IMPROVING SOFTWARE SECURITY THROUGH AN INTEGRATED APPROACH

Zaobin Gan and Dengwei Wei
Huazhong University of Science and Technology
Wuhan 430074,PR China

Vijay Varadharajan
Department of Computing, Macquarie University
Sydney, NSW 2109, Australia

Keywords: System Integration, RBAC.

Abstract: It has been recognized that the main source of problems with application software security is in most cases that the software is poorly designed and developed with respect to authentication and authorization. Aiming at preventing the security issues in the course of software design and development, this paper presents a framework for integrating a security policy specification with a system function integration. On the basis of the Role-Based Access Control (RBAC) model, this framework moves the responsibility of security through a central authorization management mechanism, Single Sign-On (SSO) access and integration management of security resources. The design can integrate the enterprise's multiple new, developing and existing application systems, and provide end users access these systems as a single system. An application instance of the framework is given in a large-sized enterprise information integrated system as well. The results show that the framework may provide enterprises with uniform and robust enforcement policies to improve the security of sensitive information systems.

1 MOTIVATION

In order to compete and survive in the information society, many enterprises have begun to set up some information management systems, such as accounting information systems, human resource management systems, product management systems, time-card management systems, and so on, in the past several years. Different systems, autonomously developed and managed, take advantage of different security policies and impose different constraints to be satisfied before allowing participants access to data. These systems, to a large extent, depend on the proper management of security resources, such as user accounts, passwords, logs, etc., in an enterprise. For example, user authentication with account and password is dependent on how well passwords are chosen, stored and managed. Relying on each application to perform authentication leads to networks with multiple disparate authentication policy enforcements based on application implementation. Users will need to remember and manage multiple accounts and passwords, and have to log in multiple times to access different application systems on the enterprise network. In addition, these systems do not provide the enterprise with a central point of control over authentication policies and mechanisms, e.g., when an employee is terminated, his/her access should be cancelled to all applications on the Intranet. These lead to much inconvenience for them and threatens the security of these systems (e.g., if a password is stolen or forgotten). The security issue of these systems is attracting attention.

Many studies on practical examples show that the main source of security risk originates from the design and implementation of these software systems themselves (Lindqvist and Jonsson, 1998)(McGraw, 2002) and also from inside the enterprises or organizations (Harmantzis and Malek, 2004). Therefore, improving software security depends on proactively managing security associated with software design and implementation.

In the meantime, the security of existing systems must be protected and improved as far as possible with available measures. In addition, developing specific security solutions for different systems is usually very expensive and time-consuming within an enterprise. Thus, in this paper, we present an integrated RBAC framework that can incorporates multiple new developing systems and existing application systems within an organization, and provide end users access them with specific collaborations as a single system. It can implement a central authorization management mechanism, single sign-on access and integration management of security resources of all systems.

The remainder of the paper is organized as follows. In Section 2 a brief overview of related work is presented. The integrated approach and the system infrastructure to improve software security are described in Section 3. Section 4 details the implementation of the integrated approach. Section 5 evaluates the characteristics. Finally, concluding remarks and further work are provided in Section 6.

2 RELATED WORK

Role-based access control emerged rapidly in the 1990s as a proven technology for managing and enforcing security in large-scale enterprise-wide systems. Its authorization mechanism is concerned with making decisions on who can access what operations on which objects and how this can be achieved. Over the past years, a significant body of research on RBAC models and experimental implementations has developed (Ferraiolo and Kuhn, 1992),(Sandhu and Samarati, 1994)(Hitchens and Varadharajan, 2000). The work on access control classifies security models into three broad categories, namely Discretionary Access Control (DAC), Mandatory Access Control (MAC) (Hitchens and Varadharajan, 2000) and Role-Based Access Control (RBAC) (Sandhu and Coyne, 1996). In fact, they all make use of the roles for controlling access to entities and objects or the mechanism-oriented approach. It has been claimed elsewhere that RBAC better suits the needs of some real world organizations than MAC or DAC based approaches (Sandhu and Feinstein, 1994).

However, they all fall into the trap of using access control mechanisms both for policy expression as well as in policy enforcement (Hitchens and Varadharajan, 2000). It is hardly possible for any given policy specification to be implemented by all of the existing mechanisms. Some language-based proposals have been presented. Simon and Zurko (Simon and Zurko, 1997) discussed the mechanisms they were implementing to support Separation of Duty and roles in Adage, a general-purpose authorization language and toolkit. Jajodia et al. (Jajodia et al., 1997) proposed a logical language for the specification of authorizations on which such a model can be based. These languages often depend upon their users having a reasonable level of understanding of logical principles. This is not always found amongst real world users, even those entrusted with the management of access control policies for a system. Hitchens and Varadharajan (Hitchens and Varadharajan, 2000) proposed a language based approach to the specification of authorization policies (called Tower). It is far better to provide the user who needs to specify access control policies with flexible mechanisms and allow them to build structures for expressing their policies. But the implementation on top of other access control list mechanisms is somewhat inefficient. In addition, they didn't take into account the security resources management and integration, and the language-based specifications are difficult to understand and implement for programmers. In this paper, we can define RBAC mechanism by means of an integrated graphical interface in the light of business management policies. In the meantime, for a new application, we make use of a set of predefined functions (Application Programming Interface: API). These can present a sufficient degree of flexibility and capability.

3 INTEGRATED SECURITY INFRASTRUCTURE

The integrate security infrastructure is as shown in Fig. 1. It consists of an Integrated Control Panel (*ICP*), an Integrated Security Management component (*ISM*), Application System components (AS_1, AS_2, , AS_n). *ISM-DB* is a database that stores all relevant security information in the whole integrated management system. AS_i-*DB* is respectively a database that saves all data corresponding to application system.

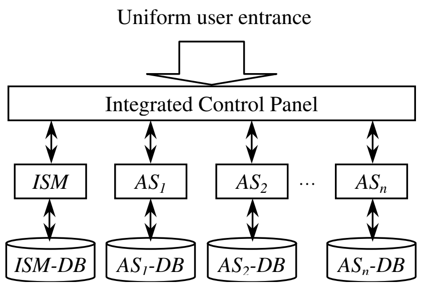
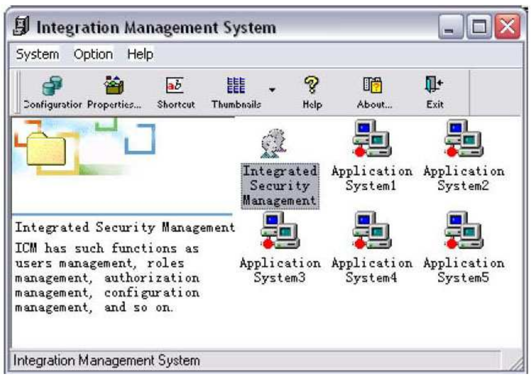


Figure 1: The Integrate Security Infrastructure.

3.1 Integrated Control Panel (*ICP*)

ICP, as depicted in Fig. 2, seems like the control panel in Windows. It integrates and manages the *ISM* component and AS_i ($i = 1, 2, \dots, n$) components. These components integrated in the *ICP* are either new developing components or any independent executable programs, e.g. some existing application systems. The *ICP* is an authentication and authorization gateway to the application components, all application components share the same authentication and authorization mechanism provided by the *ISM*. These components in the *ICP* can be invoked or loaded in terms of users' authorization. When a user clicks the icon of any component, the *ICP* first prompts the user to input username and password, communicates with the *ISM*, authenticates the user and retrieves authorizations from the *ISM-DB*. Hence, the user will not need to authenticate multiple times to access multiple application components, as shown in Fig. 3.



Integrated Security Management
ICM has such functions as users management, roles management, authorization management, configuration management, and so on.

Integration Management System

Figure 2: The Integrate Control Panel Infrastructure.

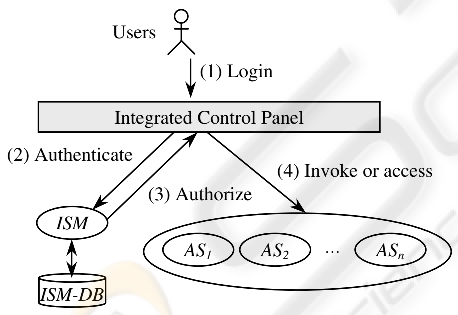
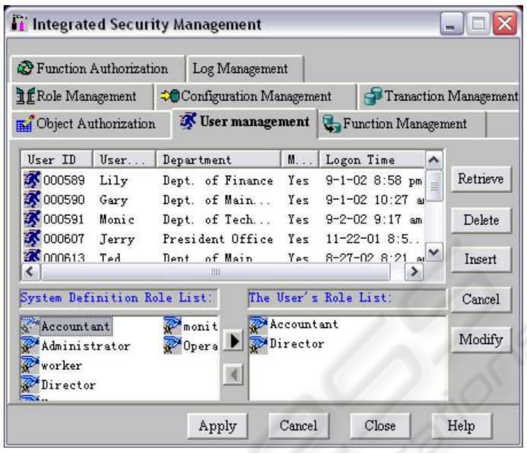


Figure 3: System Workflow.

3.2 Integrated Security Management (*ISM*)

ISM, as shown in Fig. 4, is the core of the integrated security infrastructure and the 'back-end' component that will be consulted by all the Application Components for making access decisions. It is made up of a graphical authentication and authorization interface and a set of Application Programming Interfaces (APIs). The graphical authentication and authorization interface in the *ISM* is achieved by using RBAC mechanisms to group users and enforcing the access control policies of resources.



User ID	User...	Department	M...	Logon Time
000589	Lily	Dept. of Finance	Yes	9-1-02 8:58 pm
000590	Gary	Dept. of Main...	Yes	9-1-02 10:27 a
000591	Monic	Dept. of Tech...	Yes	9-2-02 9:17 am
000607	Jerry	President Office	Yes	11-22-01 8:5..
000613	Ted	Dept. of Main	Yes	8-27-02 8:21 a

System Definition Role List:
Accountant
Administrator
worker
Director
monit
Opera

The User's Role List:
Accountant
Director

Figure 4: Integrated security management infrastructure.

Here, the RBAC mechanism has the following three essential structures (Sandhu et al., 1994):

(1) Users: which correspond to real world users of the computing system.

(2) Permissions: a description of the access users can have to objects in the system.

(3) Roles: a description of the functions of users within an organization.

In order to accurately model one real world job description, the RBAC mechanism in the integrated security infrastructure supports roles inheritance and constraints, for example, users may not be allowed to take on one role if another role they already have specifically disallows it.

A role table is a triple (RoleID, RoleName, R_Ship). RoleID is an exclusive tag of a role, RoleName is the name of a role, R_Ship is a set of constraints or role inheritance relationships among different roles. R_Ship places restrictions on how users are assigned to roles in the light of an organization's business management policies.

Permissions and users are mapped to roles, as shown in Fig. 5. The relationships are many to many. For a real world organization, we take two types of entities to which access is being controlled in the integrated security infrastructure. One specifies the object that represents all application system components; another is the function that stands for specified business functions, such as application, signature, read, write, and so on. In the RBAC model, permissions will therefore specify the object and the function which access is allowed. They can be expressed as two triples (RoleID, ObjectID, Permission) and (RoleID, FuncID, Permission). The objects, the functions and permissions all can be defined by the means of the graphical interface tool as shown in Fig. 4. All future accesses to any subsystems will retrieve authorization assertions from the database server that creates the user session. Hence, the user will not need to authenticate multiple times to access different subsystems integrated in the ICP on the network.

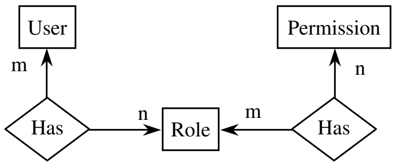


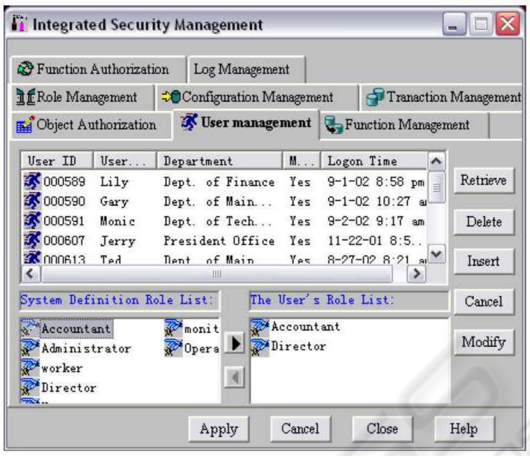
Figure 5: The relationship among user, role and permission.

3.3 Application System Components

Application system components integrated in the *ICP* include some newly developed systems and some existing systems. The set of Application Programming Interfaces (APIs)the *ISM* is a dynamic link library (DLL) that is shared by all the application system components. It includes encryption and decryption functions, authorization identification functions, users identification functions, roles code definition lists, some common variables, and so on. While a new application system is being developed, the API should be used so that the new application can seamlessly be integrated into the *ICP*. For some existing application systems, their configuration files are encrypted and stored in the database *ISM-DB*. Therefore, after they are integrated into the *ICP*, they cannot independently work again once they separated from the *ICP*. They may have their own authorization mechanism, but the *ICP* can control whether a user who only has a certain role can start up them or not. The security of these existing systems can be improved greatly. While the integrated system is installed, these components are optional according to users' requirements.

4 SYSTEM IMPLEMENTATION

In this section, we describe our implementation of a practical application system in Xiaolangdi hydropower plant by the means of the integration approach. It is made up of an *ICP*, an *ISM*, Human Resources Management System, Power Generating Management System, Finance Management System, Worksheet Management System, Real-time Information Display System and as well as some other systems (Gan, 2002), as shown in Fig. 6. The ISM is the core of system, so we only discuss it in the following.



User ID	User...	Department	M...	Logon Time
000589	Lily	Dept. of Finance	Yes	9-1-02 8:58 pm
000590	Gary	Dept. of Main...	Yes	9-1-02 10:27 a
000591	Monic	Dept. of Tech...	Yes	9-2-02 9:17 am
000607	Jerry	President Office	Yes	11-22-01 8:5..
000613	Ted	Dept. of Main	Yes	8-27-02 8:21 a

System Definition Role List:
Accountant
Administrator
worker
Director
monit
Opera

The User's Role List:
Accountant
Director

Figure 6: XLD integration information system infrastructure.

In order to achieve the goals presented in Section 3, more than ten tables were designed in the *ISM* database, such as T_User, T_Role, T_URole, T_Object, T_Auth, T_Ctrl, T_Func, T_log, and so on. The schemas are shown in Fig. 7. The implementation of the ISM supports the following functions:

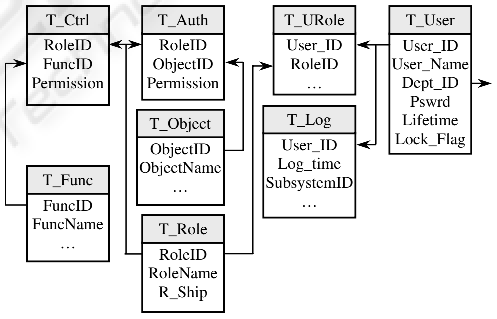


Figure 7: Relational schemas for the ISM database.

(1) Users management

Define users' detailed information, such as users' name, account, password, unit, roles, and the like. Any user can also be locked or unlocked according to business management policy, e.g. when an employee is terminated, his/her account will be locked, and his/her access will be cancelled to all applications on the Intranet.

(2) Configuration management

Configuration management centrally defines and encrypts the system parameters of all application components in the *ICP* and stores them in the database *ISM-DB*.

(3) Object management

In order to conveniently map the access authorization to roles, object management defines all application system components integrated in the *ICP*, including Human Resources Management System, Power Generating Management System, Finance Management System, Worksheet Management System, Real-time Information Display System. Here, the object is in the table T_Object.

(4) Functions management

Here, we define the function that stands for specified business functions in the hydropower plant, such as worksheet application, worksheet signature, worksheet permission, worksheet cancel, project audit, read, write, and so on. The function is in the table T_Func.

(5) Roles management

Roles delimit the functions of users within organizations by prescribing the access to objects that users have. Different roles are defined regarding to the relationships between job descriptions within the practical hydropower plant. In addition, role constraints and role inheritance are taken into account as well. These are saved in a triple table T_Role.

(6) Function and object authorization

By means of function and object authorization, The management policies in the hydropower plant are converted into users-roles-authorizations. All future access to any subsystems will retrieve authorization assertions from the database server that creates the user session. User identification and privileges determine the functionality of any application accessible through the *ICP*. They are respectively saved in two triple tables T_Ctrl and T_Auth.

(7) Log management

Log management can track and analyse software behaviour. Record user login time operation, audit trail. User information, such as login time, logout time, IP address, application system names and objects the user accesses during logon period, etc., are recorded in the table T_Log.

5 EVALUATION

Before the integrated system was implemented, both the Human Resources Management System and the Worksheet Management System were independently executed. A user had to respectively input his/her username and password twice every time while he/she needed to log on both the systems. The system administrator must maintain two sets of authorization and user management. However, after both the system was integrated in the ICP, everything is becoming simple. System administrator only needs to maintain a set of authorization and user management. They can log on all the systems after users only needs to input his/her username and password once.

In addition, compared with some Role-based access control approaches used in various application systems (Thomsen et al., 1998)(Apfrlbeck, 2001), the central security infrastructure presented in this paper has some characteristics in the following:

(1) Granularity and extensibility

The ISM is able to provide finer-grain resource access control than existing solutions since all primary resources (applications, functions, operations and data) can be separately accessed; and it allows security policies to be flexibly and extensively defined.

(2) Accountability and auditability

The log management of the ISM provides operation log in order to track record. A lot of relevant information can be recorded about actions performed by users, or processes acting on their behalf, so that the consequences of those actions can later be linked to the users in question and, if necessary, users can be held accountable and auditable for their actions. It can satisfy requirements for both accountability and auditability well.

(3) Feasibility and integrity

From Integrated Control Panel, a set of APIs can deliver enormous value to IT organizations and end users, enabling centralized access to applications in order to gain a simplified infrastructure, faster development, and enhanced programmers and users efficiency.

The *ICP* framework can seamlessly integrate existing application systems and provide users with dynamic authorization into the applications.

(4) Authentication security strengthening

The *ICP* greatly improves the authentication security of application systems by its Single Sign-On access. A user can access all authorized applications, on the basis of a single authentication that is performed when he/she initially accesses any application integrated into the *ICP*. Since the user has to remember only one username/password, and needs to type this information once, user productivity is improved and security breaches such as users writing down their passwords are eliminated.

On the other hand, the *ISM* is an independent authentication system without user interaction. It does not only centrally manage and control access to all application systems integrated into the *ICP* but also provides a highly customizable set of role-based access control policies and role mappings, which also greatly improve the security of all application systems.

6 CONCLUSIONS AND FUTURE WORK

This paper describes a central security infrastructure to enable SSO for different application subsystems via integration mechanisms, role-based access control and security resource management. A case study of the central security frame is presented as well.

In the current implementation, mapping of users to roles and mapping of roles to permissions are created and maintained in a predefined format in the ISM by the security administrator. Future work will study a certain standard format for scripting this related security information, and automatically implementing mapping of users to roles and mapping of roles to permissions so as to improve the security of sensitive information systems and to provide a more flexible mechanism. In addition, the API presented in this paper can only support the same development tool. It is also a future research direction how the API will be improved to adapt to different development tools.

ACKNOWLEDGEMENTS

This work is supported by Geheyan Hydropower Station Information Integrated System Project, Xiaolangdi Hydropower Station Information Integrated System Project and China Scholarship Fund. We thank the anonymous referees for their valuable comments and suggestions to improve the quality of this paper.

REFERENCES

Apfrlbeck, J. (2001). A role based modelling approach for the information infrastructure. In *Proceedings of the IEEE International Conference on Communications*, pages 1617–1621. IEEE CS Press.

Ferraiolo, D. and Kuhn, D. R. (1992). Role based access control. In *Proceedings of the 15th Annual Conference on National Computer Security*, pages 554–563, Piscataway, NJ, USA. IEEE Press.

Gan, Z. B. (2002). The technical report for xiaolangdi power station information integrated system. Technical report, Huazhong University of Science and Technology.

Harmantzis, F. and Malek, M. (2004). Security risk analysis and evaluation. In *Proceedings of the 2004 IEEE International Conference on Communications*, pages 1897–1901, Piscataway, NJ, USA. IEEE Press.

Hitchens, M. and Varadharajan, V. (2000). Design and specification of role-based access control policies. In *IEE Proceddings Software*, pages 117–129, Piscataway, NJ, USA. IEEE Press.

Jajodia, S., Smarati, P., and Subrahmanian, V. (1997). A logical language for expressing authorizations. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 31–42, Piscataway, NJ, USA. IEEE Press.

Lindqvist, U. and Jonsson, E. (1998). A map of security risks associated with using cots. *Computer*, 31(6):60–66.

McGraw, G. (2002). Managing software security risks. *Computer*, 35(4):99–101.

Sandhu, R. and Coyne, E. (Feb. 1996). Role-based access control models. *Computer*, pages 38–47.

Sandhu, R., Coyne, E., and et al., H. F. (1994). Role-based access control: A multi-dimensional view. In *Proceeding of the 10th Annual Computer Security Applications Conference*, pages 54–61, Orlando, FL, USA. IEEE CS Press.

Sandhu, R. and Feinstein, H. (1994). A three tier architecture for role-based access control. In *Proceedings of the 17th national computer security conference*, pages 34–46. IEEE CS Press.

Sandhu, R. S. and Samarati, P. (1994). Access control: principle and practice. *IEEE Comm.*, pages 40–48.

Simon, R. and Zurko, M. (1997). Separation of duty in role-based environments. In *Proceedings of the 10th Computer Security Foundations Workshop*, pages 183–194, Piscataway, NJ, USA. IEEE Press.

Thomsen, D., O'Brien, D., and Bogle, J. (1998). Role based access control framework for network enterprises. In *Proceedings of the 14th Annual Computer Security Applications Conference*, pages 50–58, Phoenix, Arizona. IEEE CS Press.