A Framework for Securing Component-Based Applications

Hieu Dinh Vo
PhD Student
Software Engineering Laboratory
School of Information Science, JAIST

Research Aims

Most large-scale information systems today are built based on software component technology. Using components to develop applications significantly reduces complexity, time, and development cost. Systems developed using components are easy to be maintained and evolved. In addition, a component used in a system can be re-used in other systems. However, using components in developing information systems can introduce new security risks to the systems. The main source of security issues in component-based systems is the heterogeneous of the components used to develop the systems. That is these components may be bought from several third-party vendors. In this case, the developers of the systems lack of information about how secure the used components are.

The purpose of this research is to provide a framework that helps component-based system developers to build secure systems. The framework includes a means for specifying and verifying security policies of a component-based system and mechanisms to enforce those policies.

Approach

While there are several platforms for component-based applications, our research mainly focuses on Enterprise JavaBeans platform. We implement sample applications and analyze security aspects of those applications, especially with the assumption that components are developed by third-party vendors and source code is not available. From that analysis, we propose solutions for found security issues.

Progress of 2007

In 2007, we focused on two problems of EJB applications: the error-prone way of defining declarative access controls and the loose enforcement of access control.

In J2EE applications, there are two ways for establishing RBAC: programmatic and declarative. In the former manner, the access controls are embedded inside components code via API calls. The latter allows application developers to specify access control for EJB methods in a separate configuration file. One of the advantages of programmatic access control is that this mechanism provides application developers the ability to specify fine-grained policies. However, declarative access control is simpler and more portable. By using this kind of access control, we can separate security from business logic of systems. The declarative access control is coarse-grained when be compared with programmatic access control, from another perspective however, the method is too fined-grained. Method-based access control is only suitable for systems where the number of components is small and interactions between components are not complicated. In large-scale systems, the current method becomes too fine-grained, error-prone and difficult for maintaining security policy. Our solution for this problem is instead of providing access control based on each method of beans, we establish access control on a higher level of abstraction, called business function level. From access control based on business functions, we can derive an equivalent method-based access control. This new approach can address some drawbacks of the method-based approach when used in large-scale systems.

The access control in the current EJB applications can assure that a certain method is only invoked by authorized people. However, in the context of component-based software, we also need to assure that the method only be invoked from a right place. By utilized the concept of business function above, we have developed a framework for this purpose. In our framework, we use business functions to define the call flows (at design time). Meanwhile, at runtime, a call flow is a series of methods invoked by a thread. The framework ensures that at any point during the lifetime of a given thread, it must conform to at least one of the defined business functions. We enforce this policy by assigning a set of potential business functions to each thread at the first time the thread invoke a method. This set is then updated each time the thread invokes a method. The framework blocks any thread with no business function associated with.

Future Directions

The current framework is designed based on the fact that EJB cannot create a new thread (this is prohibited in current EJB specification) and no new thread is created by the EJB container. One of our future works is to extend the framework so that it can handle threads created by components and by the EJB container. In addition, when the number of EJB components in an application grows, the task of defining business functions should become complicated. We intend to provide a tool that can extract business functions from design documents such as UML diagrams. Our further aim is to build a multi-layer framework for securing EJB applications.

Publications

Vo, H.D. and M. Suzuki. *An Approach for Specifying Access Control Policy in J2EE Applications.* in *Asia-Pacific Software Engineering Conference, 2007.*