A Framework for Multiple Authorization Types in a Healthcare Application System

Ramaswamy Chandramouli
Computer Security Division, ITL
NIST, Gaithersburg, MD 20899
(mouli@nist.gov)

Abstract

In most of the current authorization frameworks in application systems, the authorization for a user operation is determined using a static database like ACL entries or system tables. These frameworks cannot provide the foundation for supporting multiple types of authorizations like Emergency Authorizations, Context-based Authorizations etc, which are required in many vertical market systems like healthcare application systems. In this paper we describe a dynamic authorization framework which supports multiple authorization types. We use the acronym DAFMAT (Dynamic Authorization Framework for Multiple Authorization Types) to refer to this framework. The DAFMAT framework uses a combination of Role-based Access Control (RBAC) and Dynamic Type Enforcement (DTE) augmented with a logic-driven authorization engine. The application of DAFMAT for evaluating and determining various types of authorization requests for the Admissions, Discharge and Transfer System (ADT) in a healthcare enterprise is described.

1. Introduction

The need to support sophisticated authorization policies has grown tremendously in the last few years for many vertical market applications. For example, healthcare application systems dealing with patient-identifiable information will shortly be required to comply with requirements in the HIPAA Security Standards [9]. These standards stipulate that healthcare application systems should have features for user-based, role-based and context-based authorizations as well as capabilities for making emergency authorizations.

In most of the current authorization frameworks in application systems, the authorization for a user operation is determined using a static database like ACL entries or system tables. These frameworks cannot provide the foundation for supporting multiple types of authorizations like Emergency Authorizations, Context-based Authorizations etc, as required by HIPAA security standards. An authorization framework that can provide this critical functionality is proposed in this paper. The framework is based on a combination of Role-based Access Control (RBAC) and Domain Type Enforcement (DTE) access control models augmented with a logic-driven authorization engine. We have used the acronym DAFMAT (Dynamic Authorization Framework for Multiple Authorization Types) to refer to this framework. The application of DAFMAT to derive various types of authorizations for an important class of healthcare application system called the Admissions, Discharge and Transfer System (ADT) is also illustrated in this paper.

The organization of the rest of the paper is as follows: We start off by giving an overview of the DAFMAT framework by outlining its salient features in section 2. A comparison of the approach adopted in this paper to some related approaches in the area of authorization frameworks is discussed in section 3. A detailed description of the development of DAFMAT framework is given in section 4. The application of the DAFMAT framework for the ADT system is described in section 5. The assurance measures needed for DAFMAT administration are given in section 6. Section 7 presents conclusions and scope for future work.

2. Underlying Concepts in DAFMAT Framework

The DAFMAT authorization framework consists of the following components:

(a) An Hybrid Access Control Model and
(b) A logic-driven authorization engine.

The Access Control Model in DAFMAT is a combination of RBAC and DTE while the logic-driven authorization engine is first order predicate logic-based. The justifications for using a combined RBAC-DTE access control model and an augmenting logical inference engine are given in the following sections.

RBAC is a higher-level access control model that uses the abstraction concept of roles to reduce the complexity of an authorization management scheme [3]. The most

important constructs are users, roles and permissions and the relations involving these constructs. In RBAC users are assigned to roles and permissions are assigned to roles. Users derive all their permissions by virtue of their role memberships. A single user can be assigned to multiple roles and a single role can be assigned to multiple users. In addition we can also define structures for organizing roles found within an enterprise (e.g., a hierarchical structure).

RBAC being a high-level model requires intermediate structures to implement its abstraction concepts on lower level access control mechanisms on a platform (e.g., permission bits in Unix). A candidate for such a structure is a lower level mandatory access control mechanism that predates RBAC and is called the Domain and Type Enforcement (DTE) model [2]. In DTE subjects (or transaction programs) are assigned "Domain" labels and objects are assigned "Type" labels. Associated with each Domain-Type pair is a set of allowable access modes. The data structure that gives the access modes for all Domain-Type pairs is called the Domain-Type Access Matrix. The operations available to a subject are thus constrained by the domain to which it is assigned.

Combining DTE with RBAC lends a structure to the universe of permissions on a platform by providing ways of organizing subjects, objects and operations that are its constituent components. More specifically the definition of a domain in a DTE model reflects the semantics of processes that are relevant to the platform. Hence in the DTE model for a Unix operating system the daemons, file systems and administration utilities form domains. For application systems, a domain can be defined to represent a specific business process. Since the role in an RBAC model represents an organizational job function, one or more roles can be assigned to a domain based on policy concepts that have gone into the definition of the role. Now the business processes are carried out using chunks of executable code called transactions. These transactions are embedded within programs called subjects that can be invoked by a user. Hence it follows that the right to invoke a subject could be provided to multiple RBAC roles although the semantics of execution of transactions within a subject depends upon the role that operates the subject. In other words, the behavior of a subject depends upon the role from which it is invoked. Typical examples of such subjects are Oracle Stored Procedures and Oracle Dictionary Access Routines.

From the above discussions it should be clear that since both the execution logic for the subjects and the domain assignment are tied to the semantics of the role, these twin relationships constrain the assignment of subjects to domains. It is these tightly coupled constraints that help in the realization of policy concepts which have gone into the role definition by helping prevent arbitrary assignments of subjects to roles which could potentially defeat the purpose of defining the role in the first place. An implementation of RBAC on a type-enforced operating system called LOCK6 has been well described by Hoffman [5].

So far we have discussed only the hybrid access control model of the DAFMAT framework. The second component, the logical-driven authorization engine, has the following two functions:

(a) From the user action (e.g., choosing a menu option from an application), formulate the authorization request predicate using some session-related functions. Based on the value of the priority code and the context variable in the authorization request predicate, the request is designated as one of the three types of authorization (Normal, Emergency and Context-based).
(b) Using the validation conditions for each authorization type, determine whether the current authorization request is valid.

If the current request has been determined to be valid by the logical-driven authorization engine, then the appropriate domain corresponding to the invoked subject (as given in the subject-domain mapping table) is assigned to the user session. Based on the activated domain in the user session, the actual object-level permissions are read off from the Domain-Type Access Matrix.

3. Comparison with Related Work

Combining RBAC with DTE was first illustrated by Hoffman [5]. Making use of the fact that RBAC provides good higher-level abstraction mechanisms for expressing different types of policies ([8], [1]) like the Principle of Least Privilege and Conflict of Interest etc, this work illustrated a way of implementing those policies using the control mechanisms of DTE on a secure operating system. However Hoffman's implementation is based on static associations between users, roles, subjects and domains and did not provide a mechanism for incorporating transient information since it is not relevant from an operating system perspective. Tidswell and Potter [10] illustrated a method of dynamically changing the configuration of a DTE Model. They came up with a set of Prolog rules for making changes to Domain-Type Access Matrix (adding or deleting new domains and types or changing permission sets for a domain-type pair) in such a way that those changes maintain the security level of the current configuration. Just like Hoffman, Tidswell and Potter's work is also in the context of an operating system (specifically a Unix OS) and hence there was no necessity to incorporate any context-based information within the Prolog rules so as to affect the user permissions during run-time. Even in instances of DTE deployment for application systems ([4], [7]) there was no attempt to incorporate any context-based information. Hence in all

DTE implementations that do not incorporate contextual information, authorizations are based on entries in the Domain-Type Access Matrix and the question of classifying an individual authorization request does not arise.

The authorization framework described in this paper differs from the approaches described in the previous paragraph in two ways: First, the focus of the authorization framework described in this paper is in the context of an application system and not an operating system. The second difference is that with the inclusion of contextual information, each individual authorization request is assigned a type and the conditions needed to satisfy the requirements for that authorization type are checked dynamically. Once the conditions are checked using certain contextual information such as the "current user work assignments", the DTE subject-domain table is read to assign the correct domain (based on the invoked subject) to the user session. The actual permissions required for the subject to carry on its intended operation are read off from another DTE table – i.e., the Domain-Type Access Matrix. Thus the DAFMAT framework strives to use as much information as possible using the hybrid RBAC-DTE model structure while providing support for dynamic multiple authorization types instead of obtaining all the data needed for authorization from logical rules as in [6]. This design goal reduces the information that is to be processed by the logic-driven authorization engine and hence the complexity of the logical implications as well.

4. Development of DAFMAT Framework

Let us now examine in detail each of the two components (i.e., hybrid RBAC-DTE access control model and logic-driven authorization engine) of DAFMAT framework mentioned in section 2. The hybrid access control model consists of:

(a) Authorization Entities
(b) Relationships among Authorization Entities and
(c) Constraints governing the relationships.

The logic-driven authorization engine consists of a first order predicate logic based processor capable of processing Prolog-like rules. The authorization engine will also have additional capabilities to print out the predicates and the binding values that determined the approval or denial of an authorization request.

4.1 Authorization Entities

The main authorization entities in DAFMAT framework are:

(a) USER
(b) ROLE
(c) SUBJECT
(d) DOMAIN and
(e) OBJECT-TYPE.

Roles in DAFMAT framework represent job positions. Subjects represent the programs or executables that contain transactions for carrying out business process functions that a job position demands. Domains represent the higher-level enterprise functional area within which roles should perform. For example, doctor roles and nurse roles perform within the domain of patient care. The roles of Lab Technicians and Radiologists fall within the domain of clinical testing. An Object-Type represents a grouping of objects carrying related information with reference to an application system or a healthcare function. For example a Patient-Registration Type may consist of a collection of objects or records pertaining to patient demographic, insurance and allergies information. A Patient-Clinical Type may consist of a collection of objects or records pertaining to various clinical tests performed on a patient like X-rays, MRIs, EKG, blood tests etc.

4.2 Relationships among Authorization Entities

The relationships in DAFMAT are mappings from a set of source authorization entities to target authorization entities. There are two types of mappings that are used in DAFMAT. If several (two or more) instances of source entities map to a single instance of target entity, such a mapping is designated as a many-to-one (denoted by the symbol N:1) mapping and the mapping function is represented by the symbol: >>_>. On the other hand, if several instances of source entities map to several instances of target entities, such a mapping is designated as a many-to-many (denoted by the symbol M:N) mapping and the mapping function is represented by the symbol: >>_>>. In the logic database (which is used by the authorization engine) both types of mapping are represented as predicates. The predicate corresponding to each of the mapping functions in DAFMAT is given in parenthesis in italicized font following the mapping function representation.

4.2.1 User-Role Mapping

In DAFMAT every user is assigned a unique role and each role may be assigned several users since a role is a semantic construct for a job position within the healthcare enterprise. Hence the user to role mapping is a many-to-one mapping represented as:

UserRole(user) >>-> role
(*User_Role(user,role)*) (4.2.1)

In the above mapping function representation, "UserRole" is the name of the mapping function, the parameter "user" stands for the source entity USER, the symbol >>->

denotes the many-to-one mapping type and the symbol "role" stands for the target entity ROLE. The corresponding predicate that is used by the authorization engine is given in parenthesis. The name of the predicate is "*User_Role"* and the predicate variables are: *user* and *role.*

4.2.2 Role-Domain Mapping

A domain represents a functional area within an enterprise. In a healthcare enterprise a person occupying a job position (which is represented by a role) rarely performs any tasks outside his/her general functional area as it involves legal and professional competency issues. Hence several roles may be associated with a domain but a role always belongs to a unique domain. Therefore the role to domain mapping is a many-to-one mapping defined using the function name RoleDomain as:

RoleDomain(role) >>-> domain (*Role_Domain (role, domain*)) (4.2.2)

4.2.3 Subject Mappings

Subjects stand for program entities or user agents that perform one or more transactions on behalf of a user and their execution semantics is dictated by the role from which they are invoked. Subjects are often designed as generic program entities that can be invoked by multiple roles and the semantics of execution of a given subject will depend upon the role from which it is invoked. Similarly a role may have to invoke multiple subjects in order to perform a designated task. Hence the subject to role mapping is a many-to-many mapping defined using the function name SubjectRole as:

SubjectRole (subject) >>->> role (*Subject_Role (subject, role)*) (4.2.3)

A subject's access to objects is to be mediated based on the domain to which it belongs and therefore the domain associated with a subject must be unique. In other words every subject is assigned a unique domain (although many subjects may be assigned to a single domain) making the subject to domain mapping a many-to-one mapping which is defined using the function name SubjectDomain as:

SubjectDomain (subject) >>-> domain (*Subject_Domain (subject, domain)*) (4.2.4)

4.2.4 Object-Type Mapping & Domain-Type Access Matrix

An Object-Type stands for a collection of objects carrying related information. Hence each object maps to a unique object-type and there could be many objects within an object- type. Hence the object to object-type mapping is a many-to-one mapping defined using the function name TypeMap as:

TypeMap(object) >>-> object-type (*Type_Map(object, type)*) (4.2.5)

Associated with each object (and hence object-type) is a set of valid access modes (Create, Update, Delete, View etc). In a DTE implementation accesses from a subject to an object are based on the subject's domain and object's type. The set of allowable access modes are represented in the form of a matrix called Domain-Type Access Matrix with domains as rows, object-types as columns and each cell in the domain-object-type pair contains the valid set of access modes granted for that pair. Also an access mode denotes a particular way of accessing an object which is generic to any type of object. Hence the mapping from a domain-object-type pair to an access mode is a many-to-many mapping defined using the function name DteEntry as:

DteEntry (domain, object-type) >>->> access

(*Dte_Entry(domain,type,access)*) (4.2.6)

The entire set of relationships among authorization entities in DAFMAT showing the type of mapping between any two entities is given in Figure 4.1.

4.3 Constraints on Relationships among Authorization Entities

While describing relationships among authorization entities, we have concerned ourselves with mappings involving only a pair of entities. Constraints in DAFMAT are concerned with restrictions imposed on mapping instances for a given pair of entities when taking into account the mapping instances each member of the pair has with a third entity. To give a concrete example let us consider the pair of authorization entities – subject and role and the domain as the third entity. The constraint – "all the roles from which a subject can be invoked should all be assigned to the same unique domain which is associated with the subject" is expressed as:

$\forall$ (s,d,r) {*Subject_Domain (s,d) & Subject_Role(s,r)* ® *Role_Domain (r,d)* }

4.4 Dynamic Authorization Rules and Relevant Facts

The purpose of using a logic-driven authorization engine in DAFMAT in addition to the hybrid access

control model is to support multiple authorization types. Since an authorization type is determined based on context parameters, contextual information is the major type of information that the dynamic authorization rules processed by the authorization engine use. It must be mentioned that the major portion of information that an authorization module based on DAFMAT framework uses pertains to all the mapping information (that forms the hybrid access control model) described in section 4.2. The contextual information and the number of dynamic authorization rules to be evaluated are kept to the minimum to limit the processing requirements of the logic driven authorization engine. The context-based authorization rules are expressions of organizational policies within a healthcare enterprise that seek to limit the locality of service function (e.g., specialty, ward assignment etc) for various healthcare workers like Doctors and Nurses. The locality assignments are done to realize multiple objectives like accountability, integrity and competency of service. For example to order a test, prescribe a medication or authorize a diagnostic procedure for a patient in a Cardiology Ward can only be done by the attending physician for the ward at that time.

The various processing steps involved in DAFMAT framework for authorizing a user action are described in the following sub-sections:

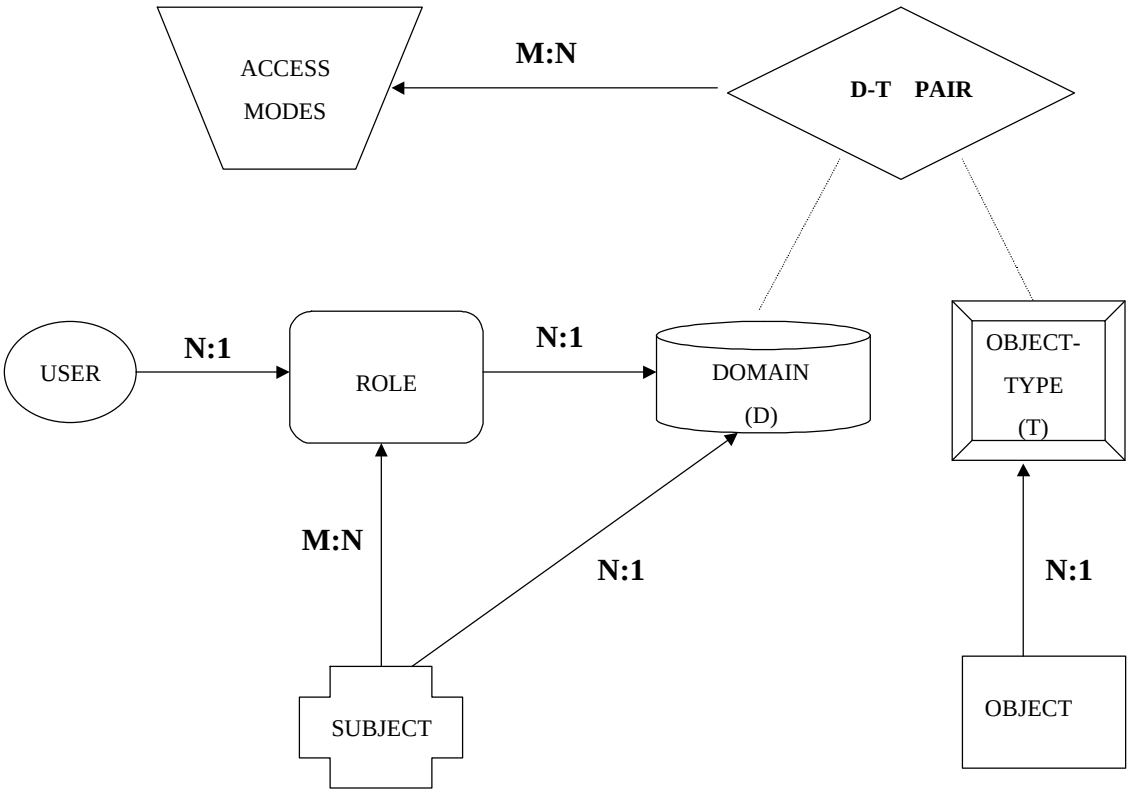

Figure 4.1 – Relationships between Authorization Entities in DAFMAT

4.4.1 Formulation of Authorization Request Predicate

DAFMAT is an authorization framework for supporting applications. Hence it should have mechanisms for translating user actions in an application to the appropriate authorization request which can then be validated. This translation process is enabled by information linking the invoked menu option and application session with the user, role, the subject invoked, the relevant context variable and the priority code. These linkages are obtained using the following session-related functions in DAFMAT.

Current_Action (menu_option, session) – the menu option and session pertaining to the current user action. (4.4.1)

Session_User (session, user) – the user associated with the application session (4.4.2)

Session_Role(session, role) – the role activated for the session (4.4.3)

Menu_Operation(menu_option, subject) – subject invoked by the menu option (4.4.4)
Menu_Context(menu_option, cv) – the context variable associated with menu option (4.4.5)
Session_Priority(session, pc) – the priority code for the session (4.4.6)
(pc = NR for normal request and pc = ER for emergency request)
The usage of the data obtained through above mappings is as follows. Corresponding to the exercise of a menu option in an application session (given by mapping 4.4.1), the variables associated with the action (i.e., the user, role, subject, context variable and priority code) are obtained from predicates 4.4.2 through 4.4.6 respectively. In addition there is a variable required for holding the value string associated with a context variable value (e.g., the value string 'PEDIATRIC' associated with the context variable value 'wardname'). Including this variable (named cv_value) along with other variables discussed above, the authorization request predicate is formulated as follows:

Auth_Req(user,role,subject,cv,cv_value,pc) (4.4.7)

4.4.2 Designating Authorization Type to an Authorization Request

The values for variables 'pc' and 'cv' in an instance of the authorization request predicate 4.4.7 (which represents a specific authorization request) helps to categorize the request into one of the following authorization types:

(a) Normal Authorization Request
(b) Emergency Authorization Request
(c) Context-based Authorization Request

The logic used to categorize the current request (called the authorization type designation logic) into one of above three types is as follows:

4.4.2.1 Authorization Type Designation Logic

```
IF          pc = 'ER'
THEN   Emergency_Auth_Req(user,role,subject)
ELSE
      IF          cv = 'NONE'
      THEN
Normal_Auth_Req(user,role,subject)
      ELSE
Context_Auth_Req(user,role,subject,cv,cv_value)
```

4.4.3 Rules for Validating Different Authorization Types

The last step in approving or denying a user authorization request is to determine the validation conditions for each of the authorization type designations yielded by the logic in section 4.4.2.1. These validation conditions are given below:

4.4.3.1 Validation Conditions for Normal Authorization Request (Normal_Auth_Req):

Normal authorization involves checking whether the role currently active in the user session is one of the roles assigned to the subject under invocation. No other information is involved in this type of authorization.

Subject_Role(subject,role) → Normal_Auth_Req(user,role,subject) (4.4.8)

4.4.3.2 Validation Conditions for Emergency Authorization Request (Emer_Auth_Req):

Emergency authorizations are required in situations where a person who is called upon to perform a healthcare task has the competence and qualifications but does not have the association relationship like the Attending Physician or the Doctor-on-call. Hence the context variable is not relevant for the emergency authorization. When a user creates an application session for making an emergency request (through a stronger form of authentication like two-factor authentication), that session is created with priority code pc equal to ER. Based upon the user clearance for certain emergency tasks, an emergency role instead of a regular role is assigned to the user through the Session_Role function (4.4.3). The mapping from this emergency role to a normal/regular role of the model is stored in a secure area that is different from the location where the hybrid model data for DAFMAT is stored. Verifying whether a mapped regular role is authorized to invoke the subject then becomes the validation condition for authorizing an emergency request.

ER_Role_Map(role,mapped_role) &
Subject_Role(subject,mapped_role) →
Emer_Auth_Req(user,role,subject) (4.4.9)

In summary, an emergency authorization is obtained by a stricter form of authentication and by checking whether the emergency role activated by the session is a valid proxy for a regular role that has the permission to invoke the subject. This is equivalent to finding a binding for the mapped_role variable that will make the above implication true.

4.4.3.3 Validation Conditions for Context-based Authorization Request (Context_Auth_Req)

The validation conditions for emergency authorization and normal authorizations yielded only a single validation

rule. However the validation conditions for context-based authorization will depend upon the context variable (cv) and each context variable gives rise to a different validation rule. Expressing this as a decision tree:

IF cv = 'CTXT_VAR1'
THEN
Subject_Role(subject,role) & <Context Predicate relevant for CTXT_VAR1> →
Context_Auth_Req(user,role,subject,cv,cv_value)
(4.4.10)

4.4.4 The Last Step of the Authorization Process

As already stated in section 2, the last step performed by the DAFMAT-based authorization module of the application, after verification of the validation conditions for the authorization request of the designated authorization type, is to assign the appropriate domain (based on Subject-Domain table entries) to the user session. It is useful to point out at this stage that the verification of the validating conditions for different types of authorizations is the key process that distinguishes DAFMAT from a static authorization framework. In the static framework, the authorization required for invoking the subject is directly obtained from the entries in the Domain-Type Access Matrix by determining the subjects, objects/access involved in the operation after the system determines the domains associated with subjects and types associated with objects. However, in a typical healthcare facility, the eligibility to invoke a subject by a user is not determined solely through a static role-subject relationship, but through some temporal relationships each user has with application domain specific variables (e.g., Attending Physician for a patient). The incorporation of such temporal relationships (which we have called as contextual information) into the evaluation of role-subject association means that the authorization process has to be dynamic involving processing of rules at the time of authorization requests.

5. DAFMAT Framework for ADT System

Let us now illustrate the application of DAFMAT authorization framework described in the previous section to the Admissions, Discharge and Transfer system (ADT). Before we do this a brief explanation of the functionality of the ADT system and the type of information it handles is in order.

The Admissions, Discharge and Transfer system (ADT) is designed to perform all functions relating to admission, discharge and internal transfer of patients in a healthcare facility. Examples of internal transfer functions include transfer of patients from one bed/room to another within a ward, from one hospital service/ward to another or from one status to another (e.g., inpatient to outpatient etc). The ADT system is the entry- point for capture of all patient-related information like patient demographics, insurance and allergies. In this paper we have coined the name "Patient Registration Object" to stand for an encapsulated object that contains several types of patient-related information referred to earlier. Similarly we have given the name "Patient Location Object" that contain information about the bed, room and type of wards where the patient had undergone treatment at the healthcare facility. In addition, we have the "Patient Clinical Object" that is composed of a set of objects carrying information about all the clinical tests the patient has undergone (e.g., lab tests, radiology tests etc).

Since the underlying objective in describing the application of DAFMAT framework for ADT is only to demonstrate the logic used for arriving at different authorization types, we have chosen to give only a sample of the set of authorization entities and their relevant relationships in ADT. Hence the set of roles, subjects, domains and types discussed here is not a complete list of authorization entities and relationships found in any practical ADT system deployed in a healthcare setting.

The data regarding users, roles, subjects and domains (which constitute the RBAC-DTE hybrid access control model data for ADT) and their relationships are given in section 5.1 while the data that is needed for use by the logic-driven authorization engine is given in section 5.2.

5.1 RBAC-DTE Model Data for ADT

The sample "RBAC-DTE Model Data" set used in our illustration consists of the following – 4 users, 4 roles, 4 subjects and 3 domains. The user John is an Admissions Clerk and hence assigned to the *admissions_clerk* role. Smith is in charge of making and altering the assignment of patients to rooms/beds within a ward and hence assigned to the *ward_scheduler* role. Susan is a registered nurse who puts in orders for lab tests for patients (after being authorized by the attending physician) and hence assigned to the *registered_nurse* role. Patricia is a facilities specialist who is in charge of handling the entry of patients to acute care facilities like Intensive Care Units (ICUs) and Chemotherapy facilities. The assigned role for Patricia is *facilities_specialist*. The users assigned to the admissions_clerk role invoke the Admission_proc and Discharge_proc to perform their job functions. The ward_scheduler and facilities_specialist roles invoke the Transfer_proc to perform the functions of transferring patients into the general wards and special wards respectively. The registered_nurse role can invoke the Lab_Orders_proc. The relevant user-role, role-domain, subject-role, subject-domain mappings and the corresponding Domain-Type Access Matrix entries are given in tables 5.1through 5.5

Table 5.1- User-Role Mapping

User	Role	Predicate
John	admissions_clerk	User_Role(john, admissions_clerk)
Smith	ward_scheduler	User_Role(smith,ward_scheduler)
Susan	registered_nurse	User_Role(susan,registered_nurse)
Patricia	facilities_specialist	User_Role(patricia, facilities_specialist)

Table 5.2 - Role-Domain Mapping

Role	Domain	Predicate
Admissions_clerk	patient_mgmt_domain	Role_Domain(admissions_clerk,patient_mgmt_domain)
Ward_scheduler	facility_mgmt_domain	Role_Domain(ward_scheduler, facility_mgmt_domain)
Registered_nurse	care_provider_domain	Role_Domain(registered_nurse, care_provider_domain)
Facilities_specialist	facility_mgmt_domain	Role_Domain(facilities_specialist, facility_mgmt_domain)

Table 5.3 - Subject-Role Mapping

Subject	Role	Predicate
Admission_proc	admissions_clerk	Subject_Role(admission_proc,admissions_clerk)
Discharge_proc	admissions_clerk	Subject_Role(discharge_proc,admissions_clerk)
Transfer_proc	ward_scheduler, facilities_specialist	Subject_Role(transfer_proc,ward_scheduler) Subject_Role(transfer_proc,facilities_specialist)
lab_orders_proc	registered_nurse	Subject_Role(lab_orders_proc,registered_nurse)

Table 5.4 - Subject-Domain Mapping

Subject	Domain	Predicate
Admission_proc	patient_mgmt_domain	Subject_Domain(admission_proc, patient_mgmt_domain)
Discharge_proc	patient_mgmt_domain	Subject_Domain(discharge_proc, patient_mgmt_domain)
Transfer_proc	facility_mgmt_domain	Subject_Domain(transfer_proc, facility_mgmt_domain)
lab_orders_proc	care_provider_domain	Subject_Domain(lab_orders_proc, care_provider_domain)

Table 5.5 - Domain-Type Access Matrix

Domain	Object-Type / Access Modes		
	Patient Registration Type	Patient Location Type	Patient Clinical Type
Patient_mgmt_domain	C, U, D,V	D, V	
Facility_mgmt_domain		C,U,V	
Care_provider_domain	V	V	C, U, V

Access Mode Codes: C – Create , U – Update, D – Delete , V – View

The pictorial representation of authorization entities and their relationships (which constitutes the RBAC-DTE hybrid-access control model) for ADT is given in Figure 5.1.

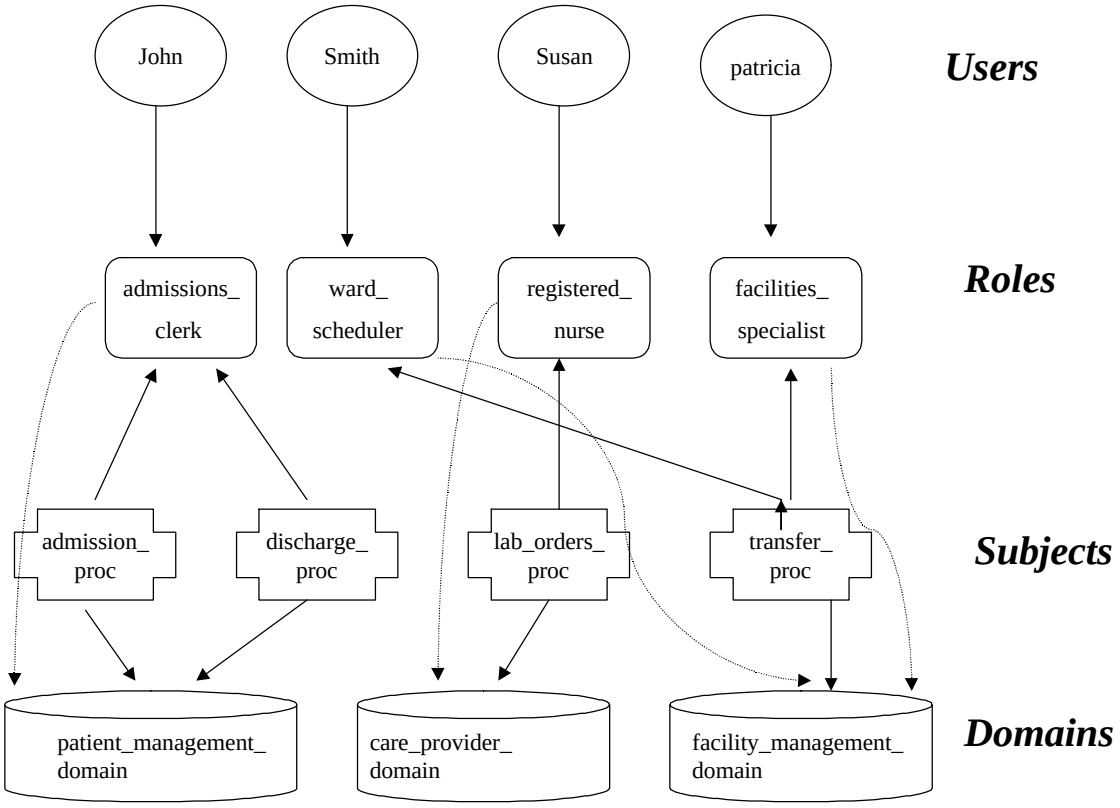

Figure 5.1 – RBAC-DTE HYBRID ACCESS CONTROL MODEL FOR ADT

(NOTE: The Role to Domain Mapping is shown in dashed lines)

5.2 Data for Logic-Driven Authorization Engine for ADT

Recall that the first processing step in DAFMAT, after building the RBAC-DTE model for the application, is the formulation of Authorization Request Predicate 4.4.7. In this predicate, the bindings for the variables user, role, subject and pc (priority code) are obtained using values of the session-related mapping functions (4.4.2 through 4.4.6) and the RBAC-DTE model data given in the tables of section 5.1. The value for cv_value (value string corresponding to the context variable value –e.g., PEDIATRIC for context variable value 'wardname') is an intrinsic parameter value in the user action in the application. Now the only value required is for cv (context variable) in predicate 4.4.7. This value can be obtained from table 5.6 which provides a mapping of Menu Options to Context Variable value in ADT system. The next category of data that we need is the instantiation of the context predicate (in expression 4.4.10) that is relevant for context variable values encountered in the ADT system. Incorporating the relevant context predicate the instantiated "validation condition for context-based authorization request" for the menu options – "Change Beds/Room" and "Transfer to Acute Care" respectively are:

Subject_Role(subject,role) & Equals(cv,wardname) & Ward_Assignment(user,cv_value) → Context_Auth_Req(user,role,subject,cv,cv_value) (5.2.1)

Subject_Role(subject,role) & Equals(cv, facilitytype) & Specialist_in_Charge(cv_value,user) → Context_Auth_Req(user,role,subject,cv,cv_value) (5.2.2)

The last but not the least important category of DAFMAT data required in ADT is for establishing the truth values for ward_assignment and specialist_in_charge predicates (in the instantiated validation conditions 5.2.1 and 5.2.2) and for the validation condition for emergency authorization. These are given in tables 5.7 through 5.9 respectively.

Table 5.6 - Menu Option to Context Variable Mapping in ADT

Menu Option	Context Variable (CV)
1. Admit Patient	NONE
2. Change Beds/ Room	wardname
3. Transfer to Acute Care	facilitytype
4. Order Lab Tests	patientname
5. Discharge Patient	NONE

Table 5.7 – Truth Values for Predicate Ward_Assignment

Ward	Administrator	Predicate
PEDIATRIC	smith	Ward_Assignment(smith,'PEDIATRIC)
MATERNITY	mell	Ward_Assignment(mell,'MATERNITY')

Table 5.8 – Truth Values for Predicate Specialist_in_Charge

Facility	Specialist	Predicate
ICU	mike	Specialist_in_Charge(ICU,mike)
CHEMO_THERAPY	patricia	Specialist_in_Charge (CHEMO_THERAPY,patricia)

Table 5.9 – Emergency to Regular Role Mappings (for validating Emergency Authorization Requests)

Emergency Role	Regular Role	Predicate
Facilities_manager	facilities_specialist	ER_Role_Map(facilities_manager,facilities_specialist)
Facilities_manager	ward_scheduler	ER_Role_Map(facilities_manager,ward_scheduler)

5.3 Authorization Processing in ADT

Based upon the data needed for ADT authorization given in the previous three sections, let us now illustrate as to how the following three user requests will be processed by the DAFMAT framework for ADT.

5.3.1 The user 'smith' wants to swap beds for a couple of patients in the pediatric ward and makes a normal request :

The menu option 'smith' will use is item 2 in table 5.6 (i.e., Change Beds/Room). Making use of mapping functions 4.4.2 through 4.4.6, the values for the various session-related variables for smith's current ADT session are obtained as follows:

user = smith
role= admissions_clerk
subject = transfer_proc
cv=wardname
pc=NR

In addition since smith is invoking this menu for performing the bed-swapping operation for patients in the pediatric ward the value string associated with context variable value 'wardname' will be:

cv_value = 'PEDIATRIC'

Hence the authorization request predicate that will be formulated for authorizing smith's request will be:

Auth_Req(smith,ward_scheduler,transfer_proc,wardname, 'PEDIATRIC',NR)

As per the logic given in section 4.4.2.1 this request will be designated as a context-based authorization type and hence transformed into

Context_Auth_Req(smith,ward_scheduler,transfer_proc,wardname, 'PEDIATRIC')

Based on the value of the context variable (i.e., wardname) the validation condition on the left hand side of logical implication 5.2.1 becomes the relevant condition to be checked for smith's request and the instantiation of this condition will yield:

Subject_Role(transfer_proc,ward_scheduler) &
Equals(cv,wardname) &
Ward_Assignment(smith,'PEDIATRIC')

Out of the three predicates above, the first one is true because of entry in table 5.3, the second one is trivially satisfied and third is true because of the entry in truth value table 5.7. Since the condition for smith's request is satisfied, smith's request to swap beds for a couple of patients in the pediatric ward will be authorized.

5.3.2 The user 'patricia' wants to transfer a patient to ICU and makes a normal request :

The menu option 'patricia' will use is item 3 in table 5.6 (i.e., Transfer to Acute Care). Again making use of mapping functions 4.4.2 through 4.4.6, the authorization request predicate for patricia's current ADT session will be:

Auth_Req(patricia,facilities_specialist,transfer_proc,facilitytype,'ICU',NR)

Again as per the logic given in section 4.4.2.1 this request will be designated as a context-based authorization type and hence transformed into:

Context_Auth_Req(patricia,facilities_specialist,transfer_proc,facilitytype,'ICU')

Based on the context variable (i.e., facilitytype) the validation condition on the left hand side of logical implication 5.2.2 becomes the relevant condition to be checked for patricia's request and the instantiation of this condition will yield:

Subject_Role(transfer_proc,facilities_specialist) &
Equals(cv, facilitytype) &
Specialist_in_Charge('ICU',patricia)

Since a binding cannot be obtained for the third predicate in the condition (i.e., Specialist_in_Charge('ICU',patricia) from the truth table 5.8, the authorization request will be denied.

5.3.3 The user 'patricia' wants to transfer a patient to ICU and makes an emergency request :

This is an identical to the authorization request in 5.3.2 except that it is an emergency request instead of a normal request from the same user for the same operation. Now the authorization request predicate for this request will be (assuming that an emergency role called facility_manager is assigned to patricia by the session):

Auth_Req(patricia,facilities_manager,transfer_proc,
facilitytype,'ICU',ER)

Again as per the logic given in section 4.4.2.1 this request will be designated as an emergency authorization type and hence transformed into

Emergency_Auth_Req(patricia,facilities_manager,transfer_proc,facilitytype,'ICU')

The validation condition for this request becomes (using expression 4.4.9):

ER_Role_Map(role,mapped_role) &
Subject_Role(subject,mapped_role)

Based on the following entries in tables 5.9 and 5.3:
ER_Role_Map(facilities_manager,facilities_specialist) and Subject_Role(transfer_proc,facilities_specialist, we have satisfied the above condition and hence the application will approve this authorization request.

6 Assurance Measures for DAFMAT Administration

Let us now at the look at some of the assurance measures that may be required for the administration of DAFMAT framework in an enterprise setting. As far as the "hybrid access control model" component is concerned, the inherent structural constraints of the model augmented with application-specific constraints provide a measure of protection against unsafe configurations. The following assurance measures are suggested for the 'logic-driven authorization engine" component: (a) Authorization Rules are created with a named Rule Set and associated with a named "hybrid access control model set". (b) Rules in the Authorization Rule set are maintained centrally by a trusted administrator while the "hybrid access control model set" for each application is maintained by the individual application/system administrators. (c) The

Authorization Type Designation assigned to each authorization request and the predicate bindings used in approving or denying the request are recorded in an audit log and periodically reviewed for correct authorization assignments.

7. Conclusions and Scope for Future Work

Authorization mechanisms that support multiple authorization types can provide effective control of access to resources in many vertical market applications. The DAFMAT framework can provide this critical functionality using a hybrid access control model and a logic-driven authorization engine that makes use of contextual information. The same authorization engine can be used for dynamic reconfiguration of Domain-Type access matrix entries as well as for dynamic User-Role and Subject-Role assignments. However the inclusion of these features may make the authorization engine difficult to build and result in performance penalties for the authorization mechanism. However sophisticated authorization rules can be implemented (without significant degradation of system response times) through the use of a common security kernel that will mediate access to a family of application systems within a healthcare enterprise as has been done in the VA healthcare settings. However the presence of a security kernel may make integration of COTS application systems into the IT resources of a healthcare enterprise an expensive operation. Since IT infrastructures in most healthcare enterprises are heterogeneous, the most preferred alternative is to build application-level controls for authorizations by making sure that appropriate access control models and mechanisms are used to capture the enterprise authorization policy requirements.

8. References

[1] J.F.Barkley, A.V.Cincotta, D.F.Ferraiolo,S.Gavrila and D.R.Kuhn. "Role based access control for world wide web" http://hissa.ncsl.nist.gov/rbac/rbacweb/paper.ps, April 1997.

[2] W.Boebert and R.Kain. "A Practical Alternative to Hierarchical Integrity Policies" Proc. 8th National Computer Security Conference, October 1985.

[3] D.Ferraiolo, J.Cugini, and D.R.Kuhn. "Role Based Access Control (RBAC): Features and Motivations" Proc. 11th Annual Computer Security Applications Conference, December 1995.

[4] P.Greve, J.Hoffman and R.Smith. Using Type Enforcement to assure a configurable guard. Proc. 13th Annual Computer Security Applications Conference, December 1997.

[5] J.Hoffman. "Implementing RBAC on a type enforced system" Proc. 13th Annual Computer Security Applications Conference, December 1997.

[6] S. Jajodia, P.Samarati and V.S.Subrahmanian. "A Logical Language for Expressing Authorizations" IEEE Symposium on Security and Privacy 1997, p31-42.

[7] K.A. Oostendorp, L.Badger, C.D. Vance, W.G. Morrison, M.J. Petkac, D.L Sherman, D.F Sterne "Domain and type enforcement firewalls" Proc 13th Annual Computer Security Applications Conference, December 1997.

[8] R.S. Sandhu, E.J.Coyne, H.L.Feinstein and C.E.Youman. "Role Based Access Control Models" IEEE Computer, vol 29, Num 2, February 1996, p38-47.

[9] Security and Electronic Signature Standards; Proposed Rule. Federal Register, Vol 63, No. 155, August 12, 1998.

[10] J. Tidswell and J. Potter "An Approach to Dynamic Domain and Type Enforcement" Microsoft Research Institute, Department of Computing, Macquarie University, NSW Australia.