Secure SOAP Requests in Enterprise SOA

Maarten Rits and Mohammad Ashiqur Rahaman
SAP Research
805, Avenue du Docteur Maurice Donat
Font de l'Orme, 06250 Mougins
+33 (0)4 92 28 62 00
{maarten.rits}@sap.com

Introduction

Web service specifications (WS*) have been designed with the aim of being composable to provide a rich set of tools for secure, reliable, and/or transacted web services. Due to the flexibility of SOAP-level security [1] mechanisms, web services may be vulnerable to a distinct class of attacks based on the malicious interception, manipulation, and transmission of SOAP messages, which are referred to as XML rewriting attacks [2]. Although WS-Security, WS-Policy and other related standards theoretically can prevent XML rewriting attacks, in practice, incorrect use of these standards may make web services vulnerable to XML rewriting attacks. All WS* security related specifications, however, introduce new headers in SOAP messages. So concerns about the operational performance of Web services security are legitimate because added XML security elements not only make use of more network bandwidth but also demand additional CPU cycles at both the sender side and at the receiver side. Therefore it is desirable to examine the performance issue of Web services security. The main achievements of this work are that we explore XML rewriting attacks [2] against web services. We propose measures detecting these attacks built on the idea of adding additional SOAP structure information. We further evaluate the performance of the proposed solution against the existing state of the art. We discuss how this work is related to Enterprise SOA, SAP's implementation of Service Oriented Architectures.

Related Work

Security protocols, described using web service specifications (WS*), are getting more complex day by day. Researchers are applying formal methods to verify and secure the protocols' and specifications' goals. As new vulnerabilities are exposed, these specifications continue to evolve. Microsoft's SAMOA [3] project is among the pioneer efforts where web services specifications are analyzed using rigorous formal techniques. One of the focus areas of the SAMOA project is to identify common security vulnerabilities during security reviews of web services with policy-driven security [2]. The authors describe the design of an advisor for web services security configurations, the first tool both to identify such vulnerabilities automatically and to offer remedial advice. While their previous work [4] is successful to generate and analyze web services security policies to be aware for vulnerabilities to XML rewriting attacks, this tool is able to bridge the gap between formal analyses and implementation quite efficiently.

Some guidelines about performance evaluation of web services are specified in [8]. They address the performance overheads of web services: the XML size, the calculation of the message size, the choice of the XML parser, the costs of the serialization and deserialization, the costs of connection establishment and the overheads at the network level. The performance of web service security in terms of its relationship in both the implemented cryptography in Java and the properties of input documents like the size and the complexity of structure are investigated in [9]. Different cryptography algorithms and various kinds of key references in XML are compared as well. They also categorize the web service security activities consisting of at least cryptography operations and XML processing. Cryptography operations are byte based, structure neutral and computation intensive. They conclude that XML processing, in particular XML canonicalization is the area where WS security performance should focus on. In our later evaluation we take such performance issues into account as well.

Objectives

Considering the specifications of WS-Security, WS-Policy, WS-SecurityPolicy, we observe that a large number of SOAP extensions is possible. SOAP header (<Security>) information never considers the SOAP message structure which is essentially the major attack point. In the context of our work we recognize the dynamic structure of SOAP messages by referring to them using the term SOAP Account. We show that including SOAP Account information in SOAP we can detect these XML rewriting attacks early in the validation process. This SOAP structure (SOAP Account) information can be integrated easily while adding

the headers. We need an inline approach to incorporate SOAP Account information into the message. The objectives of the proposed technique are:

1. To be able to pass SOAP Account information about the exchanged SOAP messages in a domain independent fashion.
2. SOAP messages should be protected while they are processed by a node such that they can be updated legitimately by the intermediaries if they are required to do so. Any tampering with pre-existing message parts by the compromising node must be detected early by the recipient before committing its resources to validate and to process the request.

Proposed Solution

At the time of sending a SOAP message, we can always keep account of the elements, by including the SOAP Account into the message:

· Number of child elements of root (Envelope).
· Number of header elements.
· Number of references for signing element.
· Predecessor, Successor, and sibling relationship of the signed object.

These SOAP Account information are computed while we are creating the message itself in the sending application. We do not incur any considerable overheads for the computation. Figure 1 shows an example of the approach in the attempt of an attack.

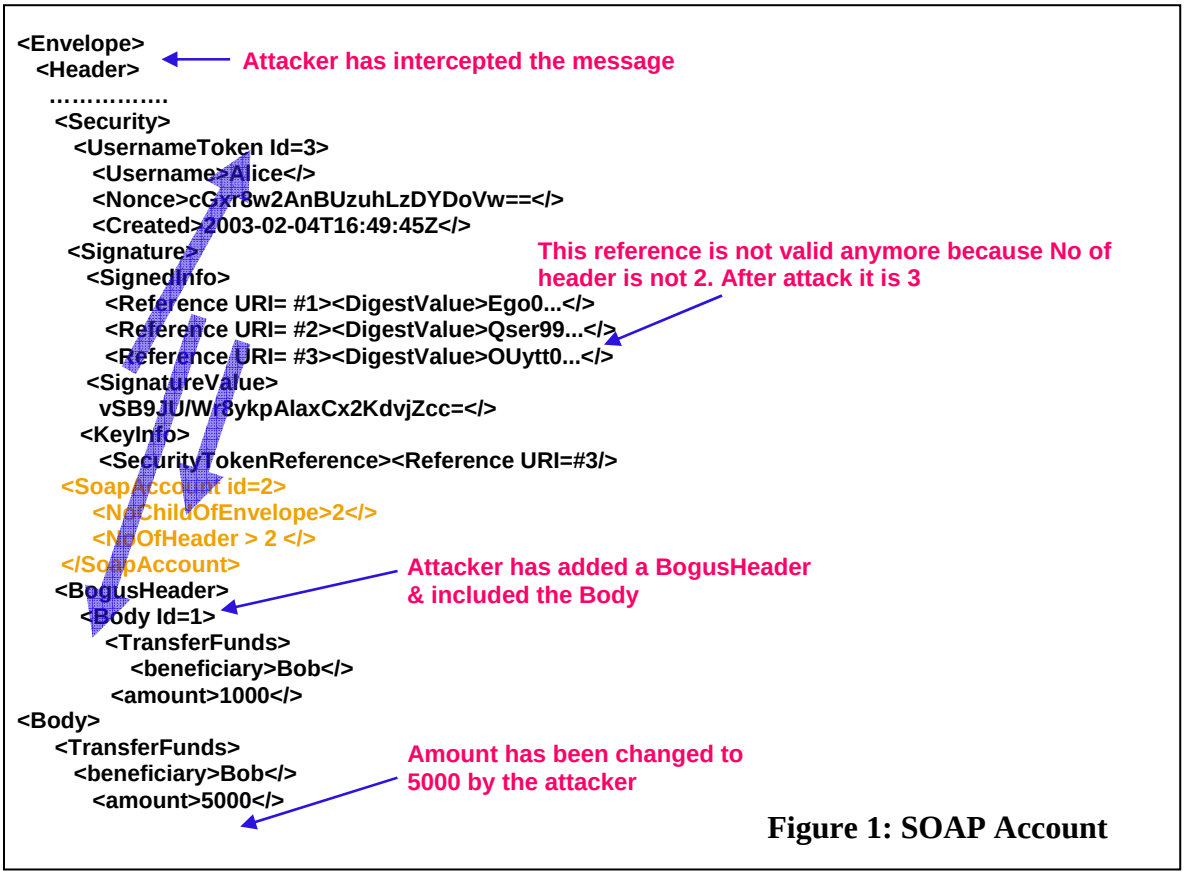

Figure 1: SOAP Account

Figure 2 shows the performance evaluation of our solution compared to the work presented in [3]. On average we get 5 times better performance. Another key advantage of our approach compared to the policy driven approach, is that the deletion of headers and elements can be detected without restricting the XML format. Deletion is a stronger form of a rewriting attack. In order to prevent this with the policy approach, every header and element should be declared as mandatory, which introduces first of all a performance penalty in the validation phase and more important it reduces strongly the flexibility of the XML message format. Only message elements that were defined in advance by the partners can be added. With our

approach the different intermediaries still have some flexibility to add additional information, which can be detected for deletion/ rewriting by the subsequent parties.

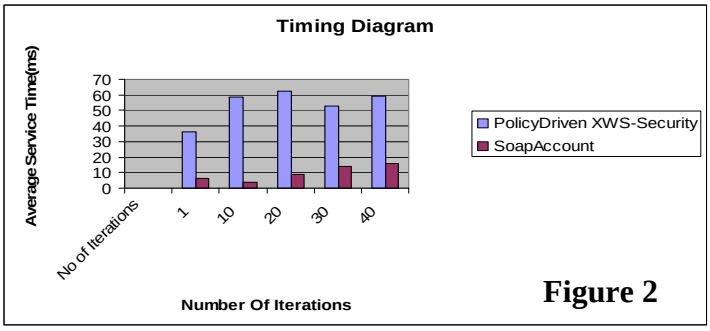

Figure 2

Future Work

We have considered the SOAP structure information to be used in the context of securing single messages. Using WS-Security independently for each message to secure the integrity of a session of messages is rather inefficient. WS-SecureConversation introduces security contexts, which can be used to secure sessions between two parties. How SOAP structure information can be used for securing a session is a future research topic. We have used also only the XWS-Security Framework as a comparable message level security implementation and have drawn some comparisons of WSE [3] implementation with our technique. It would be interesting to see how the performance scales regarding other implementation frameworks of message level security.

References

[1]http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
[2]K. Bhargavan, C. Fournet, A. Gordon, and G. O©Shea An Advisor for Web Services Security Policies, http:// research. microsoft.com/~adg/Publications/details.htm#sws05
[3]Microsoft Research; http://research.microsoft.com/projects/Samoa/
[4]K. Bhargavan, C. Fournet, and A. D. Gordon. Verifying policy-based security for web services. In 11th ACM Conference on Computer and Communications Security (CCS.04), pages 268.277, October 2004.
[5]T. Nadalin, ed., Web Services Security Policy Language (WS-SecurityPolicy),Version 1.0, 18 December 2002, http://www.verisign.com/wss/WSSecurityPolicy.pdf
[6]K. Bhargavan, C. Fournet, A. D. Gordon, and R. Pucella. TulaFale: A security tool for web services. In International Symposium on Formal Methods for Components and Objects (FMCO.03), LNCS. Springer, 2004
[7]B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In Proceedings of the 14th IEEE Computer Security Foundations Workshop, pages 82.96. IEEE Computer Society Press, 2001.
[8]Ana C.C. Machado and Carlos A. G. Ferraz. Guidelines for Performance Evaluation of Web Services, WebMedia .05, December 5-7,2005
[9]Hongbin Liu, Shrideep Pallickara Geoffrey Fox, Performance ofWeb Services Security.

Related Publications by same Authors:
[10] Mohammad Ashiqur Rahaman, Maarten Rits and Andreas Schaad. An Inline Approach for Secure SOAP Requests and Early Validation, OWASP Europe Conference, Leuven, May 30, 2006.
[11] Mohammad Ashiqur Rahaman, Andreas Schaad and Maarten Rits. Towards Secure SOAP Message Exchange in a SOA. ACM Workshop on Secure Web Services (SWS), George Mason University, Fairfax VA, USA, November 3, 2006.