A New Approach to Security System Development

Silvana Castano Giancarlo Martella Pierangela Samarati

Dipartimento di Scienze dell'Informazione
Università di Milano
Milano, Italy

Abstract

The development of a security system is generally performed through a multiphase methodology, starting from the initial preliminary analisys of the application environment, up to the physical implementation of the security mechanisms. In this framework, we propose a new approach for the development of security systems based on the reuse of existing security specifications. In the paper we illustrate how reusable specifications can be built by analyzing existing security systems, and how they can be used to develop new security systems not from scratch.

1 Introduction

Information is often of vital importance for the activity of an organization. As a matter of fact, the ability to always have at disposal precise and correct information represents an important strategic advantage over the competition. Consequently, protection of information against possible threats (e.g., unauthorized readings or modifications) is a crucial issue for an organization. Access control systems must be developed to ensure that only authorized users access information.

The development of an access control system is generally performed through a multiphase methodology. The phases of the methodology are as follows:

- *preliminary analysis*, constituted by a feasibility study of the security system;
- *requirement analysis*, which determines the needs of security of the system under consideration with respect to the vulnerability of the system and the risks to which the system is exposed;
- *security policy selection*, where the high-level access control guidelines are stated;
- *modeling*, where a formal model representing the policy is defined.

  The authorizations holding in the system can be represented in form of an authorization schema, where the elements of interests (e.g., subjects, objects, authorizations) are graphically depicted. In the following, we will use the terms authorization schema and security specification interchangeably;
- *implementation* and *verification* of the security system.

The use of a multiphase methodology has various benefits. First, it makes it possible to separate the design process into subtasks, thus allowing the developers to focus on particular security aspects in each task. Second, it allows one to distinguish between security policies (the high-level access control guidelines) and security mechanisms (meaning the low-level software/hardware functions implementing the control).

The use of a multiphase methodology requires a substantial amount of work. In particular, every time a new access control system must be developed all the phases previously listed must be followed, and the passage among the different phases controlled for correctness. To shorten the development process, and to reduce the amount of work in charge of security developers, we can observe that it is not always required to develop each new security system from scratch. Indeed, there are applications (e.g., `Project-Management`, `Personnel`, `Payroll`) in one or more organizations, that have similar security requirements. These security requirements, if identified and properly made available, can be re-used during the development of security systems for new applications.

In this paper we propose a reuse-based approach for the development of security systems in office information systems, following the recent proposals for information system and software development methodologies [2,6,9]. In particular, we illustrate techniques for constructing reusable security specifications, by analyzing existing security system specifications and identifying their commonalities.

The paper is organized as follows. Section 2 characterizes the environment of interest and discusses the problem of security specifications in such environment. Section 3 illustrates the reference authorization model used throughout the paper. Section 4 presents our approach for the construction of reusable security specifications. Section 5 discusses some research issues we are currently investigating and presents the conclusions.

2 Office environment

Many security models to ensure information security have been proposed and realized up to the present [4,10,12]. Most of them rise from either autocratic requirements (models for military security) or cooperation requirements (model developed for academic-like environment). The commercial and office environments need of 'ad-hoc' security models, which represent a midway point between the military model (based on a strict classification of users), and academic-like models (based on friendly cooperation), to assure a high protection level of information, while keeping necessary flexibility [5,13]. This is motivated by the characteristics proper of the office systems. An office can be viewed as a set of people, information sources and information manipulation tools sharing resources and objectives. Many factors make the office environment different from other systems (e.g., data processing), such as the following [21,22]:

- *Offices are social environments* in which every introduction of automatic instruments, procedural changes, or objective changes may be a cause of complication. Many systems, satisfactory from a technological point of view, have failed because of a too limited consideration of social factors.
- *Offices are dynamic systems:* changes are frequent in many areas. For example, absence from work of an employee may cause other employees to change their activities. Changes can also be caused by turn-over, promotions, competence changes, etc.
- *Office are concurrent, highly parallel, and asynchronous systems*: there are many people doing parallel activities at the same time.

From the security viewpoint, the following factors are peculiar to office systems:

- *Multiple applications:* office environments are characterized by multiple applications which serve multiple users with different purposes. In general, only some of them require protection and must be secured. These key applications must be identified before developing the security system.
- *Application-oriented security development:* in general, when developing the security system, security requirements of the applications to be secured are separately analyzed and modeled, and the development proceeds incrementally.

Recent researches focus on modeling security requirements by means of role-based models [3,8,14,18, 19]. A role can be defined as a set of actions and responsibilities associated with a particular working activity [1,11]. Then, in office information systems, users can be classified according to the activities they can execute [7]. A typical classification of office workers is based on three main roles: managers, executives, and employees. Starting from these, more specialized roles can be defined, related to more specific activities. For instance, the "programmer" role can be thought of as a specialization of the "employee" role.

In role-based access control (RBAC) models authorizations to access objects are granted to roles, not to the individual users. User access to documents is mediated by roles, played by the users, through which operations on objects can be fulfilled.

With reference to RBAC models, the initial phases of the security system development (namely, requirement analysis and modeling) require the analysis of the applications to be secured and the conceptual specification of the necessary roles and of the related authorizations. This process can be performed in the following steps:

1. *Analysis of the activities executed in the office:* this step produces the list of roles needed to execute the office tasks.
2. *Analysis of the operations each role has to execute on the documents* in order to fulfill its activities. This phase produces the authorizations that the roles must have on the objects in order to execute their tasks.
3. *Definition of activities that are incompatible:* this points out sets of activities which cannot be assigned to the same subject since this would give the subject too many privileges.
4. *Definition of an authorization schema, according to the RBAC:* the output of this step is an *authorization schema* where the allowed accesses are expressed in terms of roles, authorizations and documents featuring the system of interest, using the information produced by the previous steps.
5. *Analysis of tasks fulfilled by every user:* in this step a set of roles is assigned to every user. These roles identify the activities that every user can execute.

In this paper we propose a method for defining reusable specifications starting from a set of authorization schemas related to applications having similar security requirements. Reusable specifications are defined in form of generic roles, generic authorizations, and generic objects. These generic components are then adapted and specialized in order to define new authorization schema. The motivations behind our proposal are as follows.

- In the same organization, different applications may have a similar/common set of security requirements (i.e., access authorizations and roles). Then, the security specifications of an application can be produced starting from the security specifications of other applications.
- Different organizations are characterized by same or similar applications and, consequently,

by similar security specifications. The security specifications for a new application in an organization can then be derived from the security specifications of the same/similar applications in other organizations.

In the following, we illustrate how to identify commonalities between security specifications of existing applications in one or more organizations, and how to make them available in form of reusable specifications for the development of new similar applications. To this purpose, we will refer to a reference RBAC model, that will be illustrated in the next section.

3 A reference authorization model

In this section we illustrate the reference role-based authorization model that we will use throughout the paper.

In our reference model objects are classified according to their type (e.g., `letters`, `manuals`) or to their application area (e.g., `commercial letters`, `advertising letters`). This allows to specify authorizations on document classes to be considered applicable to all objects belonging to the class.

In RBAC two types of authorizations must be considered: for users to play roles (user authorizations) and for roles to exercise privileges on objects or roles (role authorizations). Privileges can be either elementary read/write operations as well as high-level actions.

A user authorization can be characterized as a pair of the form $\langle u, r \rangle$ stating that user u is allowed to play role r, and therefore to use all privileges specified for the role.

A role authorization can be either an authorization on objects or an authorization on roles. A role authorization on objects specifies that users playing the role can execute an action on some objects. A role authorization on objects can be characterized as triple of the form $\langle r, o, a \rangle$ stating that users playing role r can execute action a on objects of type o. For example, authorization $\langle$`manager,read,report`$\rangle$ states that a user playing the role of `manager` can read objects of type reports. A role authorization on another role specifies that the first role can execute some actions on the second role. In this case actions are generally administrative operations allowing the delegation of privileges from one role to the other. A role authorization on another role can be characterized as a triple of the form $\langle r_1, (o, a), r_2 \rangle$ stating that users playing role r_1 can grant and revoke role r_2 authorizations to execute action a on object o. For instance, authorization $\langle$`director,(read,letter),secretary`$\rangle$ states that a `director` can allow the `secretary` to read letters.

Starting from the triples of the model, an authorization schema can be defined. In the authorization schema, the roles and the documents are represented by boxes, and authorizations by edges, from the authorized role to the involved entity (either document or role). An example of authorization schema is illustrated in Figure 1. This schema refers to a `Report-Production` application, in the `Project Development` environment. Roles are represented as ovals with normal lines, while documents as ovals with bold lines. Authorizations are represented as edges connecting the involved roles and documents. Authorizations here specified state that the `Coordinator` writes the `Technical Report`, by collecting the information from the `Manuals` and from the `Technical Documentation` written by the `Programmers` and the `Analysits`, respectively. The `Project Manager` can read or approve the `Technical Report`. `Technical Reports` are filed by and the `Secretary`.

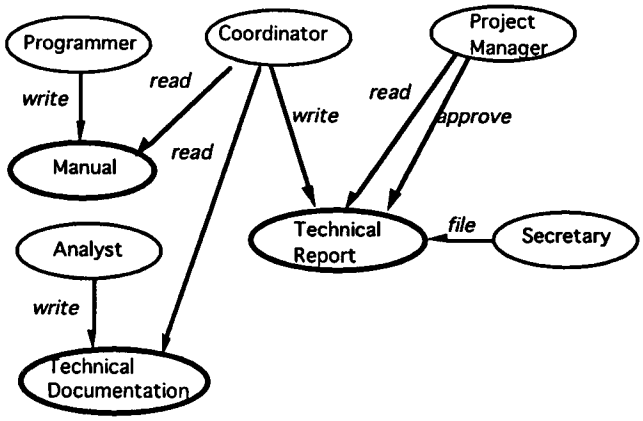

Figure 1: An example of authorization schema

In the following, we will refer to the roles and objects within an authorization schema as the *elements* of the authorization schema.

We do not make any assumptions on the types of objects or on the privileges executable on the objects. These choices depend on the specific application environment. Moreover, we do not put any restrictions on how roles are administered and used. For instance, we do not make any assumption on whether users are constrained to play one role at a time or may play multiple roles at the same time or how separation of duties is supported. These choices depend on the particular environment under consideration and any of them may be preferred over the other in particular situations. This generality makes our approach applicable to role-based models with different characteristics. Role policy issues do not affect our approach and can be considered as orthogonal to the problems of our concern in the paper.

4 Building reusable security specifications

In this section we illustrate our method for the construction of reusable specifications for security system development. Starting from properly selected authorization schemas, related to one or more applications in one or more organizations, common and similar security elements are identified, and their commonalities are factored out into reusable specifications. Reusable specifications are defined as generic autho-

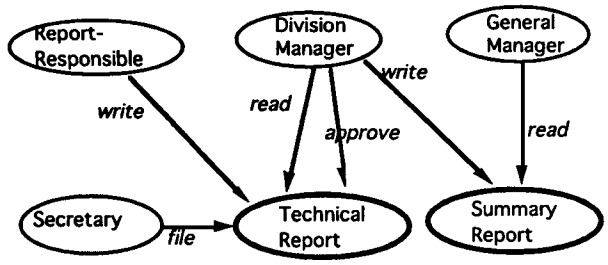

Figure 2: An example of authorization schema

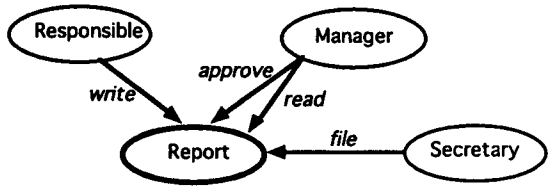

Figure 3: An example of generic authorization schema

rization schemas, composed of generic roles and objects, together with the corresponding authorizations.

To illustrate our goal, let us consider a second authorization schema related to the `Report-Production` application, shown in Figure 2, characterized by slightly different security requirements. Starting from this schema, and from the schema shown in Figure 1, we want to construct the generic schema for a generic `Report-Production` application. The security requirements of the generic schema are expressed in terms of generic roles, generic objects and authorizations, derived from the analysis of the roles and the objects of the specific authorization schemas. For instance, Figure 3 illustrates the generic schema derived from the authorization schema of Figures 1 and 2.

The method for the construction of reusable security specifications is composed of the following phases (see Figure 4):

1. selection of candidate authorization schemas;
2. selection and classification of elements of the authorization schemas;
3. design of reusable security specifications.

In the following subsections we describe more in details each phase of our method for constructing generic authorization schemas.

4.1 Selection of candidate authorization schemas

In this phase, candidate authorization schemas relevant for reuse are selected from a set of existing authorization schemas, related to one or more applications, in the domain of interest. Possible selection criteria concern the quality and relevance of the schemas, as follows:

- **Relevance of the application.** We are interested in having at disposal a set of candidate schemas that refer to key applications from security viewpoint, i.e., applications that always must be secured within organizations. This to assure that reusable specifications will be of real interest for new applications of that type. For example, the `Loan` application is a key application for organizations in the `Banking` domain, and thus it is reasonable to collect authorization schemas related to this application and to extract reusable components from them.
- **Quality and completeness of the authorization schemas.** Only authorization schemas modeling a significant number of roles and operations should be selected for a given type of application. This to restrict the analysis to those schemas really significant and reduce the reuse costs. This means that we are interested to authorization schemas that cover, possibly exhaustively, the main security requirements for the selected type of application.

The output of phase (1) is a set of candidate authorization schemas that verify previous criteria, related to one or more key applications, as they have been defined in their respective organizations. These schema will then be used in the subsequent phases to identify and extract reusable specifications. In our case, selected schemas correspond to those shown in Figure 1 and Figure 2.

4.2 Selection and classification of security specifications

For each key application, the set of collected candidate authorization schemas constitute the starting point for defining reusable specifications. To this purpose, we are interested in identifying and grouping together the elements (i.e., roles and objects) that present similarities in their respective authorization schemas. Similarity exists between roles (objects) in different authorization schemas if they describe entities that have a (possibly high) number of commonalities. Commonalities are determined on the basis of the characteristics of the elements within an authorization schema. In our reference model, we suppose that roles and object types are assigned a set of *attributes* that specify the properties of the entities they describe. For example, with reference to the schema in Figure 1, the role `Project Manager` can be defined with the attributes `department-code`, `salary`, `secretary`, `coord-projects`; the object `Technical Report` can have the attributes `author`, `approver`, `date-of-creation`, `date-of-approval`, `topic`, `project`. Moreover, in our authorization schemas, roles are related to objects/roles through links. Links describe the authorizations of the roles on the objects/roles. We refer to the set of links connecting a role R to other elements of the schemas as the *context* of R. The context of roles is important to determine the level of similarity between roles in authorization schemas. The similarity between elements

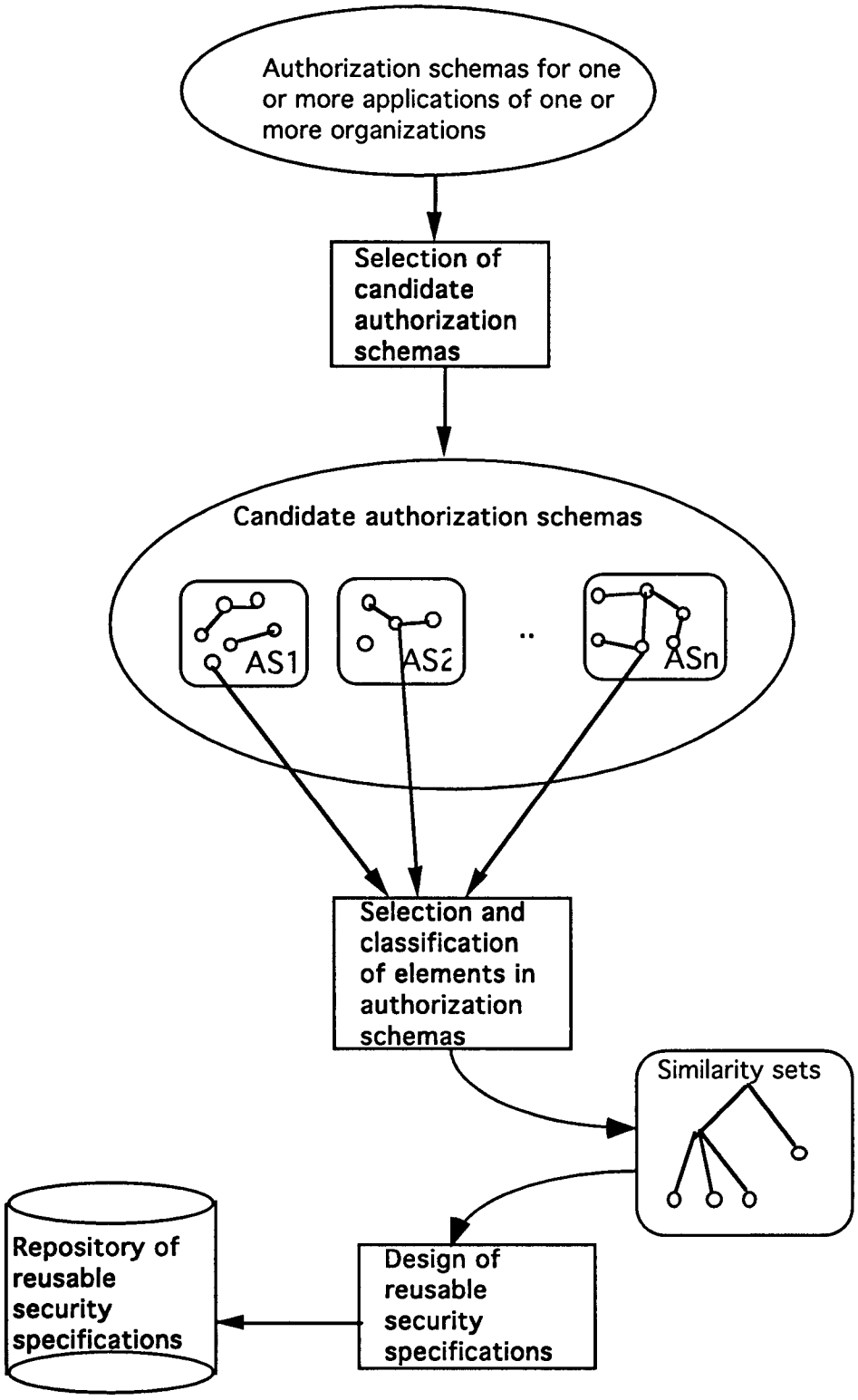

Figure 4: Construction of reusable security specifications

of authorization schemas is computed in the following way:

- **Similarity between objects**: it is computed on the basis of their names (*name similarity*) and of the number of shared attributes (*attribute similarity*). Given two objects O_1 and O_2, they are similar if their names are equal, or similar/synonyms, and if they share a (possibly high) number of attributes. For example, suppose that the object `Technical report` of Figure 1 has attributes `author`, `approver`, `date-of-creation`, `date-of-approval`, `topic`, `project` and that the object `Technical Report` of Figure 2 has attributes `author`, `approver`, `date-of-creation`, `date-of-approval`, `project`, `department`. Then, the objects `Technical Report` in schemas of Figure 1 and in Figure 2 are similar, since they have the same name and share several attributes.

- **Similarity between roles**: it is computed on the basis of their names (*name similarity*), of the number of shared attributes (*attribute similarity*), and of the number of shared authorizations (*context similarity*). Shared authorizations are authorizations defined with the same/similar names, on objects that are similar to each other. With reference to the schemas shown in Figure 1 and in Figure 2, the roles `Project Manager` and the role `Division Manager` are similar, since they have similar names, share some attributes, and share the authorizations `read` and `approve` on the `Technical Report` objects.

Since we consider names (element names, attribute names, authorization names) for similarity computation, we rely on the availability of a thesaurus for maintaining the most frequently used and the most significant names usually assigned to elements in authorization schemas. Moreover, the relationships *similar-to* and *synonym-of* are maintained between names within the thesaurus, with a numerical value that express the *conceptual distance* between two names. The thesaurus is partitioned by applications, to facilitate the analysis of authorization schemas. The metrics we use for computing element similarity are: the conceptual distance of the thesaurus for name similarity, and the Dice's metric [15], for attribute and context similarity. These metrics return value 1 for elements that are identical, value 0 for elements that don't have similarity at all, and a value in the range $(0, \ldots, 1)$ for elements that have some level of similarity. For each pair of elements (e_i, e_j) belonging to different authorization schemas, the name similarity, attribute similarity and context similarity values are summed up to determine the global level of similarity of the considered elements.

On the basis of the similarity levels, roles and objects are grouped into *similarity sets*, that constitute the starting point for the extraction of generic roles and generic objects, performed in the next phase. Examples of similarity sets for the schemas in Figure 1 and Figure 2 are:
{`Project Manager`, `Division Manager`},
{`Technical Report`, `Technical Report`},
{`Secretary`, `Secretary`}, and
{`Coordinator`, `Report-Responsible`}.

4.3 Design of reusable security specifications

Reusable specifications are defined in form of generic authorization schemas, composed of generic roles and documents, with associated authorizations. Moreover, we associate a set of *guidelines* with each generic authorization schema. Guidelines provide suggestions for adapting generic roles and documents to particular applications. Starting from the proposed similarity sets, generic roles, and objects are extracted, factoring out their commonalities (i.e., attributes and authorizations). For instance, Figure 3 illustrates the generic schema derived from the authorization schema of Figures 1 and 2.

With each generic element, a set of guidelines is associated, that express how to specialize/enrich the generic element with attributes and/or authorizations more specific, related to particular applications of the type under consideration.

5 Concluding remarks and research issues

In this paper we have proposed a new approach to security system development, based on reuse of security specifications. We have presented a reference model based on the concept of role for modeling the authorization requirements of office information systems, and we used authorization schemas defined according to this model for constructing generic security specifications. Authorization schemas of new applications are then defined by reusing and refining such generic specifications, instead of starting from scratch every time. The main advantage of our approach is related to the improvement of both the development process, by shortening the system life cycle, and the management of existing security systems.

The work presented in this paper is currently being carried out by the authors and some issues are being investigated. A first issue concerns the applicability of the approach to different models. Up to now, we considered our reference model for presenting the approach. To make the approach wide applicable, we are interested in considering different authorization models, and in mapping them into the elements of our reference model. A further issues concerns the type of authorizations in the reference model. In this paper we only considered the authorizations of roles on objects. The reuse approach needs refinements to properly deal with authorizations of roles on roles. For this purpose, similarity criteria and metrics should be defined. Another issue under investigation concerns the use of an object-oriented model to describe the properties and the specialization/generalization relationships typical of roles and objects in the reference model. The use of an object-oriented model can facilitate also the definition of reusable specification. A

further issue concerns the definition of reuse guidelines associated with elements of a generic authorization schema. In particular guidelines must be defined which embed the knowledge of developers about the main security requirements of different applications of a given type.

References

[1] R.W. Baldwin, "Naming and grouping privileges to simplify security management in large databases," *Proc. of the IEEE Symposium in Security and Privacy,* Oakland, April 1990.

[2] D.Batory, S.O'Malley, "The Design and Implementation of Hierarchical Software Systems with Reusable Components," *ACM TOSEM,* Vol.1, N.4, October 1992, pp.355-398.

[3] S. Castano and P. Samarati, "An object- oriented security model for office environment," *Proc. of the 1992 IEEE International Carnahan Conf. on Security Technology,* Canada, October 1992.

[4] S. Castano, M.G. Fugini, G. Martella, and P. Samarati, *Database Security*, Addison-Wesley, 1994.

[5] D.D. Clark and D.R. Wilson, "A comparison of commercial and military security policies," *Proc. of the IEEE Symposium in Security and Privacy,* Oakland, April 1987.

[6] V. De Antonellis, S. Castano, and L.Vandoni, "Building reusable Components Through Project Evolution Analysis," *Information Systems*, Vol. 19, No. 3, April 1994.

[7] J.E. Dobson and J.A. McDermit, "Security models and enterprise models," *Database Security, II: Status and Prospects,* C. Landwehr (ed.), North-Holland, 1989.

[8] D. Ferraiolo and R. Kuhn, "Role-based access controls," *15th NCSC National Computer Security Conference*, Baltimore, MD, October 1992.

[9] C.W. Krueger, "Software Reuse," *ACM Computing Surveys,* Vol.24, No.2, June 1992, pp.131-183.

[10] C.E. Landwehr, "Formal models for computer security," *Computer Surveys*, vol.13, no.3, Sept. 1981.

[11] F.H. Lochovsky and C.C. Woo, "Role-based security in database management systems," *Database Security: Status and Prospects,* C. Landwehr (eds.), North-Holland, 1988.

[12] J.K. Millen and C.M. Cerniglia, "Computer security models," *Techn. Report The MITRE Corporation*, 1984

[13] J.D. Moffet and M.S. Sloman, "The source of authority for commercial access control," *Computer*, Febr. 1988

[14] M. Nyanchama and S. Osborn, "Role-based security: pros, cons, & some research directions," *ACM SIGSAC Review,* Vol. 11, No. 2, Spring 1993.

[15] G. Salton, *Automatic text processing - The transformation, analysis and retrieval of information by computer,* Addison-Wesley, 1989.

[16] R. Sandhu and P. Samarati "Access control: principles and practice," *IEEE Communications,* September 1994.

[17] R. Sandhu, "Separation of duties in computerized information systems," *Database Security, IV: Status and Prospects,* S. Jajodia and C. Landwehr (eds.), North-Holland, 1991.

[18] R. Sandhu, "Transaction control expressions for separation of duties," *Proc. of the IEEE Symposium in Security and Privacy,* Oakland, April 1988.

[19] G. Steinke and M. Jarke, "Support for security modeling in information systems design," *Database Security, VI: Status and Prospects,* B. Thurainsingham and C. Landwehr (eds.), North-Holland, 1993.

[20] D.J. Thomsen, "Role-based application design and enforcement," *Database Security, IV: Status and Prospects,* S. Jajodia and C.E. Landwehr (eds.), North-Holland, 1991.

[21] D. Tsichritzis, *Office automation,* D. Tsichritzis ed., Springer-Verlag 1984.

[22] S.T. Zdonik, "An object management for office applications," in *Languages for Automations",* Shi-Kuo-Chang (ed.), 1985.