INTEGRAL SECURITY MODEL FOR THE EXCHANGE OF OBJECTS IN SERVICES ORIENTED ARCHITECTURE

Emilio Rodriguez-Priego, Francisco J. García-Izquierdo
Universidad de La Rioja (Dpto. de Matemáticas y Computación), Edificio Vives, Luis de Ulloa s/n.E-26004 Logroño
emilio.rodriguez@unirioja.es, francisco.garcia@unirioja.es

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

Abstract: Nowadays, security approaches and solutions for SOA focus mainly on messages and data, but they forget the code security (both service code and exchanged code). Moreover, some security aspects (e.g. validity, correctness...) are usually forgotten. We state that any security approach will be incomplete if the security of both data (messages) and code (service code) is not addressed in a general sense. In this paper, we extend a previous approach about securing code in SOA. We analyze general problems related to the exchange of code and state in SOA and in the specific case of Web Services architectures. A new general model of security is presented. This model covers any aspect related to the authorship, distribution, transformation, execution and validation of both code and data.

1 INTRODUCTION

The current interest in Web Services technology and Service Oriented Architecture (SOA) (OASIS, 2006, OASIS, 2008) is notorious. One critical aspect in SOA is security. SOA and Web Services Architecture (WSA) (W3C, 2004) share common concepts. WSA can be seen as a specific application of SOA model. Currently, most works related to SOA are developed under WSA. Many web services standards are evolving, most of them about security.

SOA-RM focuses on the service as a central concept of interaction between a generic consumer and a provider. It defines service as a "mechanism to enable access to one or more capabilities" and "its implementation is typically hidden from the service consumer except for (1) the information models exposed through the service interface and (2) the information required by service consumers to determine whether a given service is appropriate for their needs".

Policies and contracts are other SOA central concepts, particularly related to security policies. Most SOA based solutions are based in web services technologies and security mechanisms. They put their focus on end to end integrity, confidentiality, identity and authentication. These mechanisms work well and, above all, they are applied to messages. However, code security is not generally addressed, despite being well known that most threats to the integrity of the information are precisely focused not on the data directly but on the code that manages it (Whitman, 2003).

This paper is the continuation of a previous work (Rodríguez and García, 2007) that presented an approach about code security on SOA environments that can be applied on web services architectures. We named that approach "Web Services based Secure Code" (here-in-after WSbSC) and its reference model WSbSC-RM. This paper extends it considering more complex situation in which entities wish to exchange not only data, but also code to manage that data. So we are now considering objects (data + code) as input and/or output of services.

The remainder of the paper is organized as follows. Section 2 is devoted to show some background necessary to understand the rest of the paper. Section 3 describes how WSbSC can be extended from an object-oriented perspective. Section 4 covers implementation issues. The paper finalizes with the conclusions and a future work outline.

1.1 Related work

A considerable amount of related work has been done. (Rubin and Geer Jr. 1998, Claessens, Preneel and Vandewalle, 2003) studied security aspects about Mobile code and Mobile agents and diverse solutions for specific security threats were proposed. Besides, each one of the execution environment actors that appear in this paper –compilers and processors, e.g., virtual machines (Franz et al, 2003) and verifiers (Chang et al, 2005, Bhargavan, Fournet and Gordon, 2004) – have been presented from different points of view. Security contracts and policies were analyzed in (Gutiérrez et al, 2005, Sekar et al, 2001) and more recently in (European Project, 2006). (Foster et al, 2008) addresses the object state modelling with Web Services technologies but without considering security aspects. An old debate about distributed objects vs. Web services underlies this paper (Vogels, 2003, Birman, K.P., 2004).

2 WSBSC-RM AND PSC-CERT

To make the rest of the paper self contained we are devoting this section to present some background already published in (Rodríguez and García, 2007).

WSbSC-RM is an abstract framework for understanding significant relationships among service providers and consumers that allows an integral (data+code) secure interaction. WSbSC-RM relies on SOA-RM, adding new relationships and concepts to the modelling of the data and code exchange between services. A key concept in WSbSC-RM is code: it can be portable, executed in any compatible execution environment; the transmission, load and execution of the code can be carried out in a safe way, and it can be verified in a secure way. WSbSC-RM states that the transmission, reception, execution, load, compilation, validation of the code are services that can be offered by systems potentially remote and weakly coupled. In WSbSC code is not only externally verifiable (Seshadri et al 2006); it's also externally compilable and executable.

WSbSC-RM distinguishes these actors: *author*, the owner and creator of the code and its legal owner; *supplier*, provides the code to a consumer and distributes it with author's permission; *client*, uses the code provided by a supplier; *verifie*r, verifies the code according to a established security policy; *compiler*, given a code, it compiles another functional equivalent code; and *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 modelling of the actions 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. And this policy, and here is the contribution, does not only affect the data and the message as SOA does, but also any aspect of 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 a set of mechanisms that the provider must implement in order to accomplish the policy. The response to each service request will include, as well as the result, metadata about the required, and fulfilled, policy.

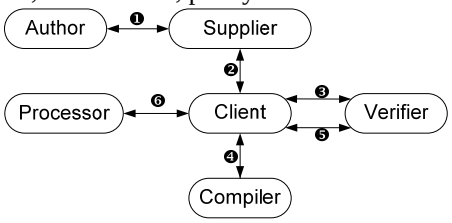

Figure 1: General WSbSC-RM

We can illustrate the WSbSC-RM relationships between actors describing a general example. Fig. 1 shows this general scenario. Interactions involve the next 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 and the supplier delivers the code. (3) The client requests the verification of the code according to the client policy from a verifier and the verifier delivers the validated code. (4) If code is not compiled for the architecture in which is going to be executed then the client requests its generation from a compiler. The compiler returns the compiled code. (5) The client can request validity of the compiled code from the verifier. The verifier returns the validated code. (6) The client requests to a processor execution of the code and the processor returns the result of the process to the client. At point (5) the code is associated to the verifier's signature that guarantees its integrity. By means of that signature the processor can verify code integrity, or even correctness with respect to a certain specification

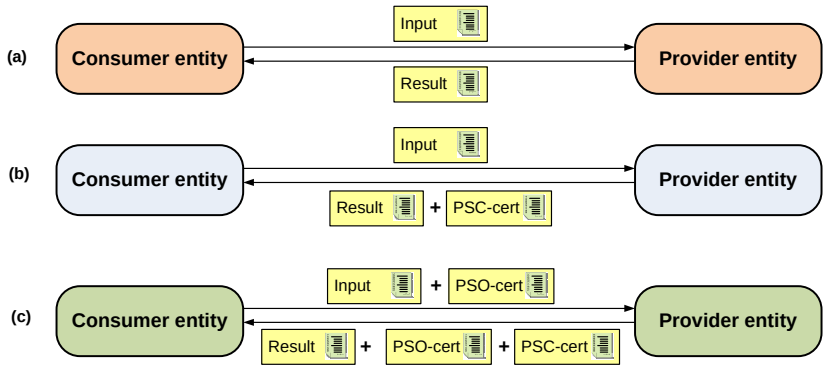

Figure 2 WSbSC-RM Information Model Cases

before execution. Moreover, the overall process can be checked if each actor signs its action.

Therefore, each step generates metadata signed by the service provider, as well as its signature; e.g., the code that results from 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 is 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). PSC-cert can be considered as a “metadata security container” that enables security exchange between entities providing and using the certified code.

The following listing outlines the structure of a simple PSC-cert. AuthoredCode block is added together with author’s signature of that block. All blocks are added in the same way by each actor when they finish their tasks. Sections marked with (*) consist of two main parts: actor related metadata related (description, e.g., by means of UDDI business entity; credentials, e.g., SAML authorization credentials ...) and metadata about its action, e.g., for the compiler: the compiler environment, the target language, and so on.

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

3. INFORMATION EXCHANGE SCENARIOS

In the next sections we describe a specific use of WSbSC-RM to offer an advanced level of end-to-end service security using objects as elements exchanged, assuming that an object is data + code. Fig. 2 shows the general picture of the three basic scenarios where a consumer requests a service from a provider. In case 2.a, the consumer and the provider exchange only data. This is the common information model found in SOA. In it, and for the particular case of Web Services, common security mechanisms such as public-key cryptography can be used to securing only the exchanged data (WS-Security, SAML, XML-DSIG, WS-Policy…). Case 2.b corresponds to a scenario described in section 2.

So far, we have proposed a model in which the secured code is located in the service provider. The next step in the discussion is the study of a scenario where the code travels between consumers and service providers. Code may travel and not be locally executed, e.g., for performance reasons (Lange and Oshima, 1999). Case 2.c depicts this new scenario. The service requires as input an object from the consumer. During its execution, the service will interact with this object and eventually it will modify its state. The object helps the service to fully accomplish its mission, allowing the service to use its methods. E.g., imagine that the scenario corresponds to the interaction between a flight reservation service and final users, who send their personal planner object. The service needs to consult the planner calendar to determine the user

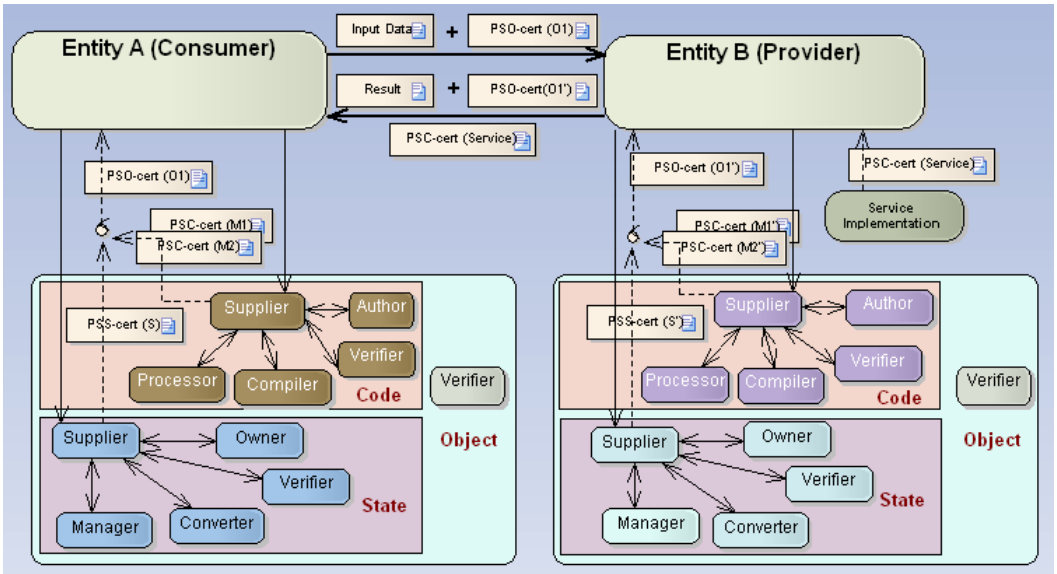

Figure 3 Portable and secure objects with WSbSC

availability, and, once the flight has been selected, the service inserts a new appointment in the planner.

Security requirements in case 2.c are more complex than for case 2.b, because the consumer and the provider want to ensure that not only the service (case 2.b) but its interactions (use and/or modification) with the input object are secure. So, they agree to use WSbSC to get a higher level of security about the performed service. But this is not enough yet. The security policy of the consumer requires that the provider accesses and/or modifies the object state only by means of the object methods.

Fig. 3 shows how we can achieve these high level security requirements for this scenario using WSbSC. Interaction begins when the consumer *A* requests a service from the provider *B*. Both *A* and *B* establishes code security policies using WSbSC-RM. Suppose that the service requires object O1 as input, being M_1 and M_2 the object methods and S its state. The initial problem is that *B* does not allow untrusted code to execute. To address this problem using WSbSC *A* gets PSC-cert(M_1) and PSC-cert(M_2), proving to *B* that code from *A* is secure.

3.1. Securing object's state. WSbSS-RM

A and *B* could need the same level of security for state than for code. Apparently, this is similar to case (2.a) where data is secured by *A* and *B* using well-known security mechanisms. However we can get a higher level of security for the state.

Similarly to code in section 2, we can distinguish actors related to object's state (data). We can consider that these actors virtually offer services about state: *owner*, the owner of the state, *A* in Fig. 3 (note that state can represent relevant data that an entity owns); *supplier*, optionally the owner can delegate to this actor the action of providing the state to another entity; *client*, uses the state provided by a supplier; *verifier*, verifies the data consistency (e.g., the verifier tests the data structure and its integrity); *converter*, optionally it may be necessary to convert the internal state from one format into another without changing the state itself (it is similar to code compilation); and *manager*, possesses an environment to manage the state (it's analogous to code execution). There is a code/data actors duality (author-owner; compiler-converter; processor-manager). In general, these actors can be local o remote. Using the analogy of WSbSC-RM for code, we call the dual reference model for the state WSbSS-RM (Web Services based Secure State – Reference Model). WSbSS-RM allows providing a further level of security on data with respect to scenarios such as the depicted in Fig 2.a. Consequently, the service consumer gets basic metadata related to the object state. Continuing the analogy with code, we will call this metadata PSS-cert (Portable and Secure State). This certificate will be generated by the data actors.

3.2 Securing objects' state and methods: WSbSO-RM

At this point, the service consumer has metadata about the whole object (state and methods) that can

be offered to another entity to demonstrate that all the methods have been created (author), provided (supplier), generated (compiler) and validated (verifier) by trusted actors that sign their actions with a certificate. Moreover, the state of the object has been identified (owner) and, eventually, it has been converted (converter) and validated (verifier) by trusted actors also. We call the certificate that aggregates both state and methods metadata PSO-cert. In our example, PSO-cert(O_1) = PSC-cert(M_1) + PSC-cert (M_2) + PSS-cert (S). Returning to Fig. 3, *A* sends O_1 with PSO-cert(O_1) to *B*. Following the notation, we name Web Services based Secure Object Reference Model WSbSO-RM. We assume that *A* and *B* trust actors described above. *A* and *B* may even share some of those actors.

Now, *B* has received the M_1 and M_2 code. Perhaps this code is not suitable for its execution in *B*'s processor, so *B* has to transform it (e.g., compile to generate code compatible with its execution environment). When *B* returns *A* the service result, it will have to certify that the recompiled versions of M_1 and M_2 (M_1' and M_2'), were securely executed in *B* as WSbSC mandates, i.e., M_1' and M_2' has been compiled, verified and executed by trusted actors. This is necessary because *A* should not allow the execution of O_1 methods, and consequently its state access/modification, in an untrusted execution environment. This fact will be returned to *A* in the form of M_1' and M_2' PSC-certs. Note that it isn't necessary to generate metadata from all WSbSC-RM actors. For example, if examining original M_1 and M_2 PSC-certs, *B* realizes that its compiler environment is compatible with M_1 and M_2, *B* does not need to compile them again and it can use compiled code offered by *A*.

B performs the service using M_1' and M_2' to access and/or modify O_1 state. We want to borrow some ideas from SOA and stress the fact that they are also observed in WSbSC and WSbSS. Surely, when the service has finished, internal *B* state has been modified but as SOA-RM says "internal actions that service providers and consumers perform as a result of participation in service interactions are, by definition, private and fundamentally unknowable". In fact, we can consider that *A* doesn't know either *B*'s internal actions to realize the service neither its initial nor final state; and *B* only handles O_1 through its methods, without needing to know the method details. Although *A* and *B* don't know neither the process nor the state details, they have shared "facts" and processes from a "real world" point of view. Global process is distributed. SOA-RM focuses on "the sets of facts shared by the parties". WSbSC-RM adds the focus on "shared process" too. SOA-RA is adding now this missing concept including a similar term (joint action) not present in SOA-RM.

A gets the service result including O_1 that may have a new state S' due to the interaction with *B* by means of M_1' and M_2'. *A* also receives the service implementation PSC-cert, as in Fig.2 case 2.b; and the PSO-cert(O_1'), that certifies that the new version of the object methods is securely executed, verified, and eventually recompiled in a trusted environment; and that the objects state may have been converted and verified by trusted parties also. Note that, despite being the same object O_1, we denote it O_1' to stress the fact that, eventually, the state may have changed by new versions of the methods.

A can check integrity of M_1' and M_2', S' and the service implementation as was described previously. Note that for performance reasons, both consumer and provider haven't to get PSO-certs every time. There is a trade-off between performance and security. Consumer and provider policies may state in which cases a PSO-cert update is required.

3.3. Securing legal access to state

It must be observed that, for the moment, the security level that the PSO-cert(O_1') built by *B* is not enough. The service consumer *A* has not the sufficient guaranty to be sure that the new state S' has been obtained my means of M_1 and M_2 (or their transformed versions M_1' and M_2'). In general, it must be guaranteed that the object state can only be managed by means of its methods, wherever the object is used. To reach such a guaranty, a new verifier actor is needed. While central problems about data and code verification have been addressed from different points of view, the problem of securing legal access to the state through the object methods is a more complex task, being the mobile agents scope the research field where the problem has been mainly addressed (Claessens, Preneel and Vandewalle, 2003). The verifier also adds metadata to the PSO-cert(O_1) and sign its information block in such a way that the consumer can also verify its identity and the verification itself.

We are planning to address this problem from a WSbSO-RM perspective using secret sharing techniques. (Shamir, 1979). Another approach (Miao and Wei, 2003) may be that the verifier reproduces the state modification following the same sequence of operations that *B* has performed. Therefore, *B*'s processor must provide the verifier with this sequence of actions, e.g. using a log file.

4. IMPLEMENTING THE MODEL WITH WEB SERVICES.

In previous sections, a conceptual framework has been described based on concepts from SOA and WSA. We can outline how we can address the implementation of this model for each specific scenario using Web services standards. The central point in this approach is that both service consumer and provider have to define their respective security policies related to WSbS*-RM. Expression of these policies can be based on WS-Policy. On the other hand, PSO, PSC and PSS can be transmitted in a secure manner using WS-Security standard. In order to define PS*-certs, XML-Dsig, XML-Encryption, SAML and XML-Schema enable security and mechanisms to ensure all security issues described. Finally UDDI, could be used to get a reference for the identity of each WSbS* actors.

5. CONCLUSIONS

The main contribution of our paper is the definition of a conceptual framework for the assurance of integral code and state security in SOA. It is a framework where not only the message security is considered, but also the security of the code that processes it. Besides, we propose an extra level of security in a service interaction considering both code and state. Finally, an incremental model of security based on certificates issued by each model actor provides a means for ensure security and achieve a trusted environment.

Our main lines of research are: (1) to work on the implementation of the model in several real world scenarios; (2) to improve security between state and methods using secret sharing techniques (as commented in section 3.3); (3) we have realized that both code and state share similar actors. This has suggested us a new line of research for the definition of a meta-model that would describe uniformly the structure and the behaviour of the state and the code security models.

ACKNOWLEDGEMENTS

Partially supported by project FOMENTA 2008/01 of the Comunidad Autónoma de La Rioja.

REFERENCES

Bhargavan, K., Fournet, C., Gordon, A.D., 2004. Verifying policy-based security for web services. In Proceedings of the 11th ACM conference on Computer and communications security, October 2004

Birman, K.P., 2004. Like it or not, web services are distributed objects. In Communications of the ACM, december 2004

Chang, B-Y. E., Chlipala, A., Necula, G.C., Schneck, R.R:, 2005. The open verifier framework for foundational verifiers. In Proceedings of the 2005 ACM SIGPLAN international workshop on Types in languages design and implementation, January 2005

Claessens, J., Preneel, B., Vandewalle, J., 2003. (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

European Project, 2006. Security of Software and Services for Mobile Systems,http://www.s3ms.org, March 2006.

Franz, M., Chandra, D., Gal, A., Haldar, V., Reig, F., Wang, N., 2003. A portable Virtual Machine target for Proof-Carrying Code. In Proceedings of the 2003 workshop on Interpreters, virtual machines and emulators, June 2003

Foster, I., Parastatidis, S.,Watson, P., Mckeown, M., 2008. How do I model state?: Let me count the ways. In Communications of the ACM, september 2008.

Gutiérrez, C., Fernández Medina, E. and Piattini, M., 2005. Web Services Enterprise Security Architecture: A Case Study. SWS'05, november 11, 2005

Lange, D.B., Oshima, M., 1999. Seven good reasons for mobile agents", Communications of the ACM, v.42 n.3, p.88-89, March 1999

Miao, C.,Wei, R., 2003. Secret Sharing for Mobile Agent Cryptography. In Communication Networks and Services Research Conference, Session B

Rodríguez Priego,E., García Izquierdo,F.J., 2007. Securing Code in Services Oriented Architecture, ICWE07. LNCS 4607, pp. 450-555. Springer-Verlag 2007.

Rubin, A.D., Geer Jr., D.E., 1998. Mobile Code Security, IEEE Internet Computing, vol. 02, no. 6, pp. 30-34, Nov/Dec, 1998

Sekar, R., Ramakrishnan, C. R., Ramakrishnan, I. V., Smolka, S. A., 2001. Model-Carrying Code (MCC): a new paradigm for mobile-code security. In Proceedings of the 2001 workshop on New security paradigms, September 2001

Seshadri, A., Luk, M., Perrig, A., van Doorn, L., Khosla, P., 2006. Externally verifiable code execution. In Communications of the ACM, september 2006.

Shamir, A., 1979. How to share a secret. Commun. ACM 22, 11 (Nov. 1979), 612-613

Whitman, M.E., 2003. Enemy At The Gate: Threats To Information Security. In Communications of the ACM, August 2003

OASIS, 2006. Reference Model for SOA v1.0

OASIS, 2008. Reference Architecture for SOA v1.0

Vogels, W., 2003. Web services are not distributed objects. In Internet Computing, Dec. 2003

W3C, 2004. Web Services Architecture