I-RBAC: Isolation Enabled Role-Based Access Control

Nagajyothi Gunti[#1], Weiqing Sun[*2], Mohammed Niamat[#3]

[#]*Department of Electrical Engineering and Computer Science, University of Toledo*
Toledo, Ohio, US - 43606

[1]ngunti@rockets.utoledo.edu

[3]mohammed.niamat@utoledo.edu

[*]*Department of Engineering Technology, University of Toledo*
Toledo, Ohio - 43606

[2]weiqing.sun@utoledo.edu

Abstract— **Access control is a means by which the ability to access the system is explicitly enabled or restricted in some way. Access control system enables an authority to control access to areas and resources in a given physical facility or computer-based information system. In Role-based Access Control (RBAC) model, access to resources is based on the role of the user in an organization. Previous RBAC models have encountered various problems in meeting the growing and diverse security needs of the organizations. Hence, we propose Isolation Enabled Role-based Access Control (I-RBAC) model. The basic idea of our model is to incorporate the isolation concepts into the NIST (National Institute of Standards and Technology) RBAC model such that it can be applied in a broader domain. In I-RBAC, the operation on an object by the role is executed inside isolation environment if the role or the operation is predefined to be isolated. Typical roles include inexperienced personnel (such as intern doctors) and delegatee. At the end of each session a security check for accumulated modifications will notify the user about the consistency of the environments. Evaluation of the implementation of the prototype on a Health Care System demonstrates the effectiveness of I-RBAC model. The key idea is to ensure system availability at all times for all the roles, while simultaneously ensuring the system integrity and security. Another main advantage is that, it would be a cost-effective alternative to building a separate RBAC system to enable otherwise disallowed accesses, such as the training roles.**

Keywords— **Access Control Model, Role-based Access Control, Isolation, Security Policy, I-RBAC**

I. INTRODUCTION

Access is the ability to enter into a computer resource. Access control is a means by which the ability to access is explicitly enabled or restricted in some way. An access control system enables an authority to control access to areas and resources in a given physical facility or computer-based information system [1]. Role-based Access Control (RBAC) [2] is a method of regulating access to a computer or a network's resources based on the roles of individual users within an organization. The multi-user and multi-application online-systems pioneered in 1970's introduced the concept of roles. The model enables users to carry out a wide range of authorized tasks by dynamically regulating their actions according to flexible functions, relationships, and constraints. According to Sandhu et al. [3], roles can be easily created, changed or discontinued as the needs of the organization evolve. When compared to discretionary access control and mandatory access control, RBAC is considered to be flexible and easier to manage in large scale organizations. But, it is usually a rigid model where access control decisions have only two output options i.e. Grant or Deny [4], [5].

The assumption in traditional access control models as listed in [3], [6], [7] is that the access permissions are known in advance and that the roles have been set up correctly. But, in practice new roles may be added, errors may be made and unanticipated or emergency situations may occur. A strict application of previous RBAC models will tend to be either cumbersome or error-prone. In order to meet the growing needs of the organizations, a more flexible and adaptable approach is needed. An isolation environment [8] for the execution of such cases would be a feasible solution. In this approach, after the authentication of the user, the role of the user will be identified. If the role is not authorized to perform an operation on the object, isolation policies are checked. Isolation policies define the role to be isolated or the operation by the role to be isolated. If the role is isolated, then any operation by it is done in an isolation environment. If neither of the isolation policies is satisfied then "Permission Denied" message is sent to the user. Furthermore, a security check for the accumulated modifications is done at the end of each session. This check will notify the user if there is any conflict with the modifications in the session. In our model, we assume that the isolation policies are predefined by the organization.

The main goal of this paper is to integrate an isolation environment in the NIST RBAC model in a transparent and secure way so that it can be applied in a broader domain. Incorporating this idea in RBAC would add usability and efficiency to it. We name the extended model as Isolation Enabled Role-based Access Control (I-RBAC) Model.

The rest of the paper is organized as follows. Section II describes the NIST Role-based Access Control model. Section III briefly gives an overview of the related work. Section IV describes the concept of I-RBAC model along with its formal definition, implementation in a part of health care system followed by other applications. Section V gives a general

discussion and limitations of I-RBAC Model. Section VI concludes our work.

II. The NIST RBAC Model

Starting in 1970's, computer systems featured multiple applications and served multiple users, leading to heightened awareness of data security issues. System administrators and software developers focused on different kinds of access control to ensure that only authorized users were given access to certain data or resources. Role-based Access Control (RBAC) model was the one that emerged to serve the purpose.

A role is a semantic construct forming the basis of access control policy [3]. It can represent a specific task competency, such as that of a physician or a pharmacist. Roles define the specific individuals allowed to access the resources and the extent to which resources are accessed. The ability or need to access information may depend on one's job functions. This suggests associating access with the particular job of a user.

The NIST model uses a limited set of concepts to define an RBAC system. The system has users, users have sessions and sessions and users have roles assigned to them. Each role consists of permissions and the permissions are based on the objects and operations. The details of the concepts are listed as follows:

1) User is an entity wishing to access a data resource or an object. Users are both employees and network mechanisms and entities that require access to a specific resource or an object.
2) When a user logs in all the assigned roles are associated with the session. A session allows for temporary changes to the role structure, i.e., take away a role for the duration of the session or add a role for the duration of the session.
3) A role is a package of permissions based on a job function within the organization. A user can be associated with more than one role depending on the needs of their position.
4) Permissions are assigned to a role and grant access to the operations.
5) Objects are accessed through operations that a user has a permission to access through the role they are assigned.
6) Operations are lower than permissions in the RBAC element hierarchy and usually are very specific functions.

A formal definition for the model is described using the concepts USERS, ROLES, PRMS (Permissions), OPS (Operations), and OBS (Objects) as shown below:

- UA: User Assignment $\subseteq$ USERS x ROLES
- PA: Permission Assignment $\subseteq$ PRMS x ROLES
- U-S: user_sessions (u: USERS) $\rightarrow 2^{SESSIONS}$
- S-R: session_roles (s: SESSIONS) $\rightarrow 2^{ROLES}$
- PRMS: $2^{(OPS \times OBS)}$, the set of permissions

Op: (p $\in$ PRMS) $\rightarrow$ {op $\subseteq$ OPS}, the permission-to-operation mapping, which gives the set of operations associated with permission *p*;

Ob: (p $\in$ PRMS)$\rightarrow$ {op $\subseteq$ OBS}, the permission-to-object mapping, which gives the set of objects associated with permission *p*.

CheckAccess is the authorization decision making function which describes how a decision is made within NIST RBAC model by taking as inputs the current session, the requested operation and the target object. This function returns a Boolean value i.e. TRUE or FALSE.

CheckAccess: SESSIONS x OPS x OBS $\rightarrow$ BOOLEAN Value

CheckAccess(s,op,ob) = ($\exists$ r $\in$ ROLES: r $\in$ S-R(s) Λ ((op,ob),r) $\in$ PA)

The *CheckAccess* function checks if a role *r* can be mapped for the current session *s*, such that *r* has been allocated the permission to perform the operations *op* on the objects *ob*. If such a value exits, the function returns TRUE, otherwise FALSE will be returned.

Fig. 1 shows the steps followed by the process of an RBAC model. In this a user requests for the resource from the application service. The authentication service validates the user's credentials. It then returns the authenticated identity of the user and if it is an unauthorized user then a deny message is sent to the user and the request is terminated without giving access. If it is an authorized user then the application service calls for the policy check giving the user's role and its permissions as parameters. The RBAC policy check will return either a TRUE or FALSE to the application service. In case it returns FALSE then the request is terminated here. Otherwise, the user is allowed to perform the requested operation on the object. The operation returns the results to the application service which in turn returns the results to the user. Here, application service can be a user interface and an object can be a data file or a hardware resource.

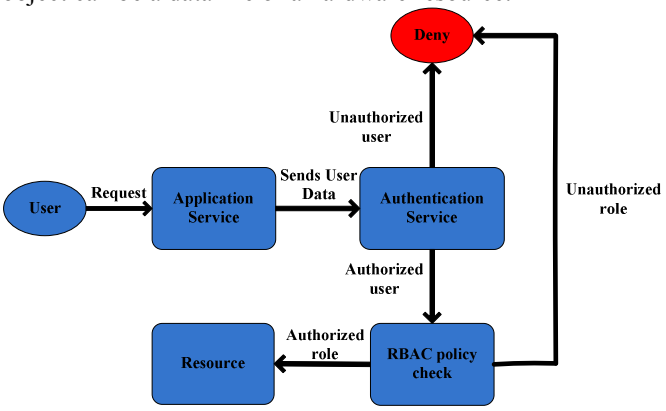

Fig. 1 Process in NIST RBAC System

A study by the National Institute of Standards and Technology (NIST) demonstrates that RBAC addresses many different needs in commercial and government sectors [9]. Access control requirements were found to be determined by the need for customer, stakeholder, and insurer confidence; personal information privacy; the prevention of unauthorized financial asset distribution and unauthorized long-distance telephone calls; and an adherence to professional standards.

Also, role-permission relationships can be predefined, which makes it simple to assign the users to the predefined roles. RBAC model supports three well-known security principles:

1) Least privilege: means only those permissions required for the tasks performed by the user in the role are assigned to the role.
2) Separation of duties: invocation of the mutually exclusive roles can be required to complete a sensitive task, such as requiring an accounting clerk and an account manager to participate in issuing a check.
3) Data abstraction: instead of read, write and execute permissions typically provided by the operating system, abstract permissions such as credit and debit for an account object can be established.

RBAC is used in many organizations because it offers advantages like expressing policies especially suited for commercial applications. It can easily mirror an organization's structure and encourage well-structured access control policies. Also, RBAC provides a powerful mechanism for reducing the complexity, cost and potential errors of assigned user permissions within the organizations.

III. Related Work

Even though RBAC has reached a good maturity level, it is recognized that there are still significant application requirements not addressed by the current RBAC models. Due to the growing and changing needs of the organizations, past and current research in the area of RBAC models has lead to many enhancements and extensions.

In many organizations, functions may have limited or periodic temporal duration. For instance, part-time staff is authorized to work within the organization only on working days between 9:00AM to 1:00PM. If part-time staff is represented by a role, then the above requirement entails that this role should be enabled only during the temporal intervals. Also, in a framework where the roles are not always enabled it is important to introduce ways to specify the dependencies among the enabling and disabling of the different roles. In this case, we can consider a doctor-on-night duty whose role is enabled during the night. Since doctors may need the assistance of a nurse, one should make sure that the corresponding role, a nurse-on-night-duty is also enabled whenever a doctor-on-night-duty is present. RBAC was extended to support these requirements and the model was named as Temporal Role-based Access Control (TRBAC) model [10]. TRBAC supports role enabling/disabling and the temporal dependencies among such actions. But this model allows the security officer to specify the role trigger requesting the enabling/disabling of roles on the basis of the occurrence of specific events i.e. enabling/disabling of other roles and/or verification of some conditions. However, there are many situations in which such a finer granularity level would be useful. For instance, a nurse on training may perform some potentially dangerous actions. In this case, we need to have a mechanism which allows the security officer to disable the role nurse-on-training for that particular nurse only and not for all the other nurses who are actually authorized to activate this role. In order to formulate the individual exceptions to role enabling and disabling, the specification language for the TRBAC is extended for individual disabling and re-enabling.

Another related work has been carried out in the context of mobile systems. Frode Hansen and Vladimir Oleshchuk [11] have proposed that the system should be able to base its access decisions depending on the spatial dimension in which the user is situated. The reason for this is that mobile computing devices and wireless networks are increasingly being utilized by organizations. This enables users to gain access to networked computer resources anywhere and anytime, through their mobile terminal. In organizations where access to resources is limited to a specific location, location-based services require means for obtaining the position of the requesting user in order to mediate the authorization request. For instance, the doctor has the permission to access a patient's record. However, due to the sensitive information that this record contains, the doctor is only authorized to access it in designated areas. If the doctor requests to access a patient's record from a less trustworthy location like cafeteria or reception where there can be considerable accumulation of people, the doctor's access request is denied. In order to meet spatial requirements Spatial Role-based Access Control (SRBAC) model was proposed to be able to specify spatial constraints on the enabling and disabling of roles. In real-time environment there would be many complications. All the techniques for finding the location of the mobile terminal use either radio signals or infrared rays. These signals can be blocked or might fail to identify the exact location of the terminal. This might be a hindrance to the role which needs to access a resource from a remote server. Also, there might be an error with the location finding system and may return a wrong location.

An access control model that allows temporal specification as well as spatial specification is proposed by Indrakshi Ray and Manachai Toahchoodee [12]. Spatio-temporal Role-based Access Control model is able to make access decisions based on the spatio-temporal information. With increase in the growth of wireless networks, sensor and mobile devices, we are moving towards an era of pervasive computing where spatio-temporal information will be necessary for the access control. By using this information we can enhance the security of an application and it can also be exploited to launch attacks. Traditional access control models [1], [6] do not take into account environmental factors before making decisions. The model is formalized by enumerating the constraints. This model can be used for applications where spatial and temporal information of a subject and an object must be taken into account before granting and denying the access. An example to support this model is a loan officer in a bank who can read and write loan files only from the loan office during working hours, that is, 9:00AM to 6:00PM, Monday to Friday [13]. This model considers the impact of time and location on role hierarchy and separation of duty.

Another extension of RBAC model introduced by Ana Ferreira et al. was 'Break The Glass' (BTG) [4]. RBAC is a

rigid model where there are only two options for the access control decision: Grant or Deny. The BTG concept will allow the users to break or override the access controls in a controlled and justifiable manner. It is integrated into RBAC model in such a way that it can be adopted generically in any domain where unanticipated and emergency situations may occur. This provides a third option where BTG grants the authorized users permission to break the glass rather than to deny the access. The BTG policy can be used to break or override the access controls in a controlled manner by a role that is not authorized to access a resource or an object.

IV. I-RBAC

This section provides a description of our model, which extends NIST RBAC model [3], [9] by adding an Isolation Environment. This is done by allowing for an option, Isolation instead of Deny in the NIST RBAC model. To be able to accomplish this, we provide a formal definition of I-RBAC and a process flowchart followed by the implementation of a prototype in a health care system. We also discuss other applications of the I-RBAC model.

A. Formal Definition

Isolation is an action or condition of setting something apart from other things. The concept of isolation provides a way to address the problem of effect containment for compromised processes in [14]. The isolation environment is a contained environment where exceptions can run without affecting the host environment. Isolated execution allows handling a pool of such contained environments for running the requests of the unauthorized roles. This feature decreases the likelihood of human error initiating an attack and limits the damage caused by the unauthorized roles [8], [15].

In general, access control models assume that the access permissions are known in advance and rules have been set up correctly. We offer an isolation strategy that is more flexible and adaptable. We name our model as Isolation Enabled Role-Based Access Control (I-RBAC). For the formal definition of I-RBAC model, we add the following elements to the NIST RBAC definition:

- Isolated Role (IR) is a package of permissions which could be executed in the isolation environment. These roles are assigned to user who has denied access to the host environment. For instance, an intern doctor in a health care system and a delegatee in case of delegation.
- Isolation value (I) is either true or false. Only the operation by the role whose isolation value is true is allowed access to the isolation environment.
- Isolation Policies (I-POLICIES) are conditions that are to be satisfied by the role to enter into the isolation environment. The condition can be either the role to be isolated or a particular operation of the role to be isolated.

In this new model we assume that IR, I and I-POLICIES are predefined based on the requirements of the organization. The operation on the object is executed in isolation if either of the following two conditions is satisfied:

- If role belongs to a set of Isolated Roles ($r \in IR$), then all the operations by that role would be executed in isolation environment, or
- If Isolation value (I) for the operation requested by the role is true ($(r,op) \in I$)), then the operation by the role is executed in isolation environment.

In Section II, we described the formal definition for NIST RBAC model. The *CheckAccess* function would either return TRUE or FALSE. If the *CheckAccess* function returns FALSE, then "Permission Denied" message is sent to the user, else the permission is granted and the requested operation is executed in the host environment. In our model, instead of denying access to the user, we check for the above two cases. If any of the two cases is satisfied, the user is allowed to execute his/her request in the isolation environment as if it were in the host environment.

For this, we introduce *CheckIsolation* function. *CheckIsolation* is an isolation decision making function. It describes how the decision for isolation is made within I-RBAC by taking the role, operation, and object as inputs. This function also returns a Boolean value (TRUE or FALSE). Here, we modify the consequences of the results for the *CheckAccess* function. The formal definition is as follows:

CheckAccess: SESSIONS x OPS x OBS $\rightarrow$ BOOLEAN Value

$$\textit{CheckAccess}(s,op,ob) = (\exists\, r \in \text{ROLES}: r \in \text{S-R}(s) \wedge ((op,ob),r) \in PA)$$

{True} $\rightarrow$ Allow access in the host environment.

{False} $\rightarrow$ *CheckIsolation()* instead of Deny.

Where,

CheckIsolation: ROLES x OPS x OBS $\rightarrow$ BOOLEAN Value

$$\textit{CheckIsolation}(r,op,ob) = (\exists\,((r \in IR) \vee (r,op) \in I)))$$

The *CheckIsolation* function checks if a role *r* has been defined as an isolated role or has been allocated the permission to perform the operation *op* on object *ob* in the isolation environment. If such a value exists, the function returns True, otherwise False.

At the end of the user session in the isolation environment, a security check on the accumulated modifications is performed. If any of the security check conditions fail to satisfy, the system continues with another user session after notifying the user. Otherwise, the system simply continues with another user session. The security check condition may be related to consistency of the object or the permissions of the role. For instance, a security check that ensures that there should not be two patient records with the same name is related to consistency condition. A situation where the person who created the record can only delete it is related to permissions of the role in the security check.

The process flowchart for the I-RBAC model is shown in Fig. 2. *Application Service* provides an interface through which the user can login into the host environment. After logging he/she can request for an access to the object. The *Authentication Service* validates the user's credentials. If the user is not authorized he/she is denied access to enter the

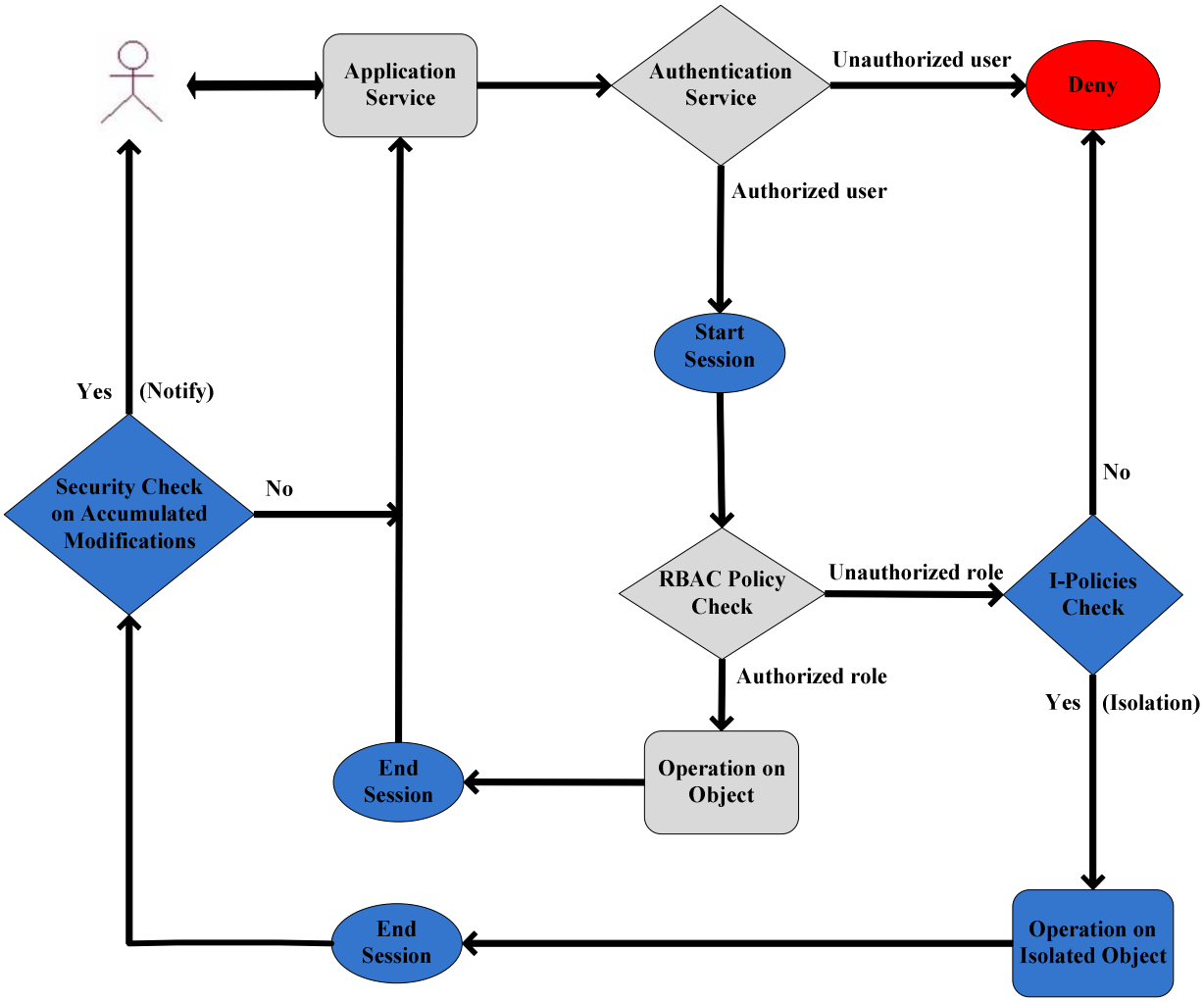

Fig. 2 Process Flow-chart of I-RBAC Model

system else we *Start Session*. The *RBAC Policy Check* will identify the roles of the user in that session. If the user has a role that is authorized to access the object, operation on the object is performed in the host environment. Else, we pass the role, operation, object inputs to the *I-Policies check*. If any of the conditions satisfies then the access for operation on the isolated object is granted. At the end of the session a *Security Check on Accumulated Modifications* will notify the user with any conflicts in consistency at the end of the user session. If the I-Policies do not satisfy, we Deny the access to the user.

B. Implementation of I-RBAC in Health Care System

There are many situations in the hospitals which would not be supported by the earlier developed RBAC models. For example, consider the case where the doctors in training and internships cannot be given complete access to resources because of the high probability of making mistakes. The doctors in internship or training might commit errors in writing the prescription to patients which would be a loss to the patient as well as the reputation of the hospital. And using a separate hospital system for them would be expensive. A doctor has read/write access to a prescription file. It is sometimes necessary for the pharmacist to have a write access when the doctor has written a wrong date on the prescription file or correcting the spelling of the medicine. So, the pharmacist must be able to edit the prescription file with limitations (like correcting or updating the date or medicine).

In order to better understand this model we would present an overview of how it would be implemented in part of the health care system considering the five elements of the RBAC model listed in Table 1.

The elements added to I-RBAC model are:

- Set of Isolated Roles i.e. IR = {Intern Doctor}.
- The operations by roles whose isolation value is true are defined in Table 2.
- Security check conditions for the accumulated modifications at the end of user session are described in Table 3.

In case of an authorized doctor, it is not needed that he be put to isolation for any of his requests because he is an authorized role to view, create, edit and delete an EPR or PF. If we consider the case of an intern doctor, this role is categorized as an isolated role. So, irrespective of the request for operation on the object by the role, the execution is passed to the isolation environment. If an intern doctor requests for creating

TABLE I
FIVE ELEMENTS OF RBAC IN A PART OF HEALTH CARE SYSTEM

Users	Roles	Permissions/Operations	Objects
Any person	Security officer Doctors Intern doctors Pharmacist	View EPR Create EPR Edit EPR Delete EPR View PF Create PF Edit PF Delete PF	EPR* PF*

*EPR means Electronic Patient Record and PF means Prescription File

TABLE II
DEFINITION OF ISOLATION VALUES

Roles	Permissions/Operations
Pharmacist	Edit PF View EPR Edit EPR

TABLE III
DEFINITION OF SECURITY CHECK FOR ACCUMULATED MODIFICATIONS

Roles	Operation	Object	Security Check – Accumulated Modifications
Intern Doctor	Create	EPR	No two EPRs should be with same patient name (EPRname1≠ EPRname2).
Intern Doctor	Edit	EPR	The EPR which is edited should be consistent.
Intern Doctor	Delete	EPR/PF	EPR should be created by the same user who is trying to delete it.
Intern Doctor	Create	PF	A patient should have only one PF on a particular date.
Intern Doctor	Edit	PF	Medicine should be within the predefined prescription database.
Pharmacist	Edit	PF	Medicine should be within the predefined prescription database.
Pharmacist	Edit	EPR	Only Date, Gender and Patient's name can be edited.

an EPR he is allowed to do it in isolation environment as if he were in the host environment. At the end of his/her session, a check is made whether there is an EPR with the same patient name. If there is one, then a notification is sent to the user about the conflict, else we simply continue with another user session. If the request is to edit an EPR, we fetch the record to be edited from the host environment to the isolation environment. The changes to the EPR can be made as if it were in the host environment. If the record is consistent, then at the end of the session we continue with another user session, else notify the user before continuing with another user session. Finally, if the request is to delete an EPR, we allow the deletion in isolation environment. Here, the security check is that the record can be deleted only by the person who created it. In all the above cases, if the I-Policies are violated, a "Permission Denied" message is sent to the user. Similar is the case with creation, editing and deletion of the PF.

A pharmacist can view PF and EPR as per the RBAC policies but will not have access rights to either edit a PF or an EPR. In our model, we allow this case through isolation strategy. When the pharmacist requests to edit a PF, he can be allowed to do it in the isolation environment. At the end of the session we notify him/her with the security check. We refer to Table 3 for the security check conditions for accumulated modifications.

We used Microsoft .Net Framework 2.0 and C# programming language for implementation of the prototype. The prototype has been implemented in Windows Vista Operating System. Microsoft .Net Framework 2.0 with Active Solution Platform set to x86 is the runtime environment required for the executable to be run. MS Office Access 2007 is the database used for storing the EPRs and PFs. We used the IR, I and security conditions specified in Table 1, Table 2 and Table 3 respectively for the prototype. During the test, we performed the authentication and the security tests and observed that the prototype is running according to the specifications. After thorough analysis and testing, the prototype proved to yield positive results with regards to usability and security. Also, this is a generic solution because it can be implemented using any kind of programming language and database. In the future, we plan to incorporate the commit operation by considering all the consistency criteria with respect to the host environment and the isolation environment.

The prototype which we implemented cannot be used for a practical management of Health Care System. So, we are currently working on open source software named OpenEMR [34]. OpenEMR provides a set of pre-defined roles and permissions upon installation. Administrators are given the ability to add any arbitrarily named roles and assign any number of privileges. The present system has a common set of health care provider roles: Administrators, Accounting, Clinicians, Physicians and Front Office. Our idea is to include Intern Doctor also as a role and assign the privileges that meet

our model requirements. This would give a more practical approach to our idea which is not completely provided by our prototype.

C. Other Applications of I-RBAC

When the organization policy is poorly documented or not adhered to, normal RBAC can be more of a hindrance than a blessing. This is because the organizational policy or layout defines a model. It is assumed that as the organization grows it will be in logically constructed manner. Thus our security model can adapt and adhere to the changing organizational policy that the model is based upon. Failure to follow detailed organizational policy will introduce "dead ends" that limit the model's scalability. Using our model we can gradually learn what role the user should be assigned to and what permissions the user should have through the isolated execution. Here, we first isolate the user in question and later the behaviours of the user. In this case, the isolation value for the operation by such roles can be defined to be true or all the users which are not assigned any role can be categorized into a set of isolated roles.

In temporal, spatial and spatio-temporal extensions to the RBAC model, there are situations where a single role has to perform two tasks at the same time. For instance, we require opening a single bank account on two systems at the same time or approving two time cards of the same duration at the same time. Moreover, a part-time worker may need to access an object when he/she is not on duty timings. In case of the spatial RBAC there are cases where a person at one location might want to access an object for which he has permission when he/she is at another location. In addition, the tools used for locating the spatial information (such as GPS, Active Badge Location system and 3D-iD system) might show errors due to environmental conditions. All these would restrict the authorized role from accessing the object. However, using I-RBAC would avoid these problems by executing the request of role in the isolation environment.

Finally, roles which do not have experience such as newly joined employees might make errors which result in loss to the organization. This situation can be handled by isolating the role accessing an object until it reaches a defined threshold experience value. The experience value can be a natural number which is incremented each time a role performs the same task successfully. In other words, all the roles which have the experience value less than the threshold value should be categorized as the isolated roles or the isolation value should be assigned based on the experience value.

All these applications would be implemented in a similar manner as in the health care system discussed above. For each of the applications, there would be different set of isolation policies in addition to the users, roles, permissions, operations, objects and security conditions.

V. DISCUSSION

Earlier enhancements to RBAC models such as Break The Glass (BTG) [4] share some similarity with our approach. But in general, they fail to address situations where there is a high possibility that certain roles such as the doctors under internship might make mistakes. It would be expensive and time consuming if we were to build a separate RBAC system for such interns to enable them to access the system. Instead, we can use our proposed model where the role is identified as intern doctor or doctor under training. Therefore, operations from the role will be allowed to run inside an isolation environment. In this case, intern doctor is categorized as an isolated role.

Also, delegation allows temporary grant or transfer of access rights [16]. Research in this direction provides one to delegate the permissions based on three factors – properties, experiences, and recommendations. The choice of a delegatee is determined by the trustworthiness value and the security policy. The trustworthiness value is calculated from the three factors mentioned above. All these factors are varying and would be different from the perspective of each role. Therefore, the trustworthiness value would also be varying at all the times. This approach would introduce an overhead for calculating the trustworthiness values before the role can be delegated. Instead, our model would simply isolate the delegated role to avoid the calculation of trustworthiness value by getting details from other roles about the delegatee.

The proposed model has loosened the security implementation by allowing greater access to users than what they are authorized to. Therefore, security officer must have intimate knowledge of how permissions are being granted, why and what operations are associated with those permissions and roles. Failure to properly communicate with the roles about the policies in our model would make it insecure. So, a company is only as good as its people i.e. I-RBAC security implementation is only as good as the staff supporting it. Also, complete care needs to be taken in defining the policies of when to perform operation on object in isolation. This requires a thorough study of the requirements of the organization. I-RBAC may cause consistency problems when committing the operations to the host environment.

On the other hand, our model can be directly implemented using the already existing RBAC system. We simply need to incorporate isolation execution functionality and define the I-POLICIES for the application. The implementation of our model would not require any duplication of existing system or beginning from the scratch. This would greatly reduce the investment in implementing our model. Also, this model would help in ensuring system availability at all the times for all the users, simultaneously ensuring the system integrity and security. The main advantage would be saving time and money in building a separate RBAC system for training of roles or access decisions based on time and location.

We can also consider Task Isolation (TI) as an extension to our I-RBAC model. TI can be defined as, irrespective of the role if the operation belongs to a set of isolated tasks, then it will be executed inside isolation. This kind of isolation requires the predefinition of Isolated Tasks (IT) and a complete study of such operations in an organization. Also,

the commit operation of the changes in isolation environment to the host environment is not considered in our present model.

VI. Conclusion

Access control models have been extended since the initial implementations of Mandatory Access Control (MAC) and Discretionary Access Control (DAC). Researchers have been considering various complexities in RBAC for innovating extensions. In this paper, we proposed I-RBAC, an extension to the basic RBAC model in an effort to address the usability issues of RBAC under various conditions. This model integrates the isolated execution concept into the NIST RBAC model in a secure and reliable way. In particular, the request to access an object is put into isolation in any of the exceptional situations-poor documentation of RBAC policies, doctors under training or internship, lack of experience and failing in attempt to get the correct temporal and spatial information. It can be implemented within any application and it provides a more flexible, dynamic and adaptable access control model.

References

[1] "An Introduction to Role-based Access Control," *NIST/ITL Bulletin,* 1995.

[2] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Trans. on Information and System Security (TISSEC),* pp. 224-274, Aug. 2001.

[3] R. S. Sandhu, E. J. Coyne and H. L. Feinstein, C. E. Youman, "Role-Based Access Control Models," in *Proc. IEEE Computer Security,* vol. 29, no.2, pp. 38-47, Feb. 1996.

[4] A. Ferreira, D. Chadwick, P. Farinha, R. Correia, Z. Gansen, R. Chilro, and L. Antunes, "How to Securely Break into RBAC: The BTG-RBAC Model," in *Proc. Annual Computer Security Applications Conference,* pp. 23-31, Dec. 7-11, 2009.

[5] A. D. Brucker and H. Petritsch, "Extending Access Control Models with Break-glass," in *Proc. Twelfth ACM Symposium on Access Control Models and Technologies,* June 2009.

[6] S. Osborn, R. Sandhu and Q. Munawer, "Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies," *ACM Trans. on Information and System Security,* vol. 3, no. 2, pp. 85-106, May 2000.

[7] S. Dranger, R. H. Sloan and J. A. Solworth, "The Complexity of Discretionary Access Control," in *Proc. First Int. Workshop on Security (IWSEC 06),* pp. 405-420, 2006.

[8] Z. Liang, W. Sun, V. N. Venkatakrishnan and R. Sekar, "Alcatraz: An Isolated Environment for Experimenting with Untrusted Software," *ACM Trans. on Information and System Security (TISSEC),* vol. 12, Jan. 2009.

[9] H. A. Weber, "Role-Based Access Control: The NIST Solution," in *SANS institute InfoSec Reading Room,* Oct. 2003.

[10] E. Bertino, P. A. Bonatti and E. Ferrari, "TRBAC: A Temporal Role-Based Access Control Model," in *Proc. Fifth ACM Workshop on Role-Based Access Control,* pp. 21-30, 2000.

[11] F. Hansen and V. Oleshchuk, "Spatial Role-Based Access Control Model for Wireless Networks," in *Proc. Fifty Eighth IEEE Vehicular Technology Conference (VTC' 03),* vol. 3, pp. 2093-2097, Oct. 6-9, 2003.

[12] I. Ray and M. Toahchoodee, "A Spatio-Temporal Role-Based Access Control Model," in *Proc. Twenty First Annual IFIP WG 11.3 Working Conference on Data and Application Security,* pp. 211-226, July 2007.

[13] S. M. Chandran and J. B. D. Joshi, "LoT-RBAC: A Location and Time-based RBAC Model," in *Proc. Sixth Int. Conference on Web Information Systems Engineering,* vol. 1995, pp. 361-375, 2005.

[14] P. Liu, S. Jajodia and C. D. McCollum, "Intrusion confinement by isolation in information systems," in *Proc. IFIP Workshop on Database Security,* pp. 3-18, 1999.

[15] S. Jajodia, P. Liu and C. D. McCollum, "Application-level isolation to cope with malicious database users," in *Proc. Fourteenth Annual Computer Security Applications Conference,* pp. 73-82, Dec. 7-11, 1998.

[16] M. Toahchoodee, X. Xie and I. Ray, "Towards Trustworthy Delegation in Role-Based Access Control Model," in *Proc. Twelfth Int. Conference on Information Security,* pp. 379-394, Sept. 2009.

[17] M. Bishop, *Introduction to Computer Security.* Prentice Hall PTR, 2004.

[18] D. Kim, I. Ray, R. France and N. Li, "Modeling Role-Based Access Control using Parameterized UML Models," in *Proc. Fundamental Approaches to Software Engineering (FASE/ETAPS),* vol. 2984, pp. 180-193, 2004.

[19] G. Zhao, D. Chadwick and S. Otenko, "Obligation for Role Based Access Control," in *Proc. IEEE Symposium on Security in Networks and Distributed Systems (SSNDS07),*vol. 1, pp. 424-431, May 21-23, 2007.

[20] G. Booch, J. Rumbaugh and I. Jacobson, *The Unified Modeling Language.* Addison Wesley Longman, 1999.

[21] C. N. Zhang and C. Yang, "Information Flow Analysis on Role-Based Access Control Model," in *Proc. Information Management and Computer Security,* vol. 10, no. 5, pp. 225-236, 2002.

[22] S. Oh and S. Park, "Task Role-Based Access Control Model," in *Proc. Information Systems,* vol. 28, no. 6, pp. 533-562, Sept. 2003.

[23] A. P. Maranda, G. Goncalves and F. Hemery, "Representation of Extended RBAC Model using UML Language," *ACM Trans. On Information and System Security (TISSEC),* pp. 413-417, 2005.

[24] Y. A. Liu and S. D. Stoller, "Role-based access control: A corrected and simplified specification," SUNY Stony Brook Univ., Comp. Sci. Dept., Technical Report DAR 05-24, Dec. 2005.

[25] M. Hitchens and V. Varadharajan, "Issues in the Design of a Language for Role Based Access Control," in *Proc. ICICS,* pp. 22-38, 1999.

[26] C. Yang and C. N. Zhang, "A Privacy Enhanced Role-Based Access Control Model for Enterprises," in *Proc. ICCNMC,* vol. 3619, pp. 1012-1021, 2005.

[27] J. Jürjens, "UMLsec: Extending UML for Secure Systems Development," in *Proc. Fifth International Conference on the Unified Modeling Language,* pp. 412-425, Oct. 2002.

[28] P. Epstein and R. Sandhu, "Towards A UML Based Approach to Role Engineering," in *Proc. Fourth ACM Workshop on Role-based Access Control,* pp. 135-143, 1999.

[29] A. D. Brucker, H. Petritsch and S. G. Weber, "Attribute-based Encryption with Break-glass," in *Proc. Information Security Theory and Practice (WISTP),* pp. 237-244, Springer Verlag, 2010.

[30] A. Ferreira, R. Cruz, L. Antunes, P. Farinha, E. Oliveira, D. W. Chadwick and A. Costa, "How to break access control in a controlled manner," in *Proc. Nineteenth IEEE Symposium on Computer-based medical Systems (CBMS' 06),* pp. 847-854, June 2006.

[31] J. Künzi, P. Koster and M. Petković, "Emergency Access to Protected Health Records," *Medical Informatics in a United and Healthy Europe,* 2009.

[32] D. Povey, "Optimistic Security: A New Access Control Paradigm," in *Proc. 1999 Workshop on New Security Paradigms,* pp. 40-45, 2000.

[33] S. Zafar, K. Winter, R. Colvin and R. G. Dromey, "Verification of Integrated Role-Based Access Control Model," in *Proc. First Asian Working Conference on Verified Software,* pp. 230-241, 2006.

[34] E. Helms and L. Williams, "Evaluating Access Control of Open Source Electronic Health Record Systems," in *Proc. Thirty Third International Conference on Software Engineering*, May 21-28, 2011