# Securing Enterprise Applications: Service-Oriented Security (SOS)

Csilla Farkas and Michael N. Huhns
*Department of Computer Science and Engineering*
*University of South Carolina*
*Columbia, SC 29208 USA*
*{farkas,huhns}@engr.sc.edu*

## Abstract

*Some of the characteristics that make service-oriented architectures appealing for enterprise applications make them vulnerable to security breaches. The vulnerabilities are primarily due to the openness of the service-execution environment, to the dynamic run-time selection and composition of services, and to the autonomy of the individual services. In this paper, we describe these vulnerabilities, as well as traditional ones, and discuss ways of mitigating them. Such ways include software agent technology and distributed database transaction semantics.*

## 1. Introduction

A Web Service (WS) driven architecture provides a hardware and language independent way to support business processes. The main goal is to offer various services that can be accessed by diverse applications. Ongoing efforts address the needs to provide clear description and composition of services, and to support the management of services [2]. Service-Oriented Architecture (SOA), built on top of the WS paradigm, addresses these needs.

Along with the increased capability and flexibility of SOA applications, new security risks have emerged. Although Web data and application security research has come a long way, from the initial syntax-based XML security to a set of standards to support WS security, the security needs of SOA are still unresolved. Independent research efforts target specific aspects of SOA security without addressing comprehensive security needs of the SOA. This myopic view of securing SOA applications can easily lead to insecure SOA development. This paper is intended to conceptualize the current research directions in SOA security, identify research needs, and discuss uncharted territories that need to be addressed to achieve SOA security.

In particular, we argue that, while necessary, current, network-centric security measures are not sufficient to provide high-assurance security for SOA. We need to address service-level security needs. This include security needs of independent services (e.g., secure software development, correct execution) and their combinations (e.g., robustness, deadlock prevention, workflow requirements, correct execution history).

We argue that to successfully develop SOA security, each member service must be securely designed, developed, deployed, and maintained. Security vulnerabilities that were introduced during combination of services must also be addressed and appropriate mitigation strategies developed. Finally, based on the unique characteristics of SOA, we can deploy non-traditional security methods, like cooperation-dependent integrity verification, intrusion detection, etc.

After presenting a brief overview of current efforts concerning the problem of building secure and reliable software applications, we argue that secure software development requires that better practices be adopted and documented by all SOA application developers. We also show, how cooperative decision making can enhance application security and reliability. Finally, arbitrary combinations of services will also create not only insecure code (e.g., invoking an insecure service) but may also result in incorrect execution. We study potential risks, such as deadlocks, livelocks, and incorrect execution, and analyze the level of assurance provided by WS transactions.

## 2. Security for SOA

With the rapid increase of Web-based applications during the last decade, the need to provide security for them has emerged. Since XML became a basic construct of these applications, initial research focused on providing security and authentication for XML documents. As the application infrastructure became more-and-more complex, e.g., distributed and large scale systems, heterogeneous sources, work flow requirements, providing security for these applications became an increasingly challenging task. We group current security research into three main categories:

Network-level security: These efforts target security needs that arise from the distributed and open

nature of WS and WS-based SOA applications. Research directions include federated architectures, identity management, authentication, trust management, and secure communication.

Business-level security: These efforts aim to provide flexibility and ease of use to combine diverse services into combined services to fulfill business needs. Security needs range from the need to protect corresponding metadata to limit the information available for malicious users, e.g., partial disclosure of Web Service Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI) [7] specification, to the enhancement of business process composition languages to express security requirements. For example, the Business Process Execution Language for Web Services (BPELWS) is widely used to combine web services. Ongoing research extends BPELWS with authentication capabilities. Another direction of current research addresses some of the security and transactional properties of complex services.

Software-level security: These efforts aim to evaluate the security of the service code, including traditional software security practices, e.g., incorporating security needs in the Software Development Life Cycle (SDLC) activities and identifying new types of security vulnerabilities of SOA applications, e.g., collaborative verification of program correctness, detection of malfunctioning code/services, etc.

High assurance security can only be achieved if all three of the above aspects are considered. Any security model that addresses only one or two of these aspects, will be insecure. For example, while it is crucial to use strong encryption to protect network traffic, it is not going to be efficient if the attacker can easily gain access to the end systems due to software vulnerabilities, such as buffer overflow. In addition to the above three areas of SOA security, special properties of SOA must also be considered. For example, traditional security measures were developed for human-machine interactions, whereas SOA targets machine-to-machine interactions, thus embedding security-relevant materials in the applications themselves.

This paper focuses on the business-level security needs of SOA applications. Network-level security research has drawn lots of attentions during the last several years. Various standards and models, such as SAML, SOAP Security, WS-Security, WS-Policy, WS-Trust, WS-Federation, Identity management framework, XML security, etc. have been proposed. Software-level security is gaining momentum, where security considerations range from automated code analyzers to a series of best=practices guidelines. However, business-level security needs are not only less studied but often ignored or left to the discretion of individual developers.

## 3. Business-Level Security

To address business-level security efficiently, we must consider both the transactional properties of SOA applications and their security requirements. Although there is an emerging trend to address both of these issues, they remain isolated, vendor driven attempts. In this section, we give a brief overview of the current support for service compositions, service transactions, and potential threats against composite services.

### 3.1. Web Service Composition

Web services as individual components are intended to expose coherent units of functionality that can serve as building blocks for the more complex business needs of enterprises. One of the major features of SOA is that it enables a loose coupling of the individual services. The Web Services Business Process Execution Language, BPEL (also abbreviated BPEL4WS or WSBPEL) [9] is used to describe the composition of services and orchestrate their interactions. BPEL leverages other Web service standards such as Simple Object Access Protocol (SOAP) [10] and WSDL [7] for communication and interface descriptions. BPEL describes the inbound and outbound process interfaces in WSDL so they can be easily integrated into other processes or applications.

Web services can be combined in two ways:

- Orchestration
- Choreography

In orchestration, which is usually used in private business processes, a central process (usually another WS) takes control of the involved WSs and co-ordinates the execution of different operations on the web services involved in the operation. BPEL is the core language for process orchestration and handling fault tolerance or compensation actions. Web-service compositions describe the local process of service orchestration, Web-Services Choreography (WSC) describes the observable interactions between services and their users. WSC is more formally described by the W3C Web Services Choreography Working group as, "...the external observable behavior across multiple clients (which are generally Web Services but not exclusively so) in which external observable behavior is defined as the presence or absence of messages that are exchanged between a Web Service and its clients."

In a typical scenario, a BPEL business process first receives a request. To fulfill the request, the process

invokes the necessary WSs and then responds to the original caller. Because the BPEL process communicates with other Web services, it relies heavily on the WSDL description of the Web services invoked by the composite WS. BPEL consists of two different types of activities: primitive activities such as <invoke>, <reply>, and <assign>, and structured activities such as <sequence>, <flow>, <switch>, and <while>. However, BPEL does not provide any support for specifying security requirements.

### 3.2. Web Service Transactions

Transaction management has been studied extensively in database applications. Concepts, such as ACID (Atomicity, Consistency, Isolation, and Durability) transactions, locking methods, and serializability, have been successfully applied to guarantee correct execution of transactions. Can we apply similar concepts to improve the security of SOA applications?

BEA Systems, International Business Machines Corporation, and Microsoft Corporation, Inc, proposed a WS transaction framework [14] that addresses correctness criteria of service compositions. They propose two coordination types: atomic transactions (AT), or short-term transactions, and long-term business activity (BA) transactions. ATs are similar to traditional database transactions and are executed within limited trust domains. BAs incorporate business logic, support workflow systems, and may operate across different trust domains. Consistency is achieved without locking or a central control. Actual protocols are presented for both types, aiming for robustness and security, such as message authentication, integrity verification, and timestamps.

If business processes can be represented as a series of transactions [6,11,12,13], can we rely on traditional database transaction management concepts to provide security and consistency for SOA application? Is compensation-based transaction management [14] sufficient to guarantee correct execution history? What are the main differences between SOA composition and traditional database transactions? What additional capabilities of SOA can we employ to enhance the chance of correctness?

One of the main differences between SOA applications and traditional transactions is that in SOA, several compensating services may exist. This increases the robustness of the system and allows new approaches to be employed that verify correctness and security of service executions. In the following section we give an overview of some of these new approaches.

### 3.3. Service-Level Dependencies

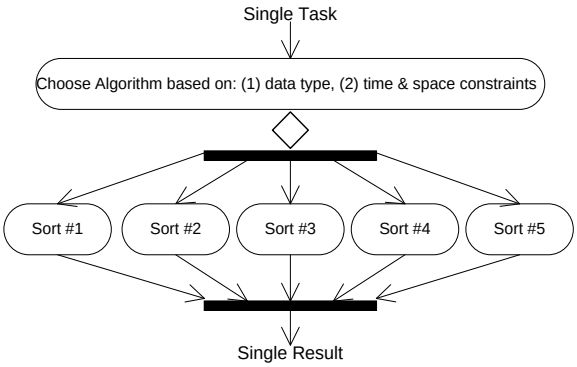


**Figure 1.** Centralized architecture for combining N versions of an algorithm into a single, more robust system

Threats, resulting from service-level (transactional) dependencies, affect both network-level and service-level security risk. Network-level threats include deadlocks, livelocks, information overload, denial of service, and network flooding, Service-level threats include service invocation overload, information starvation, and secure service compositions. These will take advantage of the following inherent characteristic of services: although developed and managed independently, services have dependencies when used as part of system-wide workflows. The services typically have no information about the workflows, and the workflows have no information about the internals of the services, their status, or any changes that might have occurred to the services. Basic network defense [1] will require and be based on an increase in the intelligence of each node in the SOA.

We are investigating a distributed network defense approach [3] that integrates the network, service, and software aspects of SOA security within a uniform framework. Our approach is different in kind from the current system of token verification, where breaking token verification gives system access. Distributed network defense depends on detecting and thwarting intruder's behavior. One protection approach is to use collaborative intrusion detection systems that pool knowledge of individual nodes across organizational boundaries to make a variety of simultaneous diverse approaches to intrusion detection [5]. Another uses software-based deception to deploy intelligent software decoys [4]. A variety of intelligent agents can be applied to detect intrusions. SOA nodes are more independent than nodes in standard systems to begin with and, moreover, must understand the messages they respond to at a deeper semantic level (beyond

XML to possibly using SAWSDL and OWL-S). Distributed network defense techniques should be more easily and effectively applied to SOA than to conventional systems. They can initiate, grow, and maintain a trust and reputation layer.

We propose a multiagent approach to handle independent versions of the software. This is done by wrapping or "agentizing" each algorithm. This produces a flexible and adaptable platform to handle multiple versions. A centralized approach, as shown in Figure 1, would use an omniscient preprocessing algorithm to receive the input data (demand) and would choose the best algorithm to perform the task. Each module's characteristics would have to be encoded into the central unit. The central unit could use a simplistic algorithm for determining best, based on known facts about each of the modules. The difficulties with this approach are (1) the preprocessing algorithm might be flawed and (2) its maintenance is difficult as new algorithms are added and existing algorithms become unavailable. Also, only one module at-a-time executes, there is low CPU usage, and results are taken as-is when completed.

## 4. Conclusions

In this paper we describe vulnerabilities of SOA applications and discuss approaches to mitigate security risk. We argue that current security solutions, focusing on network-centric security needs, are necessary but not sufficient to secure SOA. We call for a comprehensive approach to SOA security, ranging from secure code development to transactional properties of SOA. We also recommend novel security methods, such as software agent technology and distributed database transaction semantics, to enhance security capabilities. While there are some efforts in progress to provide comprehensive SOA security, additional work is needed. This paper is intended to encourage collaboration between academia and industry experts in defining and developing methods and techniques to achieve SOA security

## 5. References


[1] Distributed System Security Architecture, http://en.wikipedia.org/wiki/Distributed_System_Security_Architecture

[2] J. Epstein, S. Matsumoto, and G. McGraw, Software Security and SOA: Danger, Will Robinson!, Building Security In, 2008.

[3] D. Frincke and E. Wilhite, "Distributed Network Defense," *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, United States Military Academy, West Point, NY, 5-6 June, 2001

[4] J. B. Michael, N. C. Rowe, H. S. Rothestein, M. Auguston, and D. Drusinsky, "Phase I Report on Intelligent Software Decoys: Technical Feasibility and Institutional Issues in the Context of Homeland Security," Defense Technical Information Center No. ADA410039, December 2002.

[5] Morton Swimmer, "Using the danger model of immune systems for distributed defense in modern data networks," *Computer Networks*, vol. 51, no. 5, 11 April 2007, pp. 1315-1333.

[6] M. Verma, "Web services transactions," IBM Publications, 2005.

[7] W3C Schools, WSDL and UDDI, http://www.w3schools.com/WSDL/wsdl_uddi.asp, 2008.

[8] IBM, Business Process Execution Language for Web Services version 1.1, 2007, http://www.ibm.com/developerworks/library/specification/wsbpel/ , 2008.

[9] OASIS Web Services Business Process Execution Language (WSBPEL), 2007, http://www.oasisopen.org/committees/tc_home.php?wg_abbrev=wsbpel , 2008.

[10] W3C, Simple Object Access Protocol (SOAP), 2007, http://www.w3.org/TR/soap/ , 2008.

[11] OASIS Web Services Coordination (WS-Coordination), 2007, http://docs.oasis-open.org/wstx/wscoor/2006/06 , 2008.

[12] OASIS Web Services Atomic Transaction (WS-AtomicTransaction), 2007, http://docs.oasis- open.org/ws-tx/wsat/2006/06 , 2008.

[13] OASIS Web Services Business Activity (WS-BusinessActivity), 2007, http://docs.oasis-open.org/ws-tx/wsba/2006/06 , 2008.

[14] W. Cox, F. Cabrera, G. Copeland, T.Freund, J. Klein, T. Storey, S. Thatte, Web Services Transactions, BEA Systems, International Business Machines Corporation, Microsoft Corporation, Inc, 2004.