Role-Based Scheduling and Synchronization Algorithms to Prevent Illegal Information Flow

Tomoya Enokido[1], Valbona Barolli[2], and Makoto Takizawa[2]

[1] Rissho University, Japan
eno@ris.ac.jp
[2] Tokyo Denki University, Japan
valbona@takilab.k.dendai.ac.jp, makoto.takizawa@computer.org

Abstract. Information systems have to be consistent and secure in presence of multiple conflicting transactions. The role-based access control model is widely used to keep information systems secure. Here, a role is a set of access rights, i.e. permissions. A subject is granted a family of roles, i.e. one or more than one role. A subject s is allowed to issue a method op to an object o only if an access right $\langle o, op \rangle$ is included in the roles granted to the subject s. In the access control models, even if every access request satisfies the access rules, illegal information flow might occur as well known confinement problem. In this paper, we define a legal information flow relation ($R_1 \Rightarrow R_2$) among a pair of role families R_1 and R_2. This means, no illegal information flow occur if a transaction T_1 with a role family R_1 is performed prior to another transaction T_2 with R_2. In addition, we define which role families are more significant than others in terms of types of methods and security classes of objects. Conflicting methods from different transactions are totally ordered in the significancy of roles of the transactions. We discuss how to synchronize transactions so as to prevent illegal information flow and how to serialize conflicting methods from multiple transactions in terms of significancy and information flow relation of roles families.

1 Introduction

It is critical to discuss how to make information systems more consistent and secure in presence of various kinds of security attacks and conflicting accesses to resources. In the basic access control model [2], only a subject s, i.e. user and program is allowed to issue a method op to an object o like a database system [13,16] only if an access right (or permission) $\langle o, op \rangle$ is granted to the subject s. In an enterprise, each person plays one or more than one role. Here, a role shows a job function. In the role-based access control models [9,10,15,17], a role is specified in a collection of access rights, which shows what subjects playing the role can do on resources of an enterprise. While the access control models are widely used in information systems like database systems [13,16], illegal information flow among subjects through objects may occur even if each subject can safely manipulate objects according to the access rights. This is *confinement* problem [3,12]. In the lattice-based access control model is discussed [4,14], each

T. Enokido, L. Barolli, and M. Takizawa (Eds.): NBiS 2007, LNCS 4658, pp. 238–247, 2007.

entity, i.e. subject and object is classified into a security class. Information flow relation among security classes is defined. Access rules on read, write, and modify are defined according to the information flow relation.

In this paper, we newly discuss a concurrency control mechanism to synchronize conflicting transactions to achieve two objectives. First, a more significant method should be performed prior to others from the application point of view. Next, no illegal information flow should occur. First, we discuss how to prevent illegal information flow in the role-based access control model. A subject s issues a transaction T to manipulate objects. The transaction T is assigned a subfamily of the roles granted to the subject s. The subfamily of the roles is referred to as *purpose* of the subject s to perform T. First, we define a *legal information flow* relation $R_1 \Rightarrow R_2$ among purposes R_1 and R_2. This means that there occur no illegal information flow if a transaction T_1 with the purpose R_1 is performed prior to another T_2 with R_2. We also discuss which role family is more significant than another role family based on the significancy of methods [5,6,7]. For example, *write* is more significant than *read* since the object state is changed by *write* and *withdraw* is more significant than *deposit* in a bank application. The types of schedulers to perform a method from a more significant transaction prior to a less significant one are also discussed. We discuss how to serialize conflicting read and write methods from multiple transactions so that illegal information flow is prevented based on the legal information flow relation of the purposes.

In section 2, we overview the role-based access control models and briefly discuss what role family is more significant than another. In section 3, we define the legal information flow relation among roles. In section 4, we discuss the role-based scheduling and synchronization mechanisms of multiple transactions to prevent illegal information flow while serializing conflicting transactions.

2 Role-Based Access Control Models

2.1 Roles

In the access control models [2,4,9,10,14,15,17], a system is composed of two types of entities, subjects and objects. A subject is an entity who issues methods to objects like user. An object is an entity which performs methods from subjects like database. Only a subject s granted an access right (or permission) $\langle o, op \rangle$ is allowed to issue a method op to an object o.

In the role-based access control models [9,10,15,17], a role is given as a set of access rights and shows a job function in an enterprise. Let O be a set $\{o_1, ..., o_m\}$ of objects in the system. Each object o_i supports methods for manipulating data ($i = 1, ..., m$). Let R be a set of roles $\{r_1, ..., r_n\}$ in a system. Each role r_i is a collection $\{\alpha_{i1}, ..., \alpha_{il_i}\}$ of access rights. Each access right α_{ij} is a pair $\langle o_{ij}, op_{ij} \rangle$ of an object o_{ij} in O and a method op_{ij} for manipulating o_{ij}. A subject s is granted one or more than one role. Let $SR(s)$ ($\subseteq R$) be a family of roles granted to a subject s. A subject s is allowed to issue a method op to an object o only if an access right $\langle o, op \rangle$ is in the role family $SR(s)$.