Securing Code in Services Oriented Architecture

Emilio Rodriguez Priego and F.J. García

Departamento de Matemáticas y Computación
Universidad de La Rioja
Edificio Vives, Luis de Ulloa s/n
E-26004 Logrono (La Rioja, Spain)
{emilio.rodriguez, francisco.garcia}@unirioja.es

Abstract. SOA proposed security mechanisms are only centered in the data transmitted between service provider and consumer. However, it's well known that the biggest threats to the integrity of the information are precisely focused not on the data directly but on the code that manages it. Our main statement is that it will only be possible to reach an acceptable level of security if the protection mechanisms cover not only the data but also the code that process these data. In this paper we present a new approach about mobile code security based on the Services Oriented Architecture Reference Model and Web Services technology. This new model allows the development of systems with end-to-end security, where all elements (code and data) are secure.

Keywords: Web services security, mobile code security, Service Oriented Architecture.

1 Introduction to the Problem

Nowadays, there is a growing interest in Web Services technology and Service Oriented Architecture (SOA), whose Reference Model has been recently approved as an OASIS standard [2]. In this model, a consuming entity requests from a supplier entity, one or more services under a set of conditions (interaction, visibility, execution context, policies and contracts). At the same time, security is one of the aspects that requires more attention due to the application of this technology to environments where information exchange is made through public networks like Internet, in which there potentially exists diverse security threats. Recently different standard-based solutions have been proposed to solve the problem of secure sending and reception of messages. According to SOA, the performed action details of the supplied service are not typically visible by the consumer [1, 2]. Therefore SOA does not address the subjects related to the realization of a service, like, e.g. securing it, delegating the effective resolution of these problems to the supplier.

The proposed security mechanisms of the SOA model and the technology of web services are centered in the transmitted data (message), and they put their focus on end to end integrity, confidentiality, identity and authentication. These mechanisms work well and in practice they reach the objective. However, it's well known that the biggest threats to the integrity of the information are precisely focused not on the data directly but on the code that manages it [8,9].

L. Baresi, P. Fraternali, and G.-J. Houben (Eds.): ICWE 2007, LNCS 4607, pp. 550 – 555, 2007.
http://www.springerlink.com/content/m210g77836215101/

Therefore, our main statement is that it will only be possible to reach an acceptable level of security if the protection mechanisms cover not only the data but also the code that process these data.

Independently of the web services development and SOA model, the security problem of mobile code and its interaction with the environment in which code is executed has been studied in the last years, particularly for the singular case of mobile agents [6]. Both SOA and mobile code have specific and common aspects of security but until now had not been treated jointly. The main contribution of this article is the proposal of a new approach to the problem of the security of mobile code, named "Web Services based Secure Code" (here-in-after WSbSC) that it's based on SOA Reference Model and the Web Services Architecture.

WSbSC allows for the improvement of security in a typical interaction between consumer and provider without forcing the participants to necessarily know the details of implementation of the services.

Our model is virtually applicable to any SOA situation in which an integral model of security, involving data and code was required. However, in certain situations code visibility, integrity and/or portability are more important, because code integrity, the code in itself, the source of the code or all of these elements, take part or are exchanged as part of services provided by the supplier. Typical examples of these application environments are distributed processes, hosting or rent of processes, auditing and validation of code by certification entities, and so on.

Once the problem that we want to address has been stated, the rest of the paper is organized in two main sections, each one describing its own objectives, resolution outline and methodology. In Section 2, we provide a general description of the WSbSC reference model. Section 3 presents the application of the model to a basic SOA interaction. The last section summarizes the main contribution, related work, the status of the research and indicates the future work.

2 WSbSC Reference Model

The reference model of WSbSC (here-in-after WSbSC-RM) is an abstract framework for understanding significant relationships among service entities (providers and consumers) that allows an integral (data and code) secure interaction, enabling the development of specific architectures using consistent standards or specifications.

WSbSC-RM relies on SOA-RM and it adds new concepts and relationships to the modeling of data and code exchange based on services.

The central concept in WSbSC-RM is the code, just as it has been defined traditionally but with some specific features: (1) the code can be portable: i.e., it can be sent from one system to another without manual intervention; (2) the code can be executed in any compatible execution environment; (3) transmission, load and execution of the code can be carried out in a safe way, applying the same basic principles of secure transmission of data (identity, integrity and confidentiality); and (4) the code can be verified in a secure manner.

There have been different proposals related to code portability, validation and execution in distributed environments [4,5,7]. Most proposals are based on hardware or software techniques for execution in a local environment.

WSbSC-RM states that the transmission, reception, execution, load, compilation and validity of the code are *services* that can be offered by systems potentially remote and weakly coupled. As we see below, with WSbSC *code is not only externally verifiable* [3]*, but also externally compilable and externally executable.*

WSbSC-RM distinguishes the following actors:

- *Author*: it's the owner and creator of the code and its legal owner.
- *Supplier*: provides the code to a consumer and distributes it by author permission.
- *Client*: uses the code provided by a supplier.
- *Verifier*: verifies the code according to a security policy previously established.
- *Compiler*: given a code, it compiles another functional equivalent code.
- *Processor*: possesses an execution environment that executes the code.

All WSbSC-RM actors are service consumers or providers, from the point of view of SOA-RM. Besides WSbSC-RM allows the modeling (recursively) of the actions (local or remote) of a service by composing services offered by these actors and according to code-centric policies and contracts. What is a key added factor of our approach with reference to SOA, is that actors playing the role of consumers in any relationship to a provider may impose a certain security policy to regulate the service that the provider is going to perform. This policy, and here is the contribution, does not only affect the data (message) as SOA does, but also the service implementation. This policy refers to one or several security aspects (such as integrity, confidentiality, validity, and so on) and may specify a mechanism or set of mechanisms that the provider must implement to accomplish the policy. These security requirements can be used by the consumer to select the most suitable provider in each case, depending on the mechanisms the former can implement. The response to each service request will include, as well as the result, metadata about the required, and fulfilled, policy.

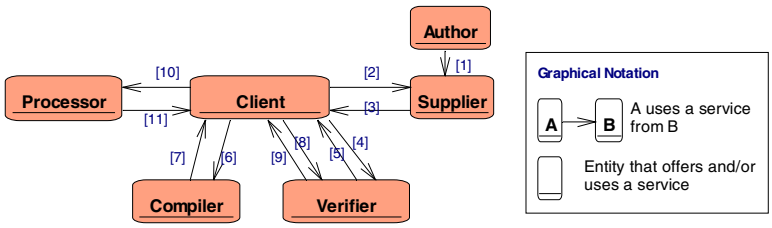

Fig. 1. General WSbSC-RM scenario

At this point, we have not studied yet all relationships between WSbSC-RM actors like service actors, but we can illustrate these relationships by describing a general example that shows some key concepts in SOA-RM: service, interaction, policies and contracts, real world effect, etc. Fig. 1 shows this general scenario. Interactions involve the following steps: (1) An *author* creates the code and sends it to a *supplier* for distribution. (2) A *client* localizes and requests the code that satisfies its needs from the *supplier*. (3) The *supplier* delivers the code. (4) The *client* requests the verification of the code according to the *client* policy from a *verifier*. (5) The *verifier* delivers the validated code. (6) If code is not compiled for the architecture in which is going to be

executed then the *client* requests its generation from a *compiler*. (7) The *compiler* returns the compiled code. (8) The *client* can request validity of the compiled code from the *verifier*. (9) The *verifier* returns the validated code. (10) The *client* requests to a *processor* execution of the code. (11) The *processor* returns the result of the process to the *client*.

At point (9) the code is associated to the *verifier*'s signature that guarantees its integrity. The *Processor* can verify code integrity, or even correctness with respect to a certain specification, before execution by means of that signature. Moreover, the overall process can be checked if each actor signs its action. As a result, each step generates metadata signed by the service provider, as well as its signature; e.g., the result code of the *compiler* can include metadata related to that compilation. This means that at the end of the process we can get a code qualified as "secure" since it's created (*author*), provided (*supplier*), validated (*verifier*) and generated (*compiler*) by trusted identified entities. This code, that we'll name Portable Secure Code (PSC) is formally "portable" and "secure". We have that PSC = Code + PSC-cert (cert stands from certificate). As you can see in Fig. 2, the PSC-cert accompanies the service result returned to the service consumer. This PSC-cert allows a client to test that the code is PSC while the code itself is not revealed.

There can be diverse variations of this general scenario. For example, after the step (2), the *client* asks the *processor* for the execution of the code and the *processor* manages the communication with the *verifier* and the *compiler* to get the PSC.

We will use the following methodology to develop the model outlined here: (1) we will describe in detail concepts and relationships among actors in the model and (2) study in more detail the relationships with SOA and Web Services Architecture.

3 Service Implementation by WSbSC

In this section a specific use of WSbSC to offer an advanced level of end-to-end service security is described. We consider a consumer entity that uses a service offered by a provider entity. Fig. 2 shows how a provider entity relies on WSbSC to get a higher security level.

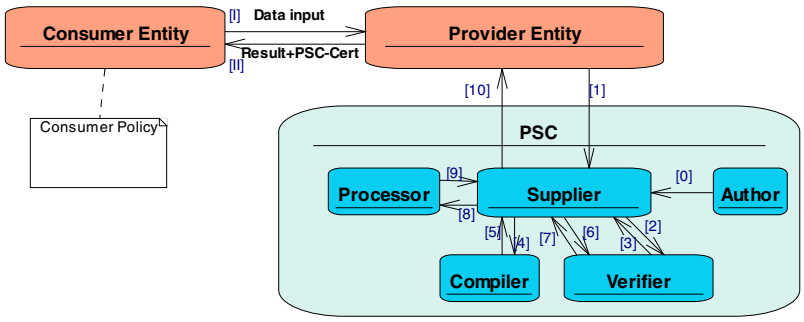

Fig. 2. Service Implementation with WSbSC

Applying WSbSC, interactions in this basic scenario are: (I) a *consumer entity* requests a *service* from a *provider entity*. The security policy of *consumer entity* establishes that the *provider entity* must implement the service using WSbSC and, therefore the *provider entity* must get a PSC of the service implementation to achieve this policy. (0-9) The *provider entity*, if it hasn't done so previously, carries out the process of creation, supply, validation and generation of the PSC and executes the service. (II) It returns the result of the service together with the PSC-cert. Note that (0-9) illustrate another case in which the *provider entity* delegates in the *supplier* the process to get the PSC.

By means of PSC-cert, the consumer entity can obtain an extra security level that certifies that the service was created, supplied, verified, compiled and executed by trust entities without being necessary to know the code itself. There can be diverse variations: e.g., the *provider* entity could submit a copy of the code, encrypted with the verifier's public key. If we suppose that both the *consumer* and the *provider* entities trust the *verifier*, then the *consumer* entity could request code validation to the *verifier*. Code confidentiality is guaranteed by encryption, and the *verifier*'s public key encryption guarantees that only the *verifier* can decrypt, analyze and evaluate the code validity. Periodical tests and the use of several verifiers can improve security even more. To illustrate the work to be performed, the following listing outlines the structure of a PSC-cert in a simple case.

```
<wsbsc:psc xmlns:wsbsc=.. xmlns:uddi=.. xmlns:ds..>
<wsbsc:code EncodingType="Base64">cHVibG..</wsbsc:code>
<wsbsc:psc-cert>
          <wsbsc:AuthoredCode> ..</wsbsc:AuthoredCode>
          <ds:Signature ..> ..</ds:Signature>
          <wsbsc:SuppliedCode> (a)</wsbsc:SuppliedCode>
          <ds:Signature ..> ..</ds:Signature>
          <wsbsc:CompiledCode>.(a).</wsbsc:CompiledCode>
          <ds:Signature ..> ..</ds:Signature>
          <wsbsc:VerifiedCode>.(a).</wsbsc:VerifiedCode>
          <ds:Signature ..> ..</ds:Signature>
</wsbsc:psc-cert>
</wsbsc:psc>
```

`AuthoredCode` block is added at point 0 together with autor's signature of that block. The following blocks are added in the same way by each actor when they have finished their tasks. Note that sections marked with `(a)` consist of two main parts: metadata related to the actor (description, e.g., by means of uddi business entity, credentials, e.g., SAML authorization credentials, etc.) and metadata about its action, e.g., for the compiler: the compiler environment, the target language, and so on.

This section has outlined one of the alternative interaction scenarios among consumers and suppliers. In order to finish developing this section, we will (1) study more alternative interaction scenarios, (2) select the most suitable web services standards to implement PSC, (3) develop the WSbSC security model extending the WS-SecurityPolicy model, and (4) we will develop an actual case relevant enough to illustrate the different alternative scenarios.

4 Contribution, Related Work, Status and Future Work

Several solutions about mobile code has been proposed in the last years: e.g., [4,5,6] focus on execution environment (compiler, verifier and/or processor). [7] and more recently [10] suggest a contract between producer and consumer of mobile code. [11] defines a model-driven approach for service-oriented software development.
The main contribution of this paper is the proposal of a new approach to the problem of the security of mobile code, WSbSC, that it's based on SOA-RM and the Web Services Architecture. WSbSC provides a level of security that covers not only the data but also the code that process these data.

A lot of work must be done, both to specify the demanded policy and the process that must be preformed to implement it (perhaps using WS-SecurityPolicy or even BPEL), and in order to select the standards to be used at each part of the PSC-cert (SAML, WS-Policy, WS-Addressing, UDDI, and so on). As a future line of research we are planning to extend the model to portable objects, i.e., securing the object state as well as the code that manages that state (behavior), making this code PSC.

Acknowledgments. Partially supported by Comunidad Autónoma de La Rioja, project ANGI-2005/19.

References

1. Web Services Architecture (February 2004) http://www.w3.org/TR/ws-arch/
2. Reference Model for Service Oriented Architecture v1.0 October 2006 http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf
3. Seshadri, A., Luk, M., Perrig, A., van Doorn, L., Khosla, P.: Externally verifiable code execution. Communications of the ACM (September 2006)
4. Franz, M., Chandra, D., Gal, A., Haldar, V., Reig, F., Wang, N.: A portable Virtual Machine target for Proof-Carrying Code. In: Proceedings of the 2003 workshop on Interpreters, virtual machines and emulators (June 2003)
5. Yau, S.S., Prasad, A., Zhou, X.: An Object-Oriented Approach to Containing Mobile and Active Codes in Large-Scale Networks, words. In: Fourth International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'99) (1999)
6. Claessens, J., Preneel, B., Vandewalle, J.: How can mobile agents do secure electronic transactions on untrusted hosts? A survey of the security issues and the current solutions, ACM Transactions on Internet Technology (TOIT) (February 2003)
7. Sekar, R., Ramakrishnan, C.R., Ramakrishnan, I.V., Smolka, S.A.: Model-Carrying Code (MCC): a new paradigm for mobile-code security. In: Proceedings of the 2001 workshop on New security paradigms (September 2001)
8. Whitman, M.E.: Enemy At The Gate: Threats To Information Security. Communications of the ACM (August 2003)
9. Sima, C.: Are your web applications vulnerable? (October 2004) http://www. securitydocs. com
10. Security of Software and Services for Mobile Systems (March 2006) http://www.s3ms.org
11. SENSORIA (October 2004) http://sensoria.fast.de/