Concepts and Capabilities of Middleware Security

S. Demurjian, K. Bessette, and T. Doan¹, C. Phillips²

¹The University of Connecticut
²United States Military Academy

9.1 Introduction

Distributed computing applications for the twenty-first century, constructed from legacy, commercial-off-the-shelf (COTS), database, and new client/server applications, require stakeholders (i.e., software architects, system designers, security officers, etc.) to architect and prototype solutions that facilitate the interoperation of new and existing applications in a network-centric environment. In these solutions, security must play a fundamental part, considered at early and all stages of the design and development lifecycle. The emergence of distributed computing technology such as DCE [6, 13], CORBA [4, 16, 18], DCOM/OLE [3], J2EE/EJB [12, 15], JINI [1, 17], and .NET [11, 14] has enabled the parallel and distributed processing of large, computation-intensive applications. Historically, the incorporation of security has often been an afterthought, dependent on programmatic effort rather than a cohesive mechanism seamlessly incorporated into the underlying technology. However, there has been a dramatic turnaround in the support of security, particularly in modern middleware platforms such as Common Object Request Broker Architecture (CORBA), .NET, and Java 2 Platform, Enterprise Edition (J2EE). The objective of this chapter is a two-fold examination of the concepts and capabilities of middleware security: exploring the security capabilities in three popular middleware platforms, namely, CORBA, .NET, and J2EE; and utilizing middleware concepts to realize complex and critical security approaches, namely, role-based and mandatory access control.

Toward this objective, in Section 9.2 of this chapter, we discuss the state of the art in support of middleware security, focusing on the security abilities of CORBA, .NET,
and J2EE. For CORBA, we explore its security features for confidentiality, integrity, accountability, and availability [5], keeping in mind that CORBA security is a meta-model that characterizes a breadth of security capabilities representing different security models, paradigms, and techniques. On the other hand, .NET and J2EE provide actual security capabilities via their respective runtime environments, and the application programmer interfaces (APIs), which provide security functionality. As such, .NET and J2EE can be considered, at some level, a realization of the CORBA security meta-model, implementing a subset of its features and capabilities. Thus, for both .Net and J2EE, we present and discuss their security potential and features with a five-prong approach: code-based access control for security embedded in actual program code, role-based access control for user-oriented security, secure code verification and execution that focuses on runtime security support, secure communication of the exchange of messages and information, and secure code and data protection to detail cryptographic capabilities. When relevant, we compare and contrast the features of .Net and J2EE, and we also note their similarities to different portions of the CORBA meta-model. Overall, in Section 9.2, we cover a range of security capabilities from conceptual (meta-model of CORBA) to practical (.NET and J2EE).

In this section, we explore the security capabilities of CORBA [5], .NET [21], and J2EE [24], highlighting their features and providing a context that explains their approaches. The major difference between the support for security in CORBA (as opposed to its realization in an actual CORBA product, e.g., Visibroker) and security in .NET/J2EE is that the CORBA security specification is a meta-model, akin to the fact that UML is a meta-model with various implementations (e.g., Together, Rational, etc.). As a meta-model, the CORBA security specification is attempting to generalize many security models and associated security principles, to arrive at a specification that is comprehensive (wide variety of security capabilities at the model level – RBAC, MAC, encryption, etc.) while