An Application of Integrating Role and Lattice Based Access Control in Database Engineering

Ioannis Mavridis [1], George Pangalos [2], Stavros Kortesis [2] and Isabella Kotini [3]

[1] Department of Applied Informatics
University of Macedonia, 54006 Thessaloniki, Greece
Email: mavridis@uom.gr

[2] Informatics Laboratory, General Department, Faculty of Technology
Aristotle University, 54124 Thessaloniki Greece
Email: {pangalos,stk}@auth.gr

[3] Department of Electrical and Computer Engineering, Faculty of Technology
Aristotle University, 54124 Thessaloniki Greece
Email: kotini@egnatia.ee.auth.gr

Abstract. The development of a database access control system is usually accomplished during the earlier stages of the life-cycle of information systems. However, to follow up the evolution of information systems there is often a need to update their security mechanisms without having to reengineer them. The significant security administration workload involved in such a repetitive process can be reduced if an appropriate security model is used and is implemented according to a clearly defined methodology for naming, designing and constructing the components of the aimed access control system. In this paper, we present an application of a step-by-step methodology for engineering an access control system in the healthcare domain, which is based on the mechanisms provided by the already known eMEDAC security model. The demonstrated application results in an easily updated access control system that is capable of providing fine-grained and multilevel access control at the same time.

1 Introduction

In today's information systems incorporating security is a process that must be taken into account from the very first steps of the software development process. Security engineering involves the process of specifying the security requirements from application requirements and organizational policies, and subsequently designing the appropriate security mechanisms that are required for their realization.

One of the principal security technologies is access control, which focuses on protecting resources (e.g. stored data) by restricting access to authorized users only with an appropriate access control system [1]. A typical access control system includes subjects, who access objects through operations [2, 3]. Functionally, it consists of a security policy and a set of authorization rules, that is information regulating the access modes to be followed by subjects upon access to objects. Moreover, it involves a set of access control mechanisms that check the access requests against the authorization rules. Requests may then be allowed, denied or modified.

The process of engineering an access control system results in a set of authorization rules that are implemented using properly configured access control mechanisms. This process must be accomplished during the design phase of development life-cycle of information system [4]. However, to follow up the evolution or to support the maintenance of existing information systems there is often a need to update their security mechanisms without having to reengineer them, especially in legacy information systems. Such a repetitive process of redefining the components of an access control system is difficult, entails a significant administrative workload, as well as a considerable possibility of human errors. Therefore it must be based on well-defined methods and guidelines for effective security requirements engineering. Moreover, becomes more obvious the need for a clearly defined engineering methodology for naming, designing and constructing the components of an access control system that implements an appropriate security model.

Access control systems that provide access control mechanisms with both mandatory and discretionary features are particularly suitable in privacy critical environments, as for example the healthcare information systems [5]. Such an access control system could be implemented according to the already known eMEDAC security model [6], which is based on the Bell-LaPadula model [7] and exploits the administrative advantages of RBAC components [8, 9]. EMEDAC provides advanced security characteristics since it enforces in a unified way permission-based and multilevel access control based on roles rather than on particular users. The advantages of eMEDAC concern also providing authorization rules with fine-grained granularity and separating application from authorization logic, while configuring access control mechanisms using application-specific information but without affecting the design scheme of primary application data.

In this paper, an application of a step-by-step methodology for engineering the mechanisms of an access control system conforming to the eMEDAC security model, during or after the completion of the design phase of the initial development or any subsequent maintenance software life-cycle of a database system [10], is presented. In section 2, we give a brief overview of the eMEDAC access control model. Then, we describe in section 3 the engineering methodology and in section 4 we present an application example in the healthcare domain. Our conclusions are given in section 5.

2 A Brief Overview of eMEDAC

According to eMEDAC security model [6], a protected object is a data set, which can be a database, as a whole or a part of any size. Subjects (users) have a number of assigned roles, which are collections of permissions (privileges) to access objects. User roles are authorized to execute specific database operations on predefined sets of data on behalf of users that activate them. Data sets can be derived according to the fragmentation method [11], which is described later. The access window of a user role can be seen as the combination of a number of those data sets. For a more precise definition of the concept of a data set one can consider it as the largest area of the database to which two or more user roles have accesses in common. On the other hand, there is no user role that has access only to a part of a specific data set. Each one data set is implemented as a view. Views, which can be defined with appropriate SELECT statements of SQL, provide a very powerful mechanism for controlling what information can be accessed [12]. As a result, instead of authorizing users to have access to the base relations of a database system, user roles are permitted to access only to view relations, which correspond to specific data sets.

The main type of access control mechanisms used in eMEDAC is called Hyper-Node Hierarchy (HNH). HNH is an abstraction and permission inheritance structure for constructing User Role Hierarchies (URH) and Data Set Hierarchies (DSH). In addition, HNHs are used for deriving security labels (that include a security level and a category set) of user roles and data sets in order to exercise multilevel access control.

A HNH is actually a set of nodes and connections of different types (Figure 1). A node is characterized as either hyper or dummy. Nodes are connected to each other with either a branch or a link. A branch connects a node with its ancestor node, which has a different (higher) security level. Links connect between nodes of the same security level. A hyper node HN has a security level that is equal to the number of branches reached when moving upward in the hierarchy. The category set of a hyper node HN consists of all its possible first ancestors.

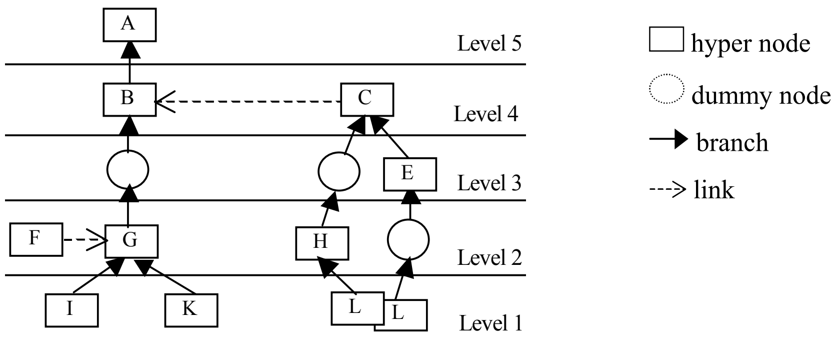

Fig. 1. The Hyper Node Hierarchy (HNH)

The HNH mechanism can provide hierarchies of any depth, with derived security levels and categories. Access control metadata can be centrally defined and stored separately from the application data holders. A flexible number of refinement levels can be specified, which is not strictly equal to the number of mandatory security levels. However, the HNH mechanism specifies (in a mandatory way) a certain number of security levels that cannot be overridden (in fig.1 maximum number of levels is 5).

3 The Engineering Methodology

The database access control engineering methodology consists of a number of steps that are presented briefly in the following subsections.

3.1 Step 1: Organizational Role Engineering

First of all, the security designer works on a set of requirement specifications of a specific application to locate the organizational user roles [13]. These roles can be categorized according to the specified blocks of activities that characterize the type of the organization under consideration, providing so an initial URH.

3.2 Step 2: Security Requirements Derived from the High Level Security Policy

Subsequently, the security designer works in the context of regulations that are in effect in the specific organization and in accordance to the overall high level security policy (HLSP) to specify the security requirements of the database system [14]. A critical starting point is the refinement of those regulations in order to produce a set of principles and guidelines that affect primarily the overall security of the organization and finally constitute the security requirements derived from the HLSP.

3.3 Step 3: Need-to-know Requirements of User Roles

The need-to-know requirements of user roles are derived from the functional design specifications that have been produced by the application designers. For every single organizational user role, the security designer records the area of the database where it needs to have access in order to accomplish a number of assigned tasks.

3.4 Step 4: Final Set of Security Constraints

The final set of security constraints is derived by combining the security requirements derived from the HLSP (step 2) and the need-to-know requirements of user roles (step 3). We distinguish between three different types of security constraints [11]:

- Simple constraints define vertical subsets of data tables to restrict users from accessing certain columns.
- Content-based constraints define horizontal subsets of data tables to express access restrictions based on the content of certain tuples.
- Complex constraints define horizontal subsets of data tables depending on the values of attributes of different tables, in order to express access restrictions that span several tables.

3.5 Step 5: Fragmentation of Data Scheme

Based on the final set of security constraints and the already specified set of data tables, the security designer performs a fragmentation (structured decomposition) of the relational data scheme that results in a set of disjoint data sets, such that the access window of a given user role is not defined over a subset of a data set only. The resulting fragmentation schema can be considered as a DSH without classification levels. More specifically, each type of security constraint causes a corresponding operation of data fragmentation [11], as follows:

- Vertical fragmentation is the projection of the relational schema onto a subset of its attributes. It is caused by simple security constraints.
- Horizontal fragmentation is a subdivision (selection) of a relation into a subset of its tuples that is caused by content-based constraints.
- Derived-horizontal fragmentation is a subdivision (selection) of a relation into a subset of its tuples that is caused by complex constraints.

3.6 Step 6: Initial Permission Set

At this point, two different types of mappings between user roles and data sets are defined:

- MRD : $UR_i \rightarrow \{DS_1, DS_2, ..., DS_j\}$, is a function mapping a user role UR to the group of data sets DS that has access,
- MDR : $DS_i \rightarrow \{UR_1, UR_2, ..., UR_j\}$, is a function mapping a data set DS to the group of user roles UR having access to this data set.

The security designer transforms the final set of security constraints (step 4) to a set of MRDs (user role to data set mappings). Then, based on specified MRDs, the security designer constructs an initial permission set, which is actually a set of MDRs (data set to user role mappings) to express the access rights of user roles on data sets.

3.7 Step 7: Data Classification

In order to assign appropriate classification levels and categories to data sets, an automated labeling process is performed, which is based on the assumption that a data set accessed by many user roles cannot contain sensitive information, whereas a data set accessed by few user roles only may have sensitive data [11]. Such a situation leads to the assignment of a low classification level to the former data set and a high classification level to the latter. Furthermore, we assume that a data set belongs to the same categories of the user role that needs to access it, as defined in the initial permission set (step 6).

Based on the cardinality of particular MDRs (how many user roles have access to every data set), the Data Set Hierarchy of step 5 is redesigned by assigning classification levels to data sets. All the ancestral data sets, as well as the base data tables, are assigned the highest classification level of their child. Then, in order to overcome possible conflicts and to encompass content-based security features in the final access control system, the security designer may perform a manual modification of the classification levels that have been assigned automatically.

3.8 Step 8: Refinement of User Roles

Based on the set of MRDs defined in step 6, a new User Role Hierarchy (without clearance levels) is produced that includes, in addition to the nodes of organizational user roles and categories, a number of possible new logical user roles, which may result from the refinement of organizational user roles.

3.9 Step 9: Final Permission Set

As a consequence of the refinement of the organizational user roles and the definition of new logical user roles, the initial permission set has to be reconstructed. In the final permission set, a number of new logical roles may replace the entries of some original organizational user roles that have been refined, providing so an adequate level of granularity for fine-grained access control.

3.10 Step 10: User Role Classification

Finally, extending the automated labeling process for user roles, it is assumed that in order to have the ability to access a number of data sets (according to a particular MRD), a user role must be assigned a clearance level that is equal to the highest classification level of those data sets. As a result, according to the final versions of the permission set and the DSH, a final URH with clearance levels is constructed.

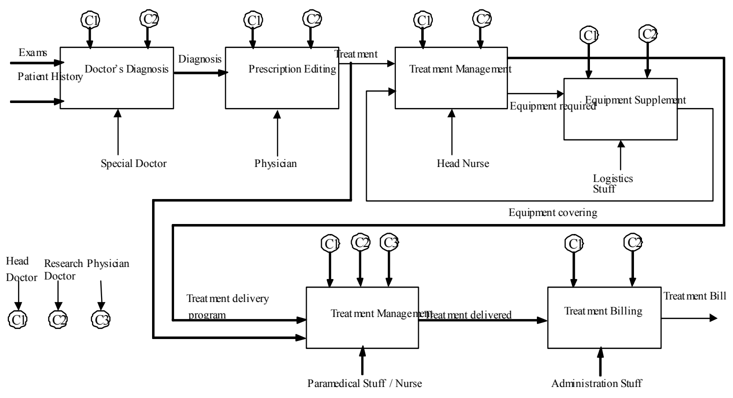

Fig. 2. The SADT diagram of the healthcare subsystem

4 Application Example in a Healthcare Subsystem

In this section, an example from the healthcare domain is described to illustrate the application of the presented engineering methodology. More specifically, it concerns a part of a healthcare information system (HIS) for patient diagnosis, prescription editing and treatment. The entire experimental implementation has taken part in co-operation with a major Greek hospital and revealed the advantages of the proposed methodology in providing clearly defined steps for engineering an efficient and flexible access control system in healthcare environments.

According to step 1, the specific functional specifications of the healthcare application subsystem are recorded in a SADT diagram [15], which includes block activities and involved organizational roles that perform those activities (Figure 2).

User roles involved in the SADT diagram are then categorized, according to the general model of a HIS architecture that has been defined by the ISHTAR group ([16]), in five main categories of activities, which correspond to ward, medical, administration, logistics and other activities (Table 1).

Categories	*User roles*
Medical (M)	Head Doctor (HD), Special Doctor (SD), Physician (PH)
Ward (W)	Head Nurse (HN), Nurse (NU), Paramedical Staff (PS)
Administration (A)	Admin Staff (AS)
Logistics (L)	Logistics Staff (LS)
Other (O)	Research Doctor (RD)

Table 1. User Role categories

As a result, an initial User Role Hierarchy is constructed to represent the user roles that are included in each category, without specifying the clearance levels of the particular user roles, as shown in the following figure 3.

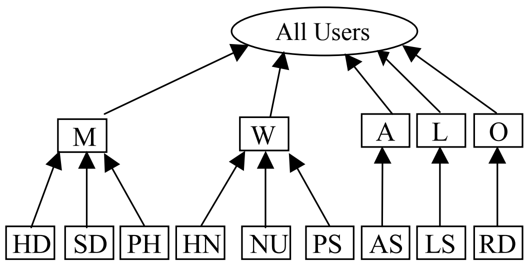

Fig. 3. The initial URH

In the rest of this example we work only with user roles of categories Ward and Others, as well as with a part of the data relational schema that includes the following data tables:

- PAtient (PAcode, PAname, PAaddress, PAtelephone, PAsocialcode, PAsex, PA-birthday, PAbloodcode, PAvipflag)
- PRescription (PRcode, PRpacode, PRclinic, PRdoctor, PRdisease, PRdrug, PRdose, PRdatefrom, PRdateto, PRtype, PRdescription, PRcareinstruct)

In step 2, a set of security requirements (principles and guidelines) for the specific healthcare institution is defined as follows:

- health care workers should have access to those medical data only that belong to patients they have a duty to care for.
- patients entrust physicians to protect their medical data and expect physicians and associated medical staff to protect their privacy rights.
- medical data may be accessed by public health officials and researchers for specific purposes only.
- personal data of patients characterized as VIPs (very important persons), may be accessed by the highest medical staff (e.g. head doctors) only.

In step 3, the need-to-know requirements of user roles HN, NU, PS and RD are specified to concern both data tables PA and PR. Then, in step 4, the following set of security constraints is defined (Table 2).

User Role	*Security Constraints on accessing parts of tables*
HN	PA: personal data (name, address, telephone, socialcode) with PAvipflag='VIP' PR: column PRdisease
RD	PA: personal data of all patients, as well as all columns when PRdisease ≠'HIV' PR: PRdisease when PRdisease ≠ ‘HIV’
NU	PA: personal data, as well as all columns when PAvipflag = 'VIP' PR: all columns when PRtype ≠ 'nursing'
PS	PA: personal data, as well as all columns when PAvipflag = 'VIP' PR: all columns when PRtype ≠ 'therapy'

Table 2. Security constraints of user roles

In step 5, a fragmentation process is performed for each data table, based on the security constraints of particular user roles, as follows:

- User role HN causes a vertical fragmentation of table PA onto data sets (fragments) PA1 (columns with personal data) and PA2 (columns with no personal data), as well as a horizontal fragmentation of data set PA1 onto data sets PA11 (for PAvipflag='VIP') and PA12 (for Avipflag≠'VIP'). User roles UN and PS cause a vertical fragmentation of data set PA2 onto data sets PA21 (for PAvipflag='VIP') and PA22 (for PAvipflag≠'VIP'). User role RD causes a derived horizontal fragmentation of data sets PA21 and PA22 onto data sets PA211, PA212, PA221 and PA222, based on the evaluation of predicate PRdisease='HIV' defined on table PR.

- User role HN causes a vertical fragmentation of table PR onto data sets PR1 (PRdisease) and PR2 (rest of columns). User role RD causes a horizontal fragmentation of data sets PR1 and PR2 onto data sets PR11, PR12, PR21 and PR22 based on the evaluation of predicate PRdisease='HIV'. User roles NU and PS cause a horizontal fragmentation of data sets PR11, PR12, PR21 and PR22 to PR111, PR121, PR211 and PR221 based on the evaluation of predicate PRtype= 'nursing', and PR112, PR122, PR212 and PR222 based on the evaluation of predicate PRtype= 'therapy', respectively.

The resulting fragmentation schema is a Data Set Hierarchy without levels (Figure 4).

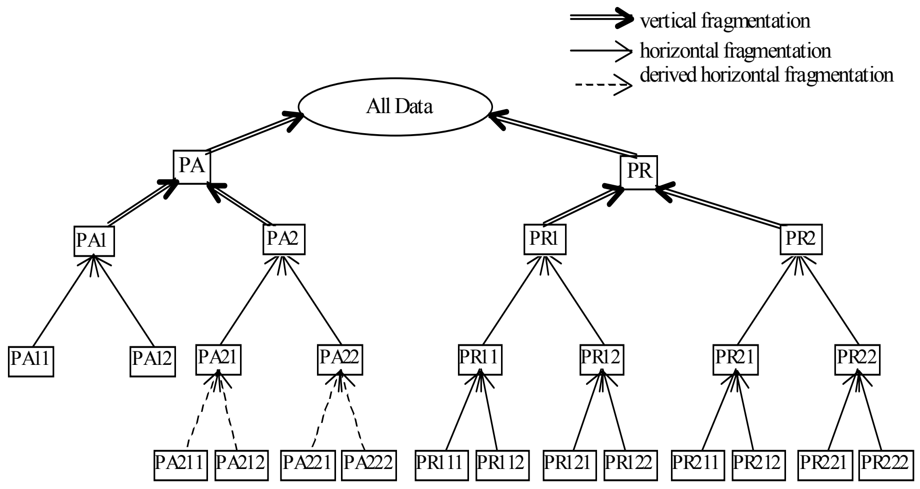

Fig. 4. The fragmentation schema

In step 6, the mappings from user roles to data sets (Table 3) and an initial permission set (set of MDRs) are defined (Table 4).

(user role)	*{data sets}*
HN	{PA12, PA2, PR2}
NU	{PA22, PR111, PR121, PR211, PR221}
PS	{PA22, PR112, PR122, PR212, PR222}
RD	{PA211, PA221, PR1, PR21}

Table 3. Mappings from user roles to data sets (MRDs)

In step 7, the automatic assignment of classification levels to data sets according to mapping cardinalities is performed, as shown in table 5.

(data set)	*{user roles}*
PA11	{}
PA12, PA212	{HN}
PA211	{HN, RD}
PA221	{HN, NU, PS, RD}
PA222	{HN, NU, PS}
PR111, PR121	{NU, RD}
PR112, PR122	{PS, RD}
PR211	{HN, NU, RD}
PR212	{HN, PS, RD}
PR221	{HN, NU}
PR222	{HN, PS}

Table 4. Mappings from data sets to user roles (MDRs)

Data Sets	*Cardinality*	*Level*
PA11	0	5
PA12, PA212	1	4
PA211, PR111, PR112, PR121, PR122, PR221, PR222	2	3
PA222, PR211, PR212	3	2
PA221	4	1

Table 5. Automatic labeling of data sets

Based on the above MDR mappings, the DSH is reconstructed, having now classification levels for data sets, as shown in figure 5.

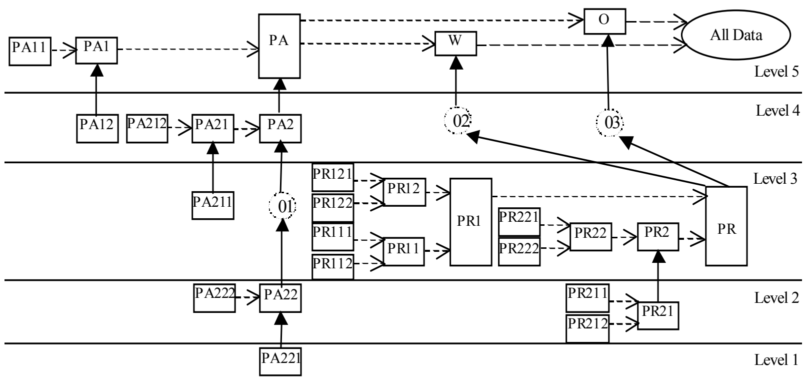

Fig. 5. The final DSH

In step 8, the security designer processes the above MRDs for each category and, according to the basic relationships of user roles, refines the organizational user roles NU and PS. The common parts (overlaps) of permissions of roles HN, NU and PS are represented in figure 6. The refinement of organizational user roles NU and PS results in five logical user roles: NU1, NU2, PS1, PS2 and NP3.

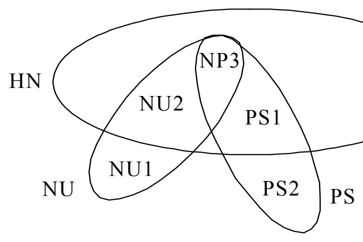

Fig. 6. Role relationships

As a result of the previous refinement, the security designer constructs an initial User Role Hierarchy (URH) without specifying any clearance levels (Figure 7). Senior (high level) user roles of the URH inherit permissions from their child nodes, e.g. user roles NU2 and PS1 inherit permissions from user role NP3.

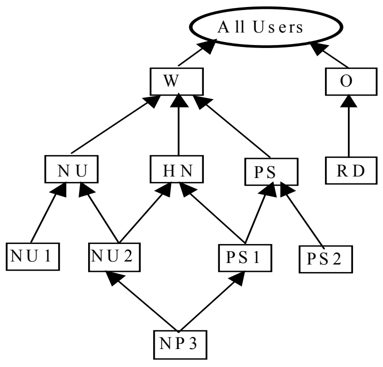

Fig. 7. URH without levels

In step 9, the initial permission set (set of MDRs) is updated in order to replace organizational user roles NU and PS with the new logical roles NU1, NU2, PS1, PS2 and NP3 (Table 6).

According to step 10, the automated labeling process is further applied to user roles and a new User Role Hierarchy (final URH) with clearance levels is produced, as shown in Figure 8.

(data set)	{roles}
PA12, PA212	{HN}
PA211	{HN, RD}
PA221	{NP3, RD}
PA222	{NP3}
PR111, PR121	{NU1, RD}
PR112, PR122	{PS1, RD}
PR211, PR221	{NU2, RD}
PR212, PR222	{PS2, RD}

Table 6. Final permission set

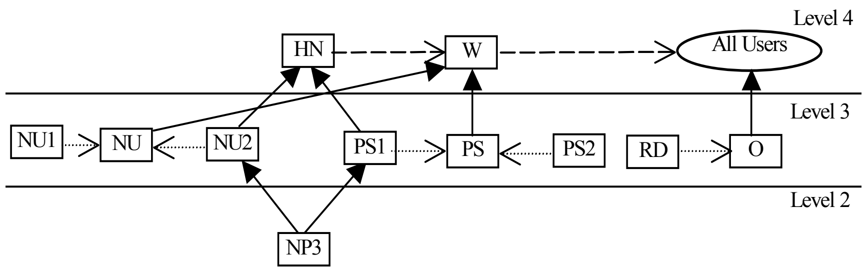

Fig. 8. The final URH

The application of the proposed methodology, as demonstrated in this implementation example, results in an access control system that consists of a set of user roles, a set of data sets, their abstraction structures (final user role hierarchy and data set hierarchy) with classification levels and categories and the authorizations of user roles to access data sets (final permission set).

5 Conclusion

In this paper we have demonstrated the applicability of a methodological approach for engineering role and lattice based mechanisms of a database access control system with a detailed implementation example based on the security requirements of a major Greek hospital. The whole process of applying the engineering methodology aims to transforming application and organization parameters into a set of authorization rules that can be implemented subsequently with the security mechanisms of the eMEDAC security model.

Applying the proposed methodology to a new or existing information system results in an access control system that allows separation of application and authorization logic, which makes application development, deployment and maintenance more cost-effective and easily enforces organizational policies across different applications. Hence, the resulting access control system can easily follow the evolution of the organization and adapt to the progress of the information system without affecting the primary application data, while maintaining a fine-grained level of granularity with multilevel access control at the same time.

References

1. R. Sandhu: Access Control: The Neglected Frontier. *Proceedings 1st Australian Conference on Information Security and Privacy*. Wollongong, Australia (1996)
2. S. Castano, M. Fugini, G. Martella, P. Samarati: *Database Security*, Addison-Wesley (1994)
3. R. Sandhu and P. Samarati: *Authentication, Access Control, and Intrusion Detection*. The Computer Science and Engineering Handbook (1997)
4. G. Pangalos and M. Khair: Design of a Secure Medical Database Systems. *Proceedings IFIP Conference on Information Systems Security - Facing the Information Society of the 21st Century*. (1996)
5. G. Pangalos: Secure Medical Databases. *Proceedings IMIA Security Conference*. Finland (1996)
6. I. Mavridis, G. Pangalos, and M. Khair: eMEDAC: Role-based Access Control Supporting Discretionary and Mandatory Features. *Proceedings 13th IFIP WG 11.3 Working Conference on Database Security*. Seattle WA, (1999)
7. D. Bell, and L. LaPadula: Secure Computer Systems: Unified Exposition and Multics Interpretation. Tech Report ESD-TR-75-306. MITRE Corporation (1976)
8. R. Sandhu: Role-Based Access Control. In: *Advances in Computers*, Vol.46. Academic Press (1998)
9. D. F. Ferraiolo, R. Sandhu , S. Gavrila , D. Richard Kuhn , R. Chandramouli: Proposed NIST Standard for Role-based Access Control. *ACM Transactions on Information and System Security,* 4(3), (2001)
10. I. Sommerville: *Software Engineering*. Addison-Wesley Publishing Company, (1996)
11. G. Pernull: Database Security. In: *Advances in Computers*, Vol.38. Academic Press (1994)
12. R. Sandhu: Relational Database Access Controls. In: *Handbook of Information Security Management* (1994-95 Yearbook). Auerbach Publishers (1994) 145–160
13. G. Neumann and M. Strembeck: A Scenario-driven Role Engineering Process for Functional RBAC Roles. *Proceedings SACMAT Conference*, Monterey, CA (2002)
14. S. Katsikas and D. Gritzalis: High Level Security Policy Guidelines. In: *Data Security for Health Care*, Vol.II Technical Guidelines. Ed. SEISMED Consortium. IOS Press (1996)
15. D.A. Marca and C.L. McGowan: *SADT: Structured Analysis and Design Technique*. McGraw-Hill, (1988)
16. B. Blobel, G. Bleumer, A. Muller, E. Flikkenschild, and F. Ottes: Current Security Issues Faced by Health Care Establishments. *Proceedings Conference on Implementing Secure Healthcare Telematics Applications in Europe (ISHTAR)* (1996)