Security Evaluation of Service-Oriented Systems Using the SiSOA Method

Christian Jung, Fraunhofer Institute for Experimental Software Engineering, Germany

Manuel Rudolph, Fraunhofer Institute for Experimental Software Engineering, Germany

Reinhard Schwarz, Fraunhofer Institute for Experimental Software Engineering, Germany

ABSTRACT

The Service-Oriented Architecture paradigm (SOA) is commonly applied for the implementation of complex, distributed business processes. The service-oriented approach promises higher flexibility, interoperability and reusability of the IT infrastructure. However, evaluating the quality attribute security of such complex SOA configurations is not sufficiently mastered yet. To tackle this complex problem, the authors developed a method for evaluating the security of existing service-oriented systems on the architectural level. The method is based on recovering security-relevant facts about the system by using reverse engineering techniques and subsequently providing automated support for further interactive security analysis at the structural level. By using generic, system-independent indicators and a knowledge base, the method is not limited to a specific programming language or technology. Therefore, the method can be applied to various systems and adapt it to specific evaluation needs. The paper describes the general structure of the method, the knowledge base, and presents an instantiation aligned to the Service Component Architecture (SCA) specification.

Keywords: *Information Systems, IT Security, Security Evaluation, Security Knowledge Base, Service-Oriented Architectures, Software Architecture*

INTRODUCTION

Service-oriented software architectures (SOA) promise enhanced reusability, interoperability, and flexibility for the implementation of business processes in information systems. However, this increase in flexibility and versatility comes at a price: It aggravates software quality assurance. The distributed, inhomogeneous, and often non-transparent nature of service building blocks stemming from different organizational domains is a supplementary constraint for the reliable determination of software quality attributes, especially those that are global properties of the overall SOA system, such as safety or security. Although technical standards such as the Web Services Security Specification (OASIS, 2010) exist, SOA systems are still vulnerable to many basic threat types.

DOI: 10.4018/jsse.2011100102

Security is an overarching quality concern that requires adequate treatment at a holistic system level. It cannot be handled effectively by analyzing the security issues only at source code level, especially not in a manual manner. To better keep track of the global security characteristics and to survey the logical security design of a system, all security-related information should be assessed in the context of a more abstract, structural level of the fundamental system architecture. Security-related information refers to system characteristics that can have a positive or negative impact on the system's security, such as code locations where security functions (e.g., authentication, encryption, integrity check) are called or where configuration parameters controlling these functions are defined. We claim that architectural views provide an adequate point of view for the security assessment of complex software systems.

In a SOA, all components have to be analyzed in their current configuration. However, the number of components, their changing orchestration and the distributed nature of SOA systems often renders a manual analysis impracticable.

System behavior, especially the dynamic security characteristics of the system in its entirety, is hard to obtain if the relevant information is scattered across many SOA components and their respective design artifacts.

In an earlier publication (Antonino, Duszynski, Jung, & Rudolph, 2010), we presented SiSOA (»Security in Service-oriented Architectures«), an assessment method for collecting security-related system properties and presenting them in architectural views for efficient evaluation. SiSOA comprises three phases: Extraction, Identification, and Analysis of security properties, as shown in Figure 1. The Extraction phase uses static analysis and standard reverse engineering techniques to gather security-related information from the system under evaluation. This information from source code, configuration, and policy files is abstracted and generalized in the subsequent Identification phase, and displayed in architectural views. Abstraction is based on security rules from a knowledge base. In the final Analysis phase the abstracted and generalized information is interactively assessed, augmented, and evaluated by the human inspector. To this end, the inspector is guided through different views where potentially harmful security issues as well as positive security features are marked. A more detailed description of SiSOA, especially of the Extraction and Identification phases together with technical details, can be found in Antonino, Duszynski, Jung, and Rudolph (2010).

In this article, we explain the SiSOA method and show how the knowledge base fits into our SiSOA methodology. In addition, we briefly describe our prototype tool that implements SiSOA including the knowledge base and provides support for semi-automatic security evaluation of SOA systems. This includes the description of our security estimation values: severity and credibility.

MODEL EXTRACTION

The purpose of the Extraction phase is to create a model of the analyzed system that stores all basic information necessary for further analysis steps. This model is called system model; it is constructed by using reverse engineering techniques (Chikofsky & Cross, 1990). The system model contains information about diverse software artifacts such as classes, packages, relations between classes or packages, and any other structural information that may potentially contribute to further security analysis. The input for building the model is the source code of the evaluated system and some complementary artifacts such as SOA configuration files.

The system model is based on the Eclipse Modeling Framework (EMF) (Steinberg, Budinsky, Paternostro, & Merks, 2008). In our prototype implementation, we reuse two existing EMF models: One is from Apache Tuscany (Davis, 2009; Laws, Combellack, Feng, Mahbod, & Nash, 2011; The Apache Foundation, 2011), a Service Component Architecture (SCA) (OSOA, 2011) implementation; the other is from SAVE (Duszynski, Knodel, & Lindvall,