Security, Authorization, and Authentication for Enterprise Computing

CSE Technical Report CSE TR-03-99

Prof. Steven A. Demurjian
Dept. of Computer Science & Engineering
The University of Connecticut
Storrs, CT 06269-3155

Abstract

Enterprise computing (EC) is the recognition that to effectively utilize and disseminate information within an entity (university, corporation, government agency, etc.) it will be necessary to design and develop integrated distributed computing environments that allow all types of existing and future systems to interoperate. In EC, there are legacy, COTS, database, and new client/server applications that all must interact to facilitate both peace and war time communications among personnel at varying locations (e.g., divisions, battalions, brigades, companies, platoons, Pentagon, etc.). When all of the diverse components that comprise EC applications (ECA) are linked, then security becomes an important issue, to insure that individuals only have access to the right information at the appropriate time. This paper details security, authorization, and authentication for enterprise computing. The paper's intent is to begin to answer the important questions: What are major and underlying security concepts for enterprise computing? What are the security requirements for ECA? What are available approaches to security for enterprise computing and its applications?

1. Introduction

Security issues in distributed computing environments are difficult to address, given the diverseness of the clients, servers, databases, legacy, and COTS applications that must be integrated. While individual databases, legacy, and COTS may have their own security policies and mechanisms, in a distributed environment, security must be designed and developed across the Internet and intranets. In a client/server environment, security policies and mechanisms must be designed to support:

- *authentication* - Is the client who s/he says they are?
- *authorization* - Does the client have permission to do what s/he wants to do?
- *privacy* - Is anyone intercepting client/server communications?

The privacy issue is particularly important when clients and servers are separated by a network, especially when the client and server are outside each other's security

perimeters. Two other accompanying concepts are the security policy and enforcement mechanism. The *security policy* represents the set of security requirements for a particular application, and is initially defined when the specification is written. The security policy contains the identification of the critical security issues for an application, the access control approaches (MAC, DAC, URBS) that will be utilized, the definition of the data that needs to be protected, and the anticipated users and their privileges against the application. Once defined, the security policy must be captured and followed at application runtime. This is accomplished via an *enforcement mechanism* which represents the set of centralized and distributed software to insure that the security policy is maintained and never violated.

Security policies and enforcement mechanisms must have *high constancy* in order to support a *high assurance*, secure system. The consistency must be maintained at all levels within the policy, to insure the constantly evolving security policy is always maintained. Consistency is the foundation upon which high integrity and assured secure systems must be built. From an assurance perspective, it is critical that the security privileges for each client are adequate to support his/her activities. Moreover, these security privileges must meet but not exceed the client's capabilities within the application. From a consistency perspective, it is critical that the defined security privileges for each client are internally consistent. This requires the attainment of the *least-privilege principle* which grants only necessary access privileges but no more. For example, a pharmacist can read a prescription for a patient, is able to modify the refills field, and can submit a claim for payment to an insurance company. In addition, the defined security privileges for the various related clients that span an application must be globally consistent. This is achieved by *mutual exclusion conditions*, where there must be a careful balance between permitting access to certain objects while simultaneously prohibiting access to other, special objects. For example, a pharmacist is also explicitly prohibited from modifying the prescription. This strong mutual exclusion situation is clearly observed by the medical profession and is mandated by law.

In general, security policies are application dependent, and consequently, data security requirements vary widely from application to application. For example, sensitive health care data must be both protected from use while simultaneously be almost instantaneously available in emergency and life critical situations. On the other hand, in some design environments such as CAD, the most up-to-date specifications on mechanical parts must be available in a shared manner to promote cooperation and facilitate productivity. In this case, the security policies aren't as life-critical, but may be just as important from a business perspective. The ultimate responsibility for any security policy is on the shoulders of the application's management personnel and organization's data security officer.

Research and development for access control in databases has traditionally taken an approach based on security clearance [Denn86, Ditt89, Keef88, Land84, Lunt90, Roug87, Stac90, Thur89] using the Bell and Lapadula security model [Bell75]. These multi-level secure approaches support *mandatory access control (MAC)* to classify and tag data with relevant security levels. Users are then given a security level, and their clearance level dictates their access capabilities. To complement MAC*, discretionary access control*

(DAC) has been proposed [Loch88, Sand96, Spoo89], to allow security emphasis to focus on the responsibilities of end users with respect to their needs against the application when establishing access rights. DAC is intended to provide discretion to allow the security policy to be customizable and restrictable on a individual-by-individual basis through the consideration of the user role or responsibilities that are needed against an application. Hence, *user-role based security (URBS)* has been proposed [Ting88] as one approach to address and realize DAC. URBS is not intended to be less secure than MAC, rather, it is intended to facilitate user interactions while simultaneously protecting sensitive data. In some ECA, it may be relevant to combine both MAC and DAC, to achieve the best of both worlds.

The remainder of this paper is organized into three sections and a conclusion. In Section 2, we examine security, authorization, and authentication for EC by identifying, delineating, and explaining the key security requirements. The presentation in this section is in four categories: Information Access and Control, Security Handlers and Processing, and Needs of Legacy/COTS Applications. In Section 3, we investigate the important issue of security for legacy/COTS applications. One of the critical needs for ECA is the ability to interoperate new applications with existing legacy/COTS applications in a distributed computing environment. We consider the application of the approaches presented in Section 3 as well as emerging technologies and their support for security for legacy/COTS. Note that the work in Section 3 is very preliminary in nature. Finally, in Section 4, we conclude this paper and outline potential future research directions.

2. Security Requirements for Distributed Applications

Two key concepts for the development of secure systems are the security policy and enforcement mechanism. The *security policy* represents the set of security requirements for a particular application, and is initially defined when its specification is written. The security policy contains the identification of the critical security issues for an application, the access control approaches (mandatory access control (MAC) [Keef88, Land84], discretionary access control (DAC) [Loch88, Sand96], or user role-based security (URBS) [Demu97] models) that will be utilized, the definition of the data that needs to be protected, and the anticipated users and their privileges against the application. Once defined, the security policy must be captured and followed at application runtime via an *enforcement mechanism* which represents the set of centralized and distributed software to insure that the security policy is maintained and never violated.

In general, security policies are application dependent, and consequently, data security requirements vary widely from application to application. For example, sensitive health care data must be both protected from misuse, while being instantaneously available in life critical situations. In manufacturing/CAD applications, the most up-to-date specifications on mechanical parts must be available in a shared manner to promote cooperation and facilitate productivity. In distributed applications that involve hundreds of interacting systems and over 10,000 active nodes, the definition, planning, management, and attainment of the security policy is a major and complex task.

In order to begin to address security, authorization, and authentication for distributed computing, the critical first step is to identify, delineate, and explain the key security requirements. This is accomplished by providing a set of motivation questions that are categorized as follows:

- **Information Access and Control**: These questions focus on the security requirements from the perspective of the users of distributed computing applications.
- **Security Handlers and Processing**: These questions focus on the handling of security during the lifetime of the application.
- **Needs of Legacy/COTS Applications**: These questions focus on the fact that legacy and existing/future COTS applications will all be part of distributed computing applications.

In the remainder of this section, we present and discuss each of the three categories of questions. We do not intend to provide answers to the questions that are presented; rather, the questions serve as a means to begin to enumerate and quantify the relevant issues that must be addressed for security, authorization, and authentication of distributed computing applications. We also briefly review ongoing and future work as it relates to security for distributed computing.

2.1 Information Access and Flow

User security privileges are a key concern when defining the security policy for distributed computing. The types of users of distributed applications need different types of information (raw, processed, aggregated) at different times based on their needs potentially dynamically changing needs. The questions related to information access and flow are presented below. The questions differ as follows: the first involves security requirements that would be spelled out in the policy, which is more static; the second deals with information that should be passed to users in normal operating situations; and the third concerns information that must be available on demand in dynamic situations.

- W*ho can see what Information at what Time?*
  - What are the Security Requirements for Different Users?
- *What Information Needs to be Sent to at what Time?*
  - What Routine Information is Forwarded to Different Users at Regular Intervals?
- *What Information Needs to be Available to at what Time?*
  - What Information will Different Users Request in Time-Critical Situations?

Note that in the case of the second and third questions, security must also consider the situation where it is important to verify that the user receiving data is the one for whom

the data is intended, to insure that an individual with contrary or hostile intent has not taken over a workstation.

2.2 Security Handlers/Processing

A security handler is a piece of software that is responsible for managing some portion of an application's security policy. Once the security requirements and policy have been determined (as discussed in Section 2.1), one must then consider the steps, approaches, and techniques that are necessary to maintain and enforce that policy in a dynamic, distributed, and interoperative environment via security handlers that interact across the distributed environment. Historically, security has been managed at a centralized level, with a common system providing access to a shared repository of information. However, in the client/server, distributed computing environment, such an approach will need to be expanded and evolved to meet more complicated and diverse requirements.

As we transition from the security requirements and policy (Section 2.1) to the maintenance and enforcement at runtime, it is necessary to define and develop various security techniques to insure that the right information is getting to the right users at the right time. These next three questions involve this issue. First, users must be sure that the information they are receiving is correct, accurate, and timely. Second, in situations of limited bandwidth, to prevent information overload, or to limit what a minimally authorized user can see, to pass only just enough information along. Third, in some situations, authorized users must be able to circumvent the prescribed or default limits on information availability, to request and receive larger volumes of authorized information.

- *What Security Techniques are Needed to Insure that the Correct Information is Sent to the Appropriate Users at Right Time?*
- *What Security Techniques are Necessary to Insure that Exactly Enough Information and No More is Available to Appropriate Users at Optimal Times?*
- *What Security Techniques are Required to Allow as Much Information as Possible to be Available on Demand to Authorized Users?*

The security techniques that are required to support these three questions must be explored in detail in order for distributed applications to be successful from a security perspective.

The security techniques that are developed for the previous group of questions must then be organized into a set of cooperating and interacting security handlers that are present at all levels of the command hierarchy. These handlers are responsible for insuring that the policy is enforced in the runtime environment. Clearly, the degree of interoperability, distribution, and interaction, with real-time and distributed requirements in some domains represents a complexity that hasn't been addressed in security for centralized computing.

- *Are there Customized Security Handlers at Different Levels of Organizational Hierarchy?*
- *What is the Impact of Distribution on Security Policy Definition and Enforcement?*

High-technology firms (computing, manufacturing, etc.), banking and investment companies, hospitals and medical centers, and so on, are all attempting to address issues related to distributed computing. Security for such a distributed, interoperable environment has only been minimally considered and must be the focus of active research and problem solving in the coming years.

2.3 Legacy/COTS Applications

The integrated, interoperative distributed application will be composed of new and existing software. Custom new software, proven legacy systems, and new and future COTS applications must all interact in order for information to be utilized in innovative ways. In the case of legacy and COTS, the level of support for security that they offer must be considered. The important point to remember, is that security may not be addressed within a legacy/COTS application. Of course, if a legacy/COTS is a database application, it is likely that security can be addressed via the primitives of the underlying database system. However, what happens when the legacy/COTS application has a programming interface that consists of a set of procedure, function, and/or method calls? Thus, the first question given below characterizes the issue from a broad perspective when attempting to support security for legacy/COTS. The second question involves the need to consider security when legacy/COTS applications are wrapped with Java to achieve a common framework for exchanging information among multiple interoperating systems, as indicated in the second question.

- *When Legacy/COTS Applications are Placed into Distributed, Interoperable Environment:*
  - Must Secure Access be Addressed?
  - How is Security Added to them if its not Present?
  - What Techniques are Needed to Control Access to them?
- *If Legacy/COTS Applications are Wrapped with a Java Application, how is Security Authorization and Enforcement Integrated?*

For the second question, one must examine the various security capabilities of Java to see whether they can be exploited to support security in legacy/COTS. The integration of security into a distributed computing environment that allows legacy/COTS applications to be managed and controlled is an important problem to be solved.

3. Security for Legacy/COTS

In enterprise computing, the interoperability of existing legacy/COTS applications with new client/server applications in the distributed computing environment is one of the major concerns. Security must be incorporated into the enterprise computing environment, and at the system level, this involves understanding the way that security can be handled by existing legacy/COTS applications. There are a wide range of possibilities when one attempts to characterize legacy/COTS applications. First, a legacy/COTS database application, running on a commercial database platform, may

already offer security for data as supported by the underlying database system. However, this may or may not be sufficient. Second, for a legacy/COTS application that is object-oriented, it may be possible to utilize one or more of the approaches given in [Demu98a] to support DAC/URBS. Third, for non-OO legacy/COTS applications, other techniques may need to be considered to incorporate security. Fourth, as Java becomes increasingly utilized, it will be necessary to support new Java client/server applications in the enterprise computing venue. Fifth, the growing interest in CORBA/ORBs will also bring another technology into the mix when considering security. It is important to realize that legacy/COTS applications were often originally constructed to be utilized in a standalone mode, with minimal interoperability with other systems. Thus, security may not have been considered, or may only work within the confines of the legacy/COTS applications.

The purpose of this section is to explore some preliminary ideas on select issues related to security for legacy/COTS. Specifically, we will consider schemes for supporting security for OO and non-OO legacy/COTS applications. Since the ideas presented herein are preliminary in nature, we make some assumptions about the composition of a general legacy/COTS application. First, we consider security for object-oriented legacy/COTS applications. In these applications, there is a high likelihood that a class-based programming interface is available, i.e., a set of classes that provide the means for application developers to write tools to integrate with legacy/COTS application. Based on this assumption, we propose two alternatives for integrating DAC/URBS for the user-role subclassing approach and basic exception approach as shown in Figure 1.

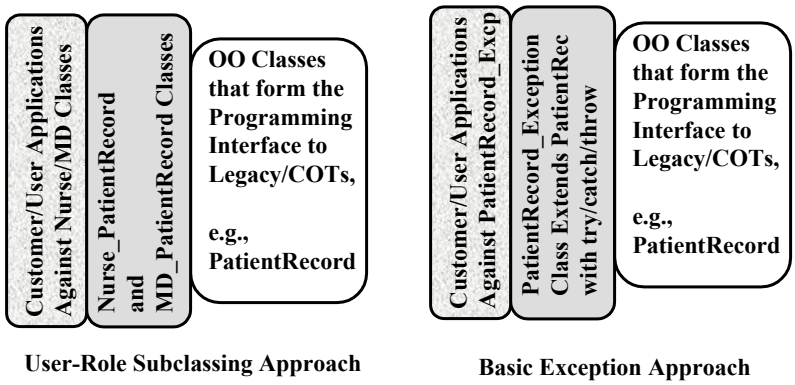

Figure 1: Two Alternatives for DAC/URBS for OO Legacy/COTS Applications.

Notice that in Figure 1, for the user-role subclassing alternative, the OO classes that form the programming interface would be analogous to the `PatientRecord` class from earlier

examples. If that is the case, then a layer of subclasses can be added for the different user roles that require access to the legacy/COTS classes, e.g., **`Nurse_PatientRecord`** and **`MD_PatientRecord`** for the Nurse and MD user roles. Then application developers will design and prototype tools against these user role subclasses, which in turn will interact with the underlying legacy/COTS superclasses. For the basic exception alternative also given in Figure 1, a similar solution is envisioned. A layer of classes that are subclasses of legacy/COTS classes are added to include the exception handling code. The **`PatientRecord_Exception`** class and the other **`Exception`** classes that correspond to legacy/COTS classes would then be utilized by application developers. For both alternatives, a security layer can be provided which is similar in concept to a wrapper.

Second, it is expected that many legacy/COTS applications will be non-object-oriented, developed in Ada83, C, Fortran, and so on. In this case, we can make the general assumption that a programming interface to the non-OO legacy/COTS application is available. This programming interface would consist of a set of procedure and functions. Based on this assumption, we propose two alternatives for integrating DAC/URBS for the user-role subclassing approach and basic exception approach, as shown in Figure 2.

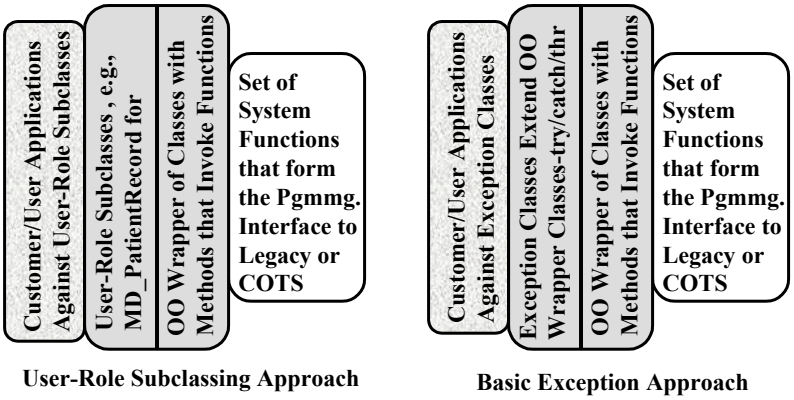

Figure 2: Two Alternatives for DAC/URBS for non-OO Legacy/COTS Applications.

In Figure 2, the user-role subclassing alternative is realized by placing an object-oriented wrapper around the non-OO legacy/COTS application. This effectively takes the procedures and functions that comprise the non-OO programming interface, and upgrades and organizes them as a set of object-oriented classes. Once this has been done, the remainder of the alternative functions as the OO case given in Figure 1, by defining user role subclasses and having application developers write tools against these subclasses.

The second alternative, for the basic exception approach, would also utilize a wrapper to upgrade the non-OO legacy/COTS application to an object-oriented context, where it could then be handled as presented for the OO case in Figure 1.

Of course, the approaches presented herein and shown in Figures 1 and 2 are preliminary in nature, and represent only two of the possible solution alternatives. It is important that we also consider other alternatives that may include Java, CORBA, or other emerging technologies. In addition, legacy/COTS database applications are strongly related to the above alternatives, since many commercial database platforms have programming interfaces. Thus, it may be possible to handle legacy/COTS database applications in a similar fashion.

4. Concluding Remarks and Future Work

In this paper has explored security, authorization, and authentication for enterprise computing environments. We have utilized a question-based approach to discerning, delineating, and exploring security for enterprise computing. The purpose of these questions was to motivate with the intention to extract and define the security requirements for enterprise computing. The questions were categorized as pertaining to: information access and flow, security handlers and processing, needs of legacy/COTS applications, and the role of MAC/DAC/URBS. Overall, we believe that these questions establish the context for future security work in enterprise computing.

Software architects and system designers must be aware of potential solutions that are appearing on the horizon in support of security for distributed computing applications. One effort has proposed alternatives for integrating security for legacy/COTS applications, which will be critical for distributed computing [Demu98]. Another effort has examined the potential impact of Java on security for distributed computing, through an investigation of the applicability of Java security primitives, in particular, the sandbox, security managers, and the access control list (ACL) [Smar98]. Other efforts have investigated security for distributed objects [Hale96] and software agents for the design and implementation of distributed security policies [Tari98]. For security in distributed computing to succeed, it is important to leverage existing security techniques in conjunction with emerging approaches to arrive at a solution for the 21st century.

References

[Bell75] D. Bell and L. LaPadula, “Secure Computer Systems: Unified Exposition and Multics Interpretation”, Technical Report MTIS AD-A023588, The MITRE Corporation, July 1975.

[Demu93a] S. Demurjian and T.C. Ting, “Shouldn't the Object-Oriented Paradigm Influence and Guide the Approach for Security?”, *Proc. of 1993 Workshop on Security for Object-Oriented Systems*, part of OOPSLA 1993, Sept. 1993.

[Demu93b] S. Demurjian, M.-Y. Hu, T.C. Ting, and D. Kleinman, "Towards an Authorization Mechanism for User-Role Based Security in an Object-Oriented Design Model", *Proc. of 1993 Phoenix Conf. on Computers and Communications*, Scottsdale, AZ, March 1993.

[Demu94a] S. Demurjian and T.C. Ting, "The Factors that Influence Apropos Security Approaches for the Object-Oriented Paradigm", *Workshops in Computing*, Springer-Verlag, 1994.

[Demu95a] S. Demurjian, T. Daggett, T.C. Ting, and M.-Y. Hu, "URBS Enforcement Mechanisms for Object-Oriented Systems and Applications", in *Database Security, IX: Status and Prospects*, D. Spooner, S. Demurjian, and J. Dobson (eds.), Chapman Hall, 1995.

[Demu95b] S. Demurjian, M.-Y. Hu, and T.C. Ting, "Role-Based Access Control for Object-Oriented/C++ Systems", *Proc. of First ACM Workshop on Role-Based Access Control*, Gaithersburg, MD, November 1995.

[Demu96a] S. Demurjian, T.C. Ting, and M.-Y. Hu, "Security for Object-Oriented Databases, Systems, and Applications", in *Progress in Object-Oriented Databases*, J. Prater (ed.), Ablex, 1997.

[Demu96b] S. Demurjian, T.C. Ting, M. Price, and M.-Y. Hu, "Extensible and Reusable Role-Based Object-Oriented Security", in *Database Security, X: Status and Prospects*, D. Spooner, P. Samarati, and R. Sandhu (eds.), Chapman Hall, 1997.

[Demu97a] S. Demurjian, T.C. Ting, and J. Reisner, "Software Architectural Alternatives for User Role-Based Security Policies", in *Database Security, XI: Status and Prospects*, T.Y. Lin and X. Qian (eds.), Chapman Hall, 1998.

[Demu98a] S. Demurjian and T.C. Ting, "Towards a Definitive Paradigm for Security in Object-Oriented Systems and Applications", accepted; to appear in *Journal of Compuer Security*, 1998.

[Denn86] D. Denning, et al., "Views for Multilevel Database Security", *Proc. of IEEE 1986 Sym. on Security and Privacy*, May 1986.

[Ditt89] K. Dittrich, et al., "Discretionary Access Control in Structurally Object-Oriented Systems", in *Database Security, II: Status and Prospects*, C. Landwehr (ed.), North-Holland, 1989.

[Elli94] H.J.C. Ellis, "An Information Engineering Approach to Unified Object-Oriented Design and Analyses", Ph.D. Degree Dissertation, The University of Connecticut, May 1994.

[Hale96] J. Hale, et al., "A Framework for High Assurance Security of Distributed Objects", *Proc. of 10th IFIP WG11.3 Working Conf. on Database Security*, Cumo, Italy, July 1996.

[Hu93a] M.-Y. Hu, S. Demurjian, and T.C. Ting, "User-Role Based Security Profiles for an Object-Oriented Design Model", in *Database Security, VI: Status and Prospects*, C. Landwehr and B. Thuraisingham (eds.), North-Holland, 1993.

[Hu93b] M.-Y. Hu, "Definition, Analyses, and Enforcement of User-Role Based Security in an Object-Oriented Design Model", Ph.D. Degree Dissertation, The University of Connecticut, May 1993.

[Keef88] T. Keefe, et al., "A Multilevel Security Model for Object-Oriented Systems", *Proc. of 11th Natl. Computer Security Conf.*, Oct. 1988.

[Land84] C. Landwehr, et al., "A Security Model for Military Message Systems"*, ACM Trans. on Computer Systems*, Vol. 2, No. 3, Sept. 1984.

[Loch88] F. H. Lochovsky and C. C. Woo, "Role-Based Security in Data Base Management Systems", in *Database Security: Status and Prospects*, C. Landwehr (ed.), North-Holland, 1988.

[Lunt90] T. Lunt and D. Hsieh, "The SeaView Secure Database System: A Progress Report", *Proc. of 1990 European Sym. on Research in Computer Security*, Oct. 1990.

[Need96] D. Needham, S. Demurjian, K. El Guemhioui, T. Peters, P. Zemani, M. McMahon, H. Ellis " ADAM: A Language-Independent, Object-Oriented, Design Environment for Modeling Inheritance and Relationship Variants in Ada 95, C++, and Eiffel", *Proc. of 1996 TriAda Conf., Philadelphia*, PA, December 1996.

[Roug87] P. Rougeau and Stearns, "The Sybase Secure Databse Server: A Solution to the Multilevel Secure DBMS Problem", *Proc. of 10th Natl. Computer Security Conf.*, Oct. 1987.

[Sand96] R. Sandhu, et al., "Role-Based Access Control Models", *IEEE Computer*, Vol. 29, No. 2, Feb. 1996.

[Shaw96] M. Shaw and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*, Prentice-Hall, 1996.

[Spoo89] D. Spooner, "The Impact of Inheritance on Security in Object-Oriented Database Systems", in *Database Security, II: Status and Prospects*, C. Landwehr (ed.), North-Holland, 1989.

[Stac90] P. Stachour and B. Thuraisingham, "Design of LDV: A Multilevel Secure Relational Database Management System", *IEEE Trans. on Knowledge and Data Engineering*, Vol. 2, No. 2, June 1990.

[Thur89] B. Thuraisingham, "Mandatory Security in Object-Oriented Database Systems", *Proc. of 1989 OOPSLA Conf*., Oct. 1989.

[Ting88] T.C. Ting, "A User-Role Based Data Security Approach", in *Database Security: Status and Prospects*, C. Landwehr (ed.), North-Holland, 1988.