task_n\} \quad (1)$$

Whereby $task_1$, $task_2$, $task_3$ … $task_n$ represents the various tasks assigned to a role. A role therefore is a collection of tasks that can be performed by a user.

For example, in the hospital environment, a doctor can perform surgery ($task_1$), diagnose illness ($task_2$), and administer treatment ($task_3$) on a patient. In performing these roles, the doctor has access to all information contained in the patient database. However, doctors are expected to perform numerous shifts and attend to various patients within shifts, and as a result, they are obliged to delegate another doctor to assist them in carrying out some of these tasks. And to accomplish this procedure, he must transfer his access rights to the delegatee who must assume the privileges of the delegator.

This approach has some security flaws as the delegatee may explore other unauthorized tasks contained in the role of that particular delegator. Given an access control policy of groups an equivalent role based policy can be

constructed by assigning a user to a role if that user is a member of a group that maps to the same set of permissions as that access control policy can be transformed to groups. This is can be achieved by making the user a member of the groups and associating this group with the set of permissions that were assigned to a role to which the user was authorized.

5. Human to Human Delegation

In this paper we explore the concept of delegation in context of RBAC. The basic idea of delegation is that some active entity in a system delegates authority to another active entity to carry out some functions on behalf of the former. Delegation in computer systems can take many forms: human to human, human to machine, machine to machine and perhaps even machine to human. In this research we focus on the human to human form of task delegation in computer systems. Specifically we consider the ability of a user in a role to delegate his role membership to another user who belongs to some other role. To perform specific task, we develop a simple but practically useful policy based approach to advert the problems associated ordinarily with the use of the Role Based Access Control (RBAC) Mechanism. It is a known fact that the most flexible form of delegation is impersonation, whereby the rights grantor allows the receiver to assume the identity of the grantor and perform any action on the grantor's behalf. However, this is also the least secure, since the delegated rights could be easily abused. Therefore, delegation should be limited to the specific task that is required.

6. Methodology

First a policy based access control mechanism was formulated based on the generic context-based access control policies in [11]. The generic policy as represented in [11] is shown in equation 2.

$$\text{Generic\_Policy}\ (P_i) = [U, P_{set}, (ac, e)\ \text{enable\_bit}] \quad (2)$$

Whereby

i. U (User) is an identity assigned to resource requestor (eg login, identifier, name group, role) in the current access context.

When the identity of requestor is omitted or it is assigned the value one, only the resource requestor assigned to the access context (ac) that meets the context constraint described in the expression (e) will get access permissions on the perfected resources.

ii. P_{set} is a set of one or more permission. Let P be permission in the set P_{set}. P is a tuple that defines the relationship between a resource and an operation ($P_i = (r, 0)\ P = 2^{Rx0}$).

iii. Ac is access context (ac $\subseteq$ AC) that describes the set of permission P $\in$ P_{set} to Users that satisfy the ac. Only the user that part of the context activating that ac will get the set of permission P_{set}.

iv. e is a context constraint expression defined using the Generic Context Condition Language (GCCL). This expression can be enforced by attributing the current value of access context objects;

v. Enable_bit indicates if the associated policy is enabled or disabled. Enable_bit has the value 1 if the

policy is enabled and 0 if the policy is disabled. By using this bit, it will be possible to maintain a policy register on access control policy repository.

For each enabled access control policy in the policy repository, we need to verify the ac by replacing the current values of context objects on the context constraint expression e. If the expression e is true, then the associated set of permission will be granted to affected users.

However, to implement organizational policies, equation 3 is formulated to represents a definition of a policy set that will be suitable for users in nomadic environment as follows;

$$NP_i = [Nu, Pset, (ac), binary\_bit] \quad (3)$$

i. NP_i represents Nomadic Policy Set
ii. Nu represents the various Nomadic users
iii. P_{set} represents the various policy sets that can be used to allow/disallow users to carry out job functions.
iv. ac represents access context. For this research study, we will consider time and organizational job schedule as our contextual information. With this, we do not need the e in the generic policy definition. In the Nomadic Policy set, the time and organizational job schedule can be used by the system to determine whether a nomadic user can be allowed or denied access to performing task(s). Other contextual information can be used as constraints and policy decision rule to allow/deny users.
v. Binary _bit is used to represent a true or false value or a 0 or 1 value at the end of the policy evaluation. If a 1 value is returned then user can access privileges to carry task else user is denied privileges.

Nomadic users (during task delegation) and system administrators can define a set of policies that is represented formally in equation 4 as follows:

$$\text{Nomadic-Policy \_ Set } (Pol_{set}) = \{P_i | P_i \text{ is a policy, } i >= 0 \text{ and } i \square N\} \quad (4)$$

Therefore in a nomadic environment the policy sets consist of system administrator defined policy sets which is based on the roles contained in the organization, role hierarchies and permission granting. The user level policies consists of delegation policies that allow a user who belong to a role to delegate responsibilities to another user.

Generally, a mathematical model representing the Nomadic Environment Policy Set (NP for short) is shown in equations 5 and 6 as follows;

$$NP_{set} = PSysDfnPol_{set} + UserDfnPol_{Set} \quad (5)$$

and

$PSysDfnPol_{set} \square UserDfnPol_{Set} = \square$ □□□□□□

□□□□□□□□□□□□□□□□□□□□□

Equation 6 implies that the system administration component cannot be responsible for user to user delegation of responsibilities.

i. $PSysDfnPol_{set}$ represents the policy sets that can be defined by the system administrator.
ii. $UserDfnPol_{set}$ represents the policy set that can be defined by the user during task delegation. This is required because this research focus on nomadic user task delegation which is often the case in nomadic work environment. However, it should be noted that this policies can only be used during delegation period for a delegatee. After revocation, the policy is erased from the policy repository. However, there is always an audit trail that keeps track of all activities within the nomadic work environment.

$PSysDfnPol_{set}$ can be seen as long time organizational policies while $UserDfnPol_{set}$ represents short term temporary policy set specified by a user to carry out task(s) on behalf of another user for a specified time period.

6.1 The Policy Set

The Policy Set consist of various policy rules that allows the system to functionally assign a nomadic user to a specific role. The policy engine component of the system is crux and it forms the nucleus of the entire system design and implementation. The policy engine is based on the inference rules (security, functional and non-functional rules) to make policy decision and enforcement.

6.2 Functional and NonFunctional Policy Rules

Functional rules are static users' requirement to role assignment. This is mainly based on users' bio data, academic credentials and the organization lay down rules to assign a role to a user. For instance, the conditions needed to assign a doctor to a particular role is required to define the functional rule requirement for that role. **The NonFunctional Rules** are required to implement additional responsibilities that can be assigned to a role-member (such as contextual information, schedule, delegation criteria etc.).

For example, a functional role requirement for a clerical officer in a given nomadic environment (e.g. nomadic hospital environment) can be represented as follows

Candidate must have a leaving school certificate (Rule 1)

Candidate must have a computer Application Certificate (Rule 2)

Non-functional rules are based on transaction process that can be implemented by the candidate having satisfied the Functional rule. E.g.

IF Candidate (Functional rule → True)

CHECK (Candidate contextual information)

CHECK (Schedule → ScheduleList)

If (CHECK) IS True))

ALLOW Candidate PERFORM Task(s)

STORE Transaction Trail IN Database

ERASE (Non-functional rule) FROM (Candidate policy set)

By this illustration a Non-functional rule is only implemented during task delegation and permission granting. During implementation, such privileges are included to temporary functional rule set to perform action.

A policy set is therefore a set of rules which hold for a particular role having considered its security, functional and nonfunctional requirements.

Mathematically;

To perform role assignment, we define a function NF(r) and FN(r) where NF(r) represents NonFunctional Rule for a role and FN(r) represents Functional Rule for a role. $r_1 \rightarrow r$ represents all rules contained in r. This is represented in equation 7 as follows:

$$NF(r), \forall Nu \in U, \forall\, r \in Rule, \forall r_1 : r_1 \rightarrow r \; and \; FN(r) \; such \; as \; \forall r \in tempRule, \forall r_1 : r_1 \rightarrow r \Longrightarrow can - delegate\ (Nu, Rule) \quad (7)$$

An algorithm for role assignment in the nomadic environment as contained in equation 7 is represented in algorithms 1 and 2 as follows;

Algorithm 1: PolicySet formulation ( )

START

CREATE Role Table (role-id, role name)

DEFINE Max functcount = N

SET Initial functcount = 1

again: WHILE (Functional Rule (functcount) < N)

READIN (Functional Rule (functcount))

STORE and LINK functional rule →Role Table

functcount = functcount + 1

GOTO again

ENDWHILE

END

In algorithm 1, all users are first organized into different roles. Each role defines the class of job function to be assigned some set of task(s) that can be performed. Initially, the functional rules are required to be defined. This enable the system to have a specified number of roles in the organization.

Algorithm 2: PolicySet Enrollment ( )

START

ENTER users' credentials

ENTER value for N

CALL Subprocedure (**PolicySet formulation (role-id, role name)**

FOR functional rules (1→N)

X = 1

LOOP: COMPARE (Selected Credentials WITH functional rule (X)) FROM role-table

WHILE (Selected Credentials MATCHED WITH functional rule(X))

DISPLAY ("Candidate has being successfully assigned :") role (role name, username, dept)

ELSE

DISPLAY ("Candidate does not meet policy standard, cannot be assigned by the policy engine")

ENDWHILE

IF (X < N) THEN

X = X + 1

GOTO Loop

ENDIF

ENDFOR

END

In algorithm 2, each nomadic user is assigned to a role name by first inputting the user credentials at the point of enrollment. The sub procedure call PolicySet formulation (role-id, name) is required to bring in the functional rules used to define the various role names in the organization. The user credentials is then compared with the organizational set standards on credentials that users must have before he can be assigned to a role. If the user credentials matches with the set standard for any particular role, then the user is assigned to that role name, a role-id is equally assigned to identify the user in the assigned role. The user department or unit is equally assigned. The department is required to enable us formulate concatenated key that can be used in querying the database.

6.3 Permission Granting in the Nomadic Environment

Permission Granting is the process of assigning access right and privileges to a nomadic user to carry out a specific task(s). In the design approach, we represent each role member with its role-id and represented each task with a task-id. With this representation, the nomadic user is independent to a particular role responsibilities. This makes tasks delegation in a nomadic environment implementable.

Instance

Suppose a user belong to a role which has $task_1$, $task_2$ and $task_3$. A procedure for granting permission to perform the task(s) is shown in Algorithm 3.

Algorithm 3: PermissionGranting( )

Start

INPUT User

CREATE role-id

CREATE $task_1$-id, $task_2$-id, $task_3$-id

ASSOCIATE Role-id WITH $task_1$-id, $task_2$-id, $task_3$-id

STORE in Task Table

'If a user wants to perform $task_1$ and $task_2$ only, the procedure is as follows

CALL User role-id, task-id FROM task table → role table

ASSIGN $task_1$-id, $task_2$-id to User role-id

CREATE Temptask table

STORE User, role-id, $task_1$-id, $task_2$-id, IN Temptask table.

AFTER Authentication, GRANT User access right to Temptask table

END

In Algorithm 3, all users assigned to a role. Each role is identified by the role-id which is appended to the user. The permissible task(s) are not assigned directly to user but rather to the role the user belong to. However, each role has a collection of organizational responsibilities that they can perform. This are represented as tasks. Each of these task(s) equally has a task identification number (task-id). Therefore granting permission to a user will require the user's name, role name, role id and the task-ids permissible by the role name. With this approach, tasks are not directly assigned to users but rather to roles; the users will be assigned to a role and the role has permissible task-ids already. However, during a transaction process, the task-ids required by the users is pulled into a Temptask table and the user can only be granted access to perform task(s) contained in that Temptask table. Therefore during task delegation, the delegated task-id(s) is pulled into a TempTask table and made visible to the delegatee. The delegatee can only have access to this TempTask table after proper authentication.

7. Implementation Scenario using *the Hospital environment*

Consider a nomadic hospital environment, the Role Based Access Control mechanism allows the delegated doctor to have access to all the permissions contained in the delegator's role. And in the course of delegating only specific task, the delegatee will still have access to the permissions of the delegator. These drawback associated with implementing task delegation in a secured manner is addressed by appending a policy engine as a top layer to the RBAC mechanism.

Typically, the policy engine consist of the following:

i. All policy rules of all the roles in the organization in assigning roles, granting permissions, task(s) delegations,
ii. all terminal addresses in the nomadic environment,
iii. all users credentials, all users job schedules and access to all users authentication information, etc.

E.g. a possible policy rule for a doctor can be:

A doctor can perform a task say task1 if and only if doctor has being authenticated; he is a qualified doctor; he is on duty; he is assigned to the patient and he is not busy.

To articulate this convoluted conditions, a Structured English is used to represent the implementation process logic as follows:

SET UP policy rule for task 1

SET UP user credentials, User Role Assignment Set and User Permission Set

IF authentication is true THEN proceed ELSE abort

ELSEIF doctor is qualified (CHECK doctor credentials component to assert this)

ELSEIF doctor is assigned (CHECK Role Assignment register component to assert this)

ELSEIF doctor is on duty (CHECK Permission Assignment component to assert this)

ELSEIF doctor is available (CHECK doctors Schedule Register component to assert this)

ELSE

Doctor is not permitted to handle patient case file

ENDIF

ENDIF

ENDIF

ENDIF

The policy engine is essentially a set of Control Statements; in this case a Nested If statements which based on certain axioms can carry out deductive reasoning from its inference module; the policy repository

8. Conclusion

A Policy Role-Based Access Control Model was formulated using mathematical modelling, algorithms and Structured English. A model representing the Policy Role Based Access Control Mechanism for a nomadic environment was developed. Algorithms to represent PolicySet formulation, PolicySet enrollment and permission granting in a nomadic environment was carefully represented and discussed. Task-ids was used expressively to implement user to user task delegation. A possible implementation scenario was illustrated using the hospital environment. A full system implementation of this model in any nomadic environment will enhance

role assignment and task delegation. The modelling approach is expressive enough to support a wide sphere of organizational policies.

References

[1] L. Kleinrock. "Nomadic Computing UCLA Computer Science Department". Supported by Advanced Research Projects Agency, ARPA/CSTO, J-FBI, 1996, pp. 93-112.

[2] E. Mikoczy, I. Kotuliak, and O. V. Deventer O.V. "Evolution of the Converged NGN Service Platforms Towards Future Networks". Future Internet Journal, 2011, Vol. 3, No. 1, pp. 45-55.

[3] S. Vytautas, D. Robertas, V. Jonas, Z. Giedrius, L. Virginija, and T. Eugenijus. "Generation of Database Interfaces for Nomadic Users". Informacines Technologijos ir Vadymas, 2003. Vol. 2, No. 27, pp. 1 – 10.

[4] A. Tatiana, and S. Noemie. " Service creation and self-management Mechanism for Mobile Cloud Computing., V. Tsaoussidis et al., (Eds). WWIC 2013, pp.43 - 55.

[5] A. Naveed and J. Christian. "A delegation framework for nomadic users". International Conference on Computer and Information Technology, CIT, 2009, Vol. 8, No. 11, pp. 66 -72.

[6] K. Zhang and T. Kindberg . "An Authorization infrastructure for nomadic computing", Mobile Systems and Services Laboratory, Hewlett-Packard Laboratories, USA, 2000, pp. 1- 9.

[7] P. Vartan. "Nomadic Computing with Mobile Devices". Cognizant 20-20 Insights., 2012, pp. 1 – 12.

[8]D. Ferraiolo, J. Barkley, and R. Kuhn. "A Role Based Access Control Model and Reference Implementation within a Corporate Internet. ACM Transactions on Information and System Security, 2(1),1999, pp.554-563.

[9] V. Ungureanu, F. Vesuna, and N. H. Minsky. "A Policy-Based Access Control Mechanism for the Corporate Web. 16th Annual Computer Security Applications ConferenceP2000, New Orleans, pp 1 – 9.

[10] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. "Role-Based Access Control Models," IEEE Computer 29(2), 1996, pp. 38-47. http://dx.doi.org/10.1109/2.485845.

[11] J. M. Convington and M. R. Sastry. "A Contextual Attribute-Based Access Control Model, In Meersman et al., 2006, pp. 64-74.