Chapter 11
Analysis of ANSI RBAC Support in EJB

Wesam Darwish
The University of British Columbia, Canada

Konstantin Beznosov
The University of British Columbia, Canada

ABSTRACT

This paper analyzes access control mechanisms of the Enterprise Java Beans (EJB) architecture and defines a configuration of the EJB protection system in a more precise and less ambiguous language than the EJB 3.0 standard. Using this configuration, the authors suggest an algorithm that formally specifies the semantics of authorization decisions in EJB. The level of support is analyzed for the American National Standard Institute's (ANSI) specification of Role-Based Access Control (RBAC) components and functional specification in EJB. The results indicate that the EJB specification falls short of supporting even Core ANSI RBAC. EJB extensions dependent on the operational environment are required in order to support ANSI RBAC required components. Other vendor-specific extensions are necessary to support ANSI RBAC optional components. Fundamental limitations exist, however, due to the impracticality of some aspects of the ANSI RBAC standard itself. This paper sets up a framework for assessing implementations of ANSI RBAC for EJB systems.

INTRODUCTION

The American National Standard for Information Technology Role-Based Access Control (ANSI RBAC) (ANSI, 2004) is a specification of an access control system in which permissions are associated with roles, and users are assigned to appropriate roles. RBAC is an approach to address the needs of commercial enterprises better than lattice-based mandatory access control (MAC) (Bell & LaPadula, 1975) and owner-based discretionary access control (DAC) (Lampson, 1971). A

DOI: 10.4018/978-1-4666-2482-5.ch011

role can represent competency, authority, responsibility, or specific duty assignments. The ANSI RBAC standard consists of two main parts: the RBAC Reference Model, and the RBAC System and Administrative Functional Specification. Both parts cover four components: the minimum set of features included in all RBAC systems (*Core RBAC*), role hierarchies (*Hierarchical RBAC*), static constraint relations (*Static Separation of Duty Relations*), and dynamic constraints (*Dynamic Separation of Duty Relations*). A major purpose of RBAC is to facilitate access control administration and review.

Many papers propose ways to support or implement RBAC using commercial technologies, e.g., Oracle (Notargiacomo, 1995), NetWare (Epstein & Sandhu, 1995), Java (Giuri, 1998), DG/UX (Meyers, 1997), J2EE (Zhang, Sheng, Niu, Wang, & Zhang, 2006; Bindiganavale & Ouyang, 2006), object-oriented systems (Barkley, 1995), object-oriented databases (Wong, 1997), MS Windows NT (Barkley & Cincotta, 1998), enterprise security management systems (Awischus, 1997). Evidence of RBAC becoming a dominant access control paradigm is the approval of the American National Standard Institute (ANSI) RBAC Standard (ANSI, 2004) in 2004.

At the same time, commercial middleware technologies−such as Common Object Request Broker Architecture (CORBA) (OMG, 1999), COM+ (Oberg, 2000), Enterprise Java Beans (EJB) (DeMichiel, Yalçinalp, & Krishnan, 2001)− matured, with distributed enterprise applications routinely developed with the use of middleware. Each middleware technology, however, comes with its own security subsystem (Eddon, 1999; OMG, 2002; Hartman, Flinn, & Beznosov, 2001), sometimes dependent on and specific to the underlying operating system (OS). For instance, COM+ security (Eddon, 1999) is tied into Microsoft Windows OS and its services.

The ability of a particular middleware technology to support specific types of access control policy is an open and practical question. It is not a simple question for the following three reasons.

First, different middleware technologies and their subsystems are defined in different forms and formats. For example, CORBA is specified in the form of open application programming interfaces (APIs), whereas EJB is defined through APIs as well as the syntax and semantics of the accompanying extensible markup language (XML) files used for configuring the EJB container. COM+ is defined through APIs as well as graphical user interfaces (GUI) for configuring the behavior of a COM+ server. The variations in the form, terminology, and format of the middleware definitions lead to the difficulty of identifying the correspondence among the (security and other) capabilities of any two middleware technologies.

Second, the capabilities of the middleware access controls are not defined in the terms of any particular access control model. Instead, the controls are defined in terms of general mechanisms which are supposed to be adequate for the majority of cases, and could be configured to support various access control models. Designed to support a variety of policy types, as well as large scale and diverse distributed applications, the controls seem to be a result of engineering compromises between, among others, perceived customer requirements, the capabilities of the target runtime environment, and their expected usage. For example, CORBA access controls are defined in the terms of the principal's *attributes*, *required*, and *granted rights*, whereas EJB controls are defined using *role mappings* and *role-method permissions*. Assessing the capability of middleware controls to enforce particular types of authorization policies is harder due to the mismatch in the terminology between the published access control models and abstractions directly supported by the controls.

Third, the security subsystem semantics in commercial middleware is defined imprecisely, leaving room for misinterpretation. We clarify the semantics of the security subsystem and analyze its