Journal of Engineering Science and Technology Review 2 (1) (2009) 48-50

Research Article

JOURNAL OF
Engineering Science and Technology Review

www.jestr.org

Designing role-based access control policies with UML

A. Cenys, A. Normantas* and L. Radvilavicius

Information Security Laboratory, Department of Information System, Faculty of Fundamental Sciences, Vilnius Gediminas Technical University Sauletekio al. 11, SRL-I-415, LT-10223, Vilnius, Lithuania

Received 18 April 2009; Accepted 18 May 2009

Abstract

The paper analyses role-based access control (RBAC) and two methodologies, namely SecureUML and UMLsec, aiming for designing RBAC policies. The features of both methodologies are represented and compared by modeling the specific system, with special attention to how RBAC policies and principles are modeled using SecureUML and UMLsec.

Keywords: UML, RBAC, UMLsec, SecureUML.

1. Introduction

In computer security, access control is the ability to permit or deny the use (access) of a particular system resource by a particular entity. Techniques on how access control can be modeled generally falls into two categories: discretionary and non-discretionary.

Discretionary access control (DAC) is an access control technique where the owner of object controls the permissions to access the object. In non-discretionary access control, called mandatory access control (MAC), permissions are handled by system, not the owner of object. Objects in these systems are assigned with security label which restrict who is allowed to access them. A label contains the required clearance level, which often ranges from unclassified to classified, secret, top secret, etc. Users of the system are assigned clearance levels.

DAC and MAC techniques existed as the only ones to describe access policies until 1992 when D.F.Ferraiolo and D.R.Kuhn separated the new technique in [3]. It was called the role-based access control (RBAC), and here the users are given the permissions to objects indirectly – through roles. RBAC is proved to be more flexible and powerful than MAC or DAC. The unified modeling language (UML) can be used to specify RBAC policies.

The main objective of this paper is the analysis of modeling languages and methodologies, which can be used to design RBAC mechanism of the system. Particularly, two security-aimed methodologies, UMLsec and SecureUML, and their general approach to RBAC are an object of research in this work.

This paper is organized as follows. In Section 2, the background of the work is explained. This section consists of general information about RBAC model. Section 3 and 4 contents are the two methodologies of secure system design, represented in this paper – UMLsec and SecureUML. Section 5 explains an important RBAC principle – separation of duty, while Section 6 focuses on RBAC policies in the specific system, and what decisions UMLsec and SecureUML methodologies can provide. Finally, conclusions are drawn in last section.

2. Role-based access control

In this work, RBAC is described as a method of regulating access to functions or operations of system by the roles of individual users. The principle scheme of RBAC is shown in Fig. 1.

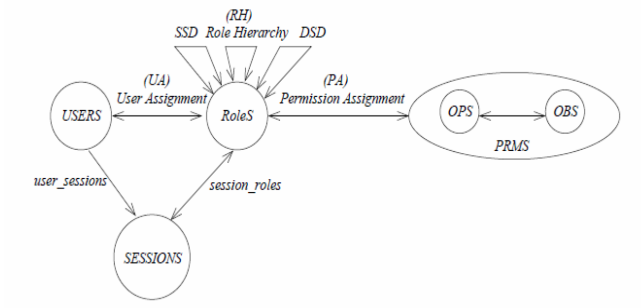

Fig. 1 Different components of RBAC [1]

The components (or entities) of the Fig. 1 are explained as follows. A *user* is defined as an individual person, software agent or another subject in the organization. A *role* is a function within that organization. An *operation* is an event of taking action on system's protected resource. The *permission* represents the authorization to execute the operation. Finally, permissions are assigned to roles and so-called privileges are created.

* E-mail address: andrius.normantas@gmail.com

ISSN: 1791-2377 © 2009 Kavala Institute of Technology. All rights reserved.

The use of RBAC to manage user access control via permission assignment to roles within a single system or application is widely accepted as a best practice. Systems including FreeBSD, Solaris, Oracle DBMS and many others effectively implement some form of RBAC.

3. SecureUML

SecureUML [2,5] is an extension of UML for specifying RBAC and other access control policies for actions on protected resources through the use of a security modeling language. The abstract syntax and semantics allow it to be combined with design modeling languages. This methodology is basically the language of RBAC extended with authorization constraints whose are expressed in Object Constraint Language (OCL).

Fig. 2 shows the SecureUML metamodel in UML class diagram where metamodel types introduced in SecureUML are marked in lighter colour. This metamodel is defined as an extension of the UML metamodel. SecureUML introduces the new metamodel types *User*, *Role* and *Permission.* Relations between those types are defined as standard UML associations. Another type introduced in SecureUML is *AuthorizationConstraint* which uses standard UML core type *Constraint* to express a pre-condition for calls of operations of user-defined resource set.

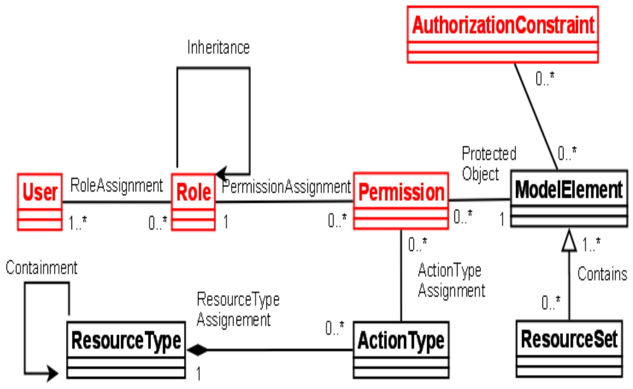

Fig. 2 SecureUML metamodel

4. UMLsec

Another UML extension for secure systems development is UMLsec, introduced by Jan Jürjens in [4]. Here, the recurring security requirements, such as secrecy, integrity, authenticity etc., are offered as specification elements. Various kinds of UML diagrams are used to indicate possible vulnerabilities.

The UMLsec extension is given in a form of using the standard UML extension mechanisms, such as stereotypes, tags, and constraints; stereotypes with tags formulate the security requirements, while constraints give criteria whether the requirements are met by the system design, or not.

A central idea of the UMLsec extension is to define labels for UML model elements (stereotypes), which, when attached, add security-relevant information to these model elements [4]. Such a label is «rbac» stereotype, which will be explained in subsequent sections.

5. Separation of duty

In RBAC, separation of duty (SOD) constraints are used to enforce conflict of interest policies [1]. These constraints can be of two types:

- Static SOD (SSD) – preventing conflicts of interests that arise when user gains permissions associated with conflicting roles, i.e. roles that cannot be assigned to the same user;
- Dynamic SOD (DSD) – place restrictions on the roles that cannot be activated within the same user session.

The static and dynamic SOD constraints can also be placed on permissions or users rather than roles. Depending on the object that constraint is attached to, constraints are referred to SSD-Role, SSD-Permission or SSD-User constraints.

6. Modeling RBAC

To illustrate modeling opportunities of UMLsec and SecureUML, we use a simple brokerage corporation application as a system of specific RBAC policies. The system is used by various employees within their company to perform their duties, and one of the main policies of the company is that information resources should be protected from unauthorized access.

The system RBAC policies are given below:

- Roles: {broker, accountant, consultant, customer service representative, internal auditor, broker department manager, customer service department manager};
- Permissions assigned to roles:
    - The broker can modify trade data, execute orders and confirm deals;
    - The accountant can generate reports;
    - The customer service representative can read information related to customers;
    - The consultant can perform select, insert, update and delete operations with customers' database;
    - The internal auditor has read-only access to all resources.
- Hierarchical (roles) policies:
    - The customer service department manager role is senior to the consultant and customer service representative roles;
    - The broker department manager role is senior to the broker and accountant roles.
- SOD constraints:
    - SSD constraints:
        - (broker department manager, customer service department manager);
        - (broker, consultant)
        - (accountant, customer service representative)
    - DSD constraints:
        - (internal auditor, *x*), where *x* is any other role, meaning that the user cannot activate internal auditor role while being assigned to any other role in the same session.

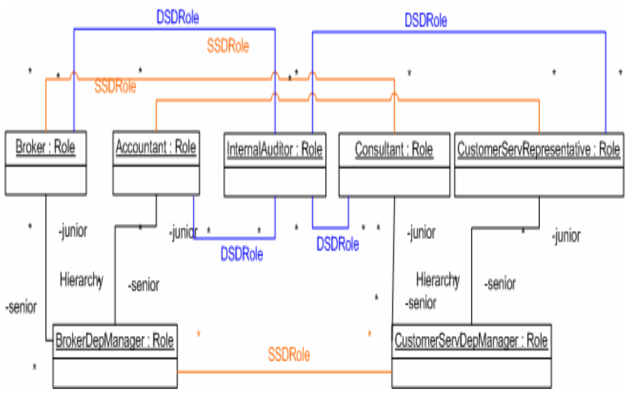

Fig. 3 SOD constraints in UML object diagram

SOD constraints and hierarchical policies are modeled in SecureUML and shown in Fig. 3. UMLsec does not offer any special features related to SOD but the set of existing tools (stereotype «rbac» and standard UML notation) makes SOD modeling possible.

The full model of particular system is shown in Fig. 4. Note that here new elements appear, namely *module* and *moduleGroup*. Generally, they extend *resource* element. The module is a conceptual system unit which uses the resources, such as database objects, files, and applications. The group of modules is a composition of such modules.

Modeling RBAC in UMLsec, one needs to know stereotype «rbac» and its tagged values:

- {protected} – for the states of activity diagram access to whose should be controlled;
- {role} – a pair of values of users assignments to roles;
- {right} – a pair of values of rights given to roles.

«rbac» stereotype is based on UML package element and activity diagram within. While roles are separated by dashed lines, protected objects are located into respective role fields. An example of «rbac» stereotype is shown in Fig. 5.

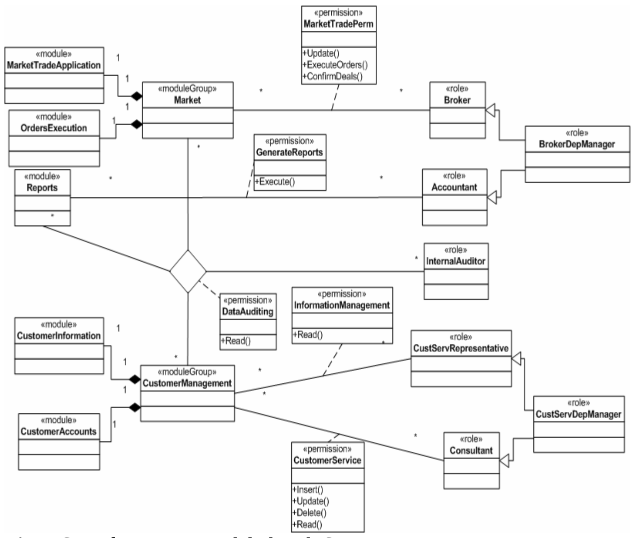
«permission» MarketTradePerm
+Update()
+ExecuteOrders()
+ConfirmDeals()
«module» MarketTradeApplication
«moduleGroup» Market
«role» Broker
«module» OrdersExecution
«permission» GenerateReports
+Execute()
«role» BrokerDepManager
«module» Reports
«role» Accountant
«role» InternalAuditor
«permission» DataAuditing
+Read()
«permission» InformationManagement
+Read()
«module» CustomerInformation
«role» CustServRepresentative
«moduleGroup» CustomerManagement
«role» CustServDepManager
«module» CustomerAccounts
«permission» CustomerService
+Insert()
+Update()
+Delete()
+Read()
«role» Consultant

Fig. 4 Specific system modeled with SecureUML

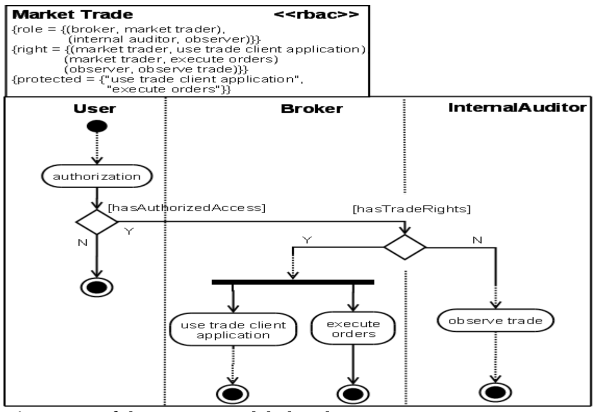
Market Trade
<<rbac>>
{role = {(broker, market trader),
(internal auditor, observer)}}
{right = {(market trader, use trade client application)
(market trader, execute orders)
(observer, observe trade)}}
{protected = {"use trade client application",
"execute orders"}}
User
Broker
InternalAuditor
authorization
[hasAuthorizedAccess]
[hasTradeRights]
Y
N
Y
N
use trade client application
execute orders
observe trade

Fig. 5 Part of the system modeled with UMLsec

7. Conclusions

Due to the fact that SecureUML is a language based on RBAC, this plays a significant role for choosing this language instead of UMLsec when it comes for designing RBAC policies in the specific system.

Both UMLsec and SecureUML need some improvements related to RBAC. UMLsec has no support for various RBAC constraints, such as separation of duty, or pre- or post- conditions, which are offered by SecureUML in OCL.

SecureUML has no sophisticated structures of resources, such as hierarchies or compositions, which might be helpful when designing complex systems.

.

References

1. I.Ray, N.Li, R.France, D.-K.Kim. Using UML To Visualize Role-Based Access Control Constraints. In Proceedings of the ninth ACM symposium on Access control models and technologies, p.115-124, Yorktown Heights, New York, USA, 2004.
2. T.Lodderstedt, D.Basin, J.Doser. SecureUML: A UML-Based Modeling Language for Model-Driven Security. UML 2002 - The Unified Modeling Language: 5th International Conference, Dresden, Germany, 2002.
3. D.F.Ferraiolo, D.R.Kuhn. Role-Based Access Controls. In 15th National Computer Security Conference, Baltimore, 1992.
4. J.Jürjens. Secure Systems Development with UML. Springer, 2004.
5. J.Doser. Analysis of SecureUML Models. Freiburg, 2003.
6. T.Lodderstedt, D.Basin, J.Doser. Model Driven Security: from UML Models to Access Control Infrastructures. ACM Transactions on Software Engineering and Methodology (TOSEM), Volume 15, Issue 1, 2006.
7. J.Jürjens. Sound Methods and Effective Tools for Model-based Security Engineering with UML. International Conference on Software Engineering, Shanghai, China, 2006.
8. Th.Doan, L.Michel, S.Demurjian. A Formal Framework for Secure Design and Constraint Checking in UML. International Symposium on Secure Software Engineering, Washington D.