Security in Service-Oriented Architectures: Challenges and Solutions

Varvana Mylläriemi
Helsinki University of Technology, P.O. Box 9210, 02015 TKK, Finland
varvana.myllariemi@tkk.fi

Abstract. Service-Oriented Architecture (SOA) has gained popularity in constructing enterprise IT architectures. However, the principles of SOA do not explicitly address security. This paper presents a literature study on the challenges that SOA poses to security, as well on existing solutions for ensuring security in SOA.

1 Introduction

Traditionally, enterprise IT architectures have been application-centric, poorly interoperable, and too rigid for business process changes. The paradigm of service-orientation promises to deliver flexible, reusable, process-oriented enterprise IT architecture. In general, Service-oriented Computing (SOC) provides the foundations for building Service-Oriented Architectures (SOA). Web services have become de facto implementation technology for service-orientation: Web services use internet for communication, SOAP for interacting with messages, WSDL for defining services, and BPEL for orchestrating services from process point of view [1].

Originally, the principles of SOA have been shaped to address, e.g., interoperability, discoverability, loose coupling, and reusability [2]. As a consequence, primitive SOA principles do not address and may even hinder other quality attributes, such as security, performance, or reliability. Although contemporary SOA tries to improve the situation to the level of more established architectural styles, it is still striving to fill the gaps in such quality attributes [2].

For large enterprises, security and securing their enterprise architectures is of utmost important, since their essential data are at stake [3]. A large number of different and partly competing specifications and proposals have been presented to addresses security in SOA and particularly in Web services framework. Despite this, lack of security is one of the main reasons given by people who are reluctant to use Web services even knowing of their advantages [4]. It has even been argued that enhanced connectivity and flexibility of Web services have come at the cost of reopening the attack paths closed by previous software architecture security solutions [5].

This paper presents a literature study on SOA and security. Firstly, this paper discusses challenges that are due to SOA characteristics. Secondly, this paper
describes a number of solutions that have been proposed to address security. Although some considerations are applicable to SOA in general, most concrete solutions stem from Web service framework, in particular various WS-* extensions. Instead of describing the solutions from specification point of view, the discussion is organised along the security concerns and generic solution schemes that address these concerns.

The rest of this paper is organised as follows. Section 2 discusses challenges to achieving secure SOA. Thereafter, Section 3 describes a number of solutions that have been proposed to address the challenges. Section 4 discusses the results of the literature study and summarises the solutions. Finally, Section 5 concludes the paper.

2 Security Challenges in SOA

Before discussing the security challenges related to SOA, we establish what security in general means. According to ISO/IEC 9126 [6], security is the capability of the software to protect information and data so that unauthorised persons or systems cannot read or modify them and authorised persons or systems are not denied access to them. That is, security is a composite of confidentiality, integrity and availability [7]. Confidentiality means absence of unauthorised disclosure of information; integrity means absence of unauthorised system alternations; and availability means readiness for correct service for authorised actions [7]. These definitions highlight the need for authorisation of persons or systems; such persons and systems are often called principals. However, in order to be able to relate allowed actions to principals, the principals must be authenticated. Authentication means that the principals provide and verify an identity claim.

As discussed in Section 1, the principles of SOA have not been designed to primarily address confidentiality, integrity, and availability, or to ease authentication and authorisation. In the following, we discuss how some characteristics of SOA affect these aspects of security.

2.1 Loose Coupling

Services must be designed to interact without the need for tight, cross-service dependencies [2]. One of the means of adding indirection is to use messages as units of communication instead of direct operation calls. In Web services framework, such messages are specified by SOAP protocol and represented in XML [2]. Typically, messages are transferred from service to another via enterprise service bus (ESB), and messages may be routed via intermediary services. Further, the requirement that services are stateless means that more intelligence is added to a message [2]. This all underlines the need to ensure the integrity and confidentiality of messages, since they may contain possibly sensitive information that needs to be protected. According to [1], a major research challenge is to be able to validate and ensure end-to-end security.
Further, using XML for messaging has its own drawbacks, since XML has several security vulnerabilities. Various vulnerabilities related to XML are discussed in [8–10]. For example, an attacker can craft XML data causing the XML to call upon itself repetitively, thus causing a memory overflow. Using SOAP messages, it is possible to perform various XML injections, such as injecting SQL queries.

2.2 Composability

In SOA, services are composable and reusable elements [2]. This means that functionality of one service can be easily composed with functionality from another service, possibly indirectly through a third service. Since services are composable and reusable, they should not make any assumptions about the context in which they are going to be used [9]. This means that there cannot exist an application-specific security model, which is the way traditional application-centric enterprise architectures have tackled with security [9]. On the other hand, service-specific security models are not enough, since security is not just about protecting single services, but security threats need to be evaluated and addressed at the level of the process as a whole.

2.3 Openness

The nature of loose coupling and modularity in SOA as well as openness of Web service paradigm enables the use of SOA for enterprise architectures that cross organisations. For example, Web services allow virtual enterprises consisting of a network of independent, geographically distributed entities to interact in order to achieve a common goal [11]. With Web services, private business processes can be exposed to partners through public composite Web applications [12]. In this sense, SOA promotes openness across enterprises. However, opening up business processes for partners in open Web means that security issues are bound to rise. Each new entry point from the outside to the enterprise architecture must be considered as a potential security hole [12].

2.4 Discoverability

SOA promotes service discoverability: services should allow their descriptions to be discovered by service requestors to make use of their operations [2]. However, combined with openness, discoverability opens new kinds of security threats. Given WSDL for defining services and UDDI for discovery of services, gaining the information one needs as a malicious attacker to perform a software exploit is even easier than before [3]. Further, WSDL as such does not provide a mechanism to separate some operations to be visible only to a local network and some operations to be visible to all [13]. Even if only public operations are discoverable to the outside world, it is possible for an attacker to try to scan WSDL operations from the endpoint to find also those operations meant for local networks only [13].
2.5 Process-Orientation

In SOA, services are composed together to perform a process, which in enterprise architecture automates a complete unit of work [2]. However, by focusing only on business processes and services that represent actions in business processes, it is easily forgotten that such actions need to access and process information [14]. Compared to a software architecture with an explicit data model, SOA does not explicitly address how information is stored and processed more than what is defined in the messages. When looking at the definition of security given in [6], security is about protecting sensitive information. Hence, it may be more difficult to implement and evaluate solutions that protect information in SOA.

2.6 Myriad of Technologies

SOA in itself is a relatively complex paradigm. To make matters worse, Web services framework is abundant with standards and specifications. From security point of view, this poses a challenge for enterprise architects. It has been argued that complexity is the enemy of security [15]. Complexity makes the enterprise architecture difficult to manage and susceptible for various vulnerabilities and faults.

Further, there are numerous vendors, each highlighting the security features provided by their products [3]. However, security is not a feature: having security software does not indicate to have software security [3]. Overreliance on security technologies is one of the pitfalls of security in software architectures: security is a process, not a product [15].

2.7 Interoperability

One the motivations of SOA was to promote interoperability. As a consequence, contemporary SOA fosters intrinsic interoperability through using open standards and supporting vendor diversity [2]. In contrast, security solutions typically constrain how services can interact. Despite this, security solutions must not hinder interoperability.

This aim is challenging for two reasons [9]. Firstly, most implementations do not support all possible choices enumerated in the standards. Secondly, even if all security solutions are standards-compliant, interoperability is not guaranteed, since standards allow administrators to make a lot of choices [9]. A list of incompatibility causes in security solutions is presented in [9].

3 Solutions for Security in SOA

Web service stack is filled with various WS-* extensions that address security in one form or another. For example, Erl [2] lists 14 security specifications that may be used as part of SOA. A problem with Web services security standards is that several organizations are involved in developing them and as a result there are many, and they are partially overlapping [4].
Fig. 1. Security solutions covered in this paper.
In the following, a number of solutions that address security concerns are discussed; these solutions as well as their relationships with each other are illustrated in Fig. 1. For each solution, related specifications are discussed. Some of the specifications can be used for addressing different security concerns; hence they will be discussed in several of the following subsections.

3.1 Confidentiality and Integrity

As discussed in Section 2, security is about protecting information and data from unauthorised reads and modifications. In SOA, confidentiality and integrity solutions have mainly focused on messages, omitting other kinds of data and information. Hence, the following discusses solutions that address message-level confidentiality and integrity. From this point of view, confidentiality means that no unauthorised party has seen the message throughout its message path [2]. In contrast, integrity means that the message contents have not changed during transmission [2].

Similarly to any internet communication, message exchange in SOA and Web services can be protected with transport-level security (Fig. 1). Examples of such means are Secure Sockets Layer (SSL) [9] and Virtual Private Networks (VPN) [16]. In some cases, using such transport-level security mechanisms is enough for ensuring confidentiality and integrity of messages. However, the problem with transport-level security is that it protects the message only at the transport level. To protect the message from intermediary services, message-level security is required [2].

The main specification for achieving message-level confidentiality and integrity for Web services is WS-Security [13, 9, 2]. WS-Security defines a header block in SOAP messages, so-called security header, to which information about used measures can be added [13]. Confidentiality can be achieved by encrypting messages (Fig. 1). For Web services, WS-Security combined with its extension XML-Encryption can be used. Since SOAP messages can pass through multiple services, WS-Security enables encrypting different parts of the messages for different receivers [9]. The encrypted fragment in the message is replaced with a special element containing the ciphertext and the used encryption method [13, 9]. References to these encrypted elements as well as possible keys are then added to the WS-Security header. [9]

Integrity can be achieved by signing messages with digital signatures (Fig. 1). For this purpose, Web services framework provides XML-Signature in conjunction with WS-Security: digital signatures are added to the security header. A challenge related to signing SOAP messages is that the algorithms used for calculating signatures do not understand XML syntax, but operate on sequences of input bytes. Hence two syntactically equivalent XML messages may get two different signatures. For this purpose, XML messages must be canonicalised. [9]

When any means of encrypting or signing messages, it must be established how keys and certificates are managed (Fig. 1). For example, [9] discusses how
WS-Security can be used in conjunction with shared secret keys, Public Key Infrastructure (PKI), and Kerberos; each strategy has its own requirements for the infrastructure.

XML-Encryption and XML-Signature only apply for one message at a time. To enable secure session (Fig. 1) between service requestor and service provider, WS-SecureConversation in conjunction with WS-Trust can be used. WS-SecureConversation describes the usage of a security context, which can be used for, e.g., sharing keys for the session [17]. In turn, WS-Trust specifies a security token service (STS), which issues tokens needed to establish a security context for the session [17]. In fact, security token service (Fig. 1) is one example of a security service [18, 9]; further examples are given in the next section.

3.2 Authentication

Section 2 discussed the need for being able to authenticate principals. Within SOA, authentication means that service requestor is able to verify a claim for identity [2]. Thus authentication (Fig. 1) involves providing an identifier, such as username or digital certificate, and an authentication token, such as password or hardware chip [9].

For Web services, WS-Security framework can be used for authentication: authentication claims can be attached to WS-Security header entry in the SOAP message. WS-Security supports many kinds of authentication technologies; these include using plain username and password, using username and password digest with a nonce and a timestamp, using Kerberos, and using digital certificates, such as X.509. [9].

Implementing authentication service-by-service basis in an enterprise architecture quickly becomes infeasible, since the authentication logic must be coded within each service. Instead, authentication can be provided as a separate security service [18, 9]; this solution is named authentication as a service in Fig. 1. Typically, authentication service is augmented with identity management as a service (Fig. 1), which deals with managing user identifiers and security-relevant user properties [18]. A reference architecture for an authentication service is presented in [18].

Firstly, an authentication service needs to communicate its decisions to the invoker of the authentication service [18, 9]. SAML (Security Assertion Markup Language) is a standard for encoding authentication statements as assertions [18]. Secondly, an authentication service needs to establish an interface that provides authentication operations to other services [9]. For this purpose, there are two specifications: WS-Trust and SAML protocol [9]. As discussed in Section 3.1, WS-Trust describes a security token service that issues security tokens. In constrast, if security service communicates its findings with SAML assertions, also SAML protocol can be used.

A challenge that is can be addressed with authentication as a service is single sign-on (SSO, Fig. 1), that is, how to propagate authentication within enterprise architectures. The use of SSO allows service requestor to be authenticated once and to have all services share this authentication. In such a scheme, service
requestor authenticates to an issuing authority service. Upon receiving service requests, service providers contact the issuing authority to ask for a clearance. As with authentication services in general, such a clearance can be expressed using SAML assertions. [2]

3.3 Authorisation

After the authentication has been verified, authorisation can take place. In SOA, authorisation (Fig. 1) means deciding whether the service requestor is allowed to access the operation in the service provider [2]. Hence authorisation is often called access control. Despite its importance, access control in Web services framework has not seen the development and adoption of many standards [19].

XACML (eXtensible Access Control Markup Language) is a standard that can be used for representing access control policies and access control decisions [10, 20]. XACML can be used to authorise requests to a single service, but it is also possible to construct authorisation as a service [18, 9] (Fig. 1). To enable an authorisation service, SAML assertions can be used to request permission to access resources. The service handling the access control policy consults a XACML policy to render a decision, which is returned as a SAML response [10]. Similarly to authentication, the interface of the authorisation service can be described using either WS-Trust or SAML protocol [9].

A challenge with authorisation comes from composing several services into one. A naïve approach is to start executing the composed service and resolve authorisation for each service separately [9]. However, this approach does not preserve atomicity of the composite service [9]. Ideally, before the composed service is invoked, the authorisation should be established. Since BPEL (Business Process Execution Language) can be used for composing services, it is possible to study from BPEL scripts whether business processes as a whole comply with access control policies; such an approach is presented in [21].

Further, distributed enterprise architectures make access control more challenging. A list of requirements for access control mechanisms that are suitable for collaborating enterprises is presented in [11]. To enable using XACML for independent enterprises, an algorithm for integrating several autonomous XACML policies is presented in [20].

3.4 Security Policies

Security policies (Fig. 1) enable codifying security solutions that impose rules and constraints on services and service interaction. There are several benefits from separating security policies from actual service implementations [9]. Within an enterprise, security policies can be used to ensure consistent use of security solutions, whereas between several enterprises they can be used to ensure interoperability, either at design time or at runtime [9].

Web services framework provides two specifications for codifying policies: WS-Policy and WS-SecurityPolicy. WS-SecurityPolicy is an extension to WS-
Policy that addresses policies related to WS-Security framework, whereas WS-Policy provides a way to combine several assertions into one [9].

Hence, security policies captured using WS-SecurityPolicy can govern confidentiality, integrity and authentication (Fig. 1), since WS-SecurityPolicy can state constraints in how incoming and outgoing requests must be encrypted, signed, and attached with authentication tokens. However, WS-Policy or WS-SecurityPolicy are not enough in capturing constraints on authorisation, since both lack the semantics necessary to meet the fine-grained access control requirements [19]. To overcome this, an integration of WS-Policy with an access control policy language is proposed in [19].

Although one of the potential benefits of explicit security policies is to enable run-time policy verification between enterprises, this goal has not yet been fully achieved [9]. In order for this to succeed, it should be possible to calculate an intersection of the security policies. While WS-SecurityPolicy allows the freedom to specify security requirements for the deployed services, policies defined according to this specification lack the semantics necessary for automated composition [22].

Finally, maintaining and writing security policies is a tedious task. To improve their usability, a tool that can be used for configuring security policies, from business scenarios to WS-SecurityPolicy descriptions, is described in [23].

3.5 Attack Prevention and Tolerance

SOA technologies inevitably contain security vulnerabilities. For example, a number of vulnerabilities related to Web services are discussed in [13], some of them being due to WS-Security protocol itself. Malicious attackers will try to exploit these vulnerabilities to succeed in various attacks.

As architectural solutions, attack prevention and tolerance in SOA can be achieved by checking messages and limiting access to a service (Fig. 1).

There are several checks that can be done to inbound and outbound messages, some of which are discussed in [5, 13]. For example, input arriving via messages can be checked against appropriate XML schema [13] and sanitised against various injection vulnerabilities [5], and too long inputs can be truncated to prevent buffer overflows [5].

Access to a service can be limited in many ways. Firstly, operations meant for internal and external use should be deployed to separate Web services [13]. Secondly, internal and external services should be deployed to different physical nodes. Firewalls and De-Militarised Zones (DMZ) are well-known tactics in software architectures to limit the access among different deployment nodes (see e.g. security tactics in [24]). In the context of SOA, both approaches can be used for preventing attacks as well as limiting damages in case of exploits [9]. For example, [12] illustrates how DMZ created by firewalls can be used to protect a situation in which some services need to published to the open internet. However, it should be noted that a firewall as such does not make SOA secure [3]. Because Web services tend to be implemented inside the organisation, worry about their inherent security is often discounted; however, the fact remains that
most firewalls will simply pass along a Web services request, including any attack code [3].

Attacks can also be prevented and tolerated by establishing security practices within several phases of SOA lifecycle (Fig. 1). For example, as new threats and vulnerabilities constantly emerge, the organisation should establish continuous practices for monitoring and patching vulnerabilities [9]. Further, there is a need to establish practices for assessing and testing security threats. However, there is a lack of security testing standards and testing specifications for testing exploit patterns for Web services [5]. To overcome this, a vulnerability analysis model for evaluating and testing various faults is presented in [5].

4 Discussion

Section 3 discussed a number of proposed security solutions for SOA and Web services in particular; Fig. 1 summarises the solutions.

In [9], SOA security solutions are categorised as message-level security, policy-oriented security, and security as a service. However, there are several problems with this classification. Firstly, some of the solutions in Fig. 1 do not fall into any of these categories, e.g., secure sessions. Further, this classification is not orthogonal, since a security service can utilise message-level security solutions.

Therefore, we propose to classify the approaches discussed in Section 3 to solutions that govern message-level issues (e.g., WS-Security), solutions that govern service-level issues (e.g., WS-Trust), solutions that govern several services in the architecture (e.g., deployment with firewalls and DMZ), and solutions that govern management issues (e.g., vulnerability practices).

Most of the solutions that have been proposed to address security in SOA are rather low-level and close to the technical implementation of the service. If focus is on ensuring message-level or even service-level security, big picture is easily lost. Fine-grained security does not mean that the architecture as a whole is secure. This situation resembles one of the pitfalls of security in software architectures, which is labeled as piecemeal security in [15]. Security in SOA not just about secure messaging, or about single-service security, but end-to-end security for the process as a whole. Therefore, it is surprising that approaches address security at the business process level have not yet received that much attention. Further, besides purely architectural solutions, there is a need for establishing security management practices within SOA.

Finally, it is worth noting that the SOA security solutions do not as such solve all challenges outlined in Section 2. Although standards and solutions are important, they do not actually make the system secure: an implementation bug or an architectural flaw in a product can leave a system that is completely standards-compliant completely insecure as well [3]. Hence, good security engineering practices for both architecture design as well as implementation are mandatory in order to achieve secure SOA.
Conclusions

This paper presented a literature study on security within service-oriented architecture (SOA). Firstly, the paper discussed some challenges to security that are due to characteristics of SOA that aim for flexible, composable, process-oriented enterprise architectures.

Secondly, the paper presented some solutions to ensure confidentiality, integrity, authentication, authorisation, security policies, and attack prevention and tolerance. Although the solutions cover these concerns, they concentrate on rather low-level technical issues. Many of them address message-level or service-level security, and do not address security at the level of the whole enterprise architecture or at the level of management.

Therefore, future work is needed to establish a link between low-level technologies and specifications and higher-level business processes and architectural best practices.

References