XPRIDE: Policy-Driven Web Services
Security Based on XML Content

Zein Radwan        Camille Gaspard        Ayman Kayssi        Ali Chehab
Department of Electrical and Computer Engineering
American University of Beirut
Beirut 1107 2020, Lebanon

Abstract - In this paper we present XPRIDE as an efficient security architecture for assuring the confidentiality and integrity of the XML-based SOAP messages in Web Services. The policy-based approach employed in XPRIDE can be easily configured and modified to provide security according to the content and sensitivity of the data. Implementation shows that XPRIDE has considerable performance gains over existing bulk encryption protocols such as SSL and over existing policy-based solutions such as WS-Security. XPRIDE is designed as a platform-independent architecture and can be seamlessly integrated into existing application servers.

I. INTRODUCTION

Web Services are becoming the most common way for enterprise applications to interoperate. This importance comes from the fact that Web Services are platform- and language-independent and easily discovered by clients. This has pushed the move for companies to implement a Service-Oriented Architecture (SOA) in their enterprise networks. Although this application-to-application structure significantly improves the interaction and development of the processes in the business world, it raises many security concerns, since it greatly increases the exposure of the critical and sensitive data that is being exchanged.

Web Services exchange information with their clients using the Simple Object Access Protocol (SOAP) messaging protocol [1]. SOAP uses XML-based documents to wrap the transmitted data. SOAP messages conform to the standardized SOAP syntax, where a SOAP message is an envelope that contains multiple headers and a body. The message header contains additional information about the message, such as security information and timestamps, while the message body contains the description and the information needed to invoke the target Web Service in case the message is a request, or it may contain the result of a called Web Service in case the message is a response. However, SOAP messages are not necessarily directly delivered to their destination, but they may pass through a number of intermediaries which act as a receiver and a sender that forward the message again after performing additional processing on it, usually involving the message headers.

Many methods are used to secure and encrypt the critical data exchanged among Web Services. The encryption can be performed at the application level by modifying the code of the application itself to encrypt the content of the SOAP message. Encryption may also be achieved by relying on the transport layer security technologies, such as Secure Sockets Layer (SSL) [2] or Transport Layer Security (TLS) [3], where these technologies present a point-to-point security solution that provides confidentiality and integrity for the data exchanged. Security may also be performed at the message level by using XML security techniques such as XML Encryption [4] and XML Digital Signature [5].

The Organization for the Advancement of Structured Information Standards (OASIS) presented the Web Services Security Specification (WS-Security) [6] as a way for Web Services to use several different security models via SOAP extensions. This specification describes enhancements to SOAP messaging to provide message integrity and confidentiality by relying on existing specifications such as XML Encryption and XML Digital Signature. It also provides a general-purpose mechanism for associating security tokens with message content.

To avoid the unnecessary effort and delay of bulk encryption, PRIDE security [7] was presented as an efficient content-based security solution [11] for protecting the privacy and integrity of web traffic exchanged between enterprise application servers on the Internet and mobile wireless devices. PRIDE is a policy-driven security architecture that employs content-based encryption and hashing methodologies to secure network data based on sensitivity and relevance. In this paper we present XPRIDE, an extension to PRIDE, as a SOAP security filter for securing the SOAP messages exchanged over the network between Web Services and their clients. XPRIDE is a policy-driven security architecture that employs content-based encryption to secure the SOAP messages based on their XML content. This system acts as a filter that performs security operations in a transparent manner to the sender and receiver, and hides the processing details and therefore avoids the application level security complexity. Moreover, XPRIDE performs content-based security depending on security policies that can be defined by administrators to specify exactly the security required for each part of the data. This approach avoids the unnecessary bulk encryption carried out by transport layer security. The advantage that this solution has over the existing content-based security provided by WS-Security, is that it supports the ability to secure specific patterns occurrences in the SOAP messages, while WS-Security, which relies on XML encryption, enables whole XML element encryption, although in many cases only parts of the SOAP response need to be encrypted. XPRIDE applies security dynamically on the data without the need to specify - at design time - these parts that need to be encrypted as in WS-Security. Furthermore, the security policy proposed in XPRIDE enables the use of
content-based security on the attachments appended to SOAP messages.

The rest of the paper is organized as follows: in Section II, we review some previous work related to Web Services security. In Section III, we describe the main design features of XPRIDE and its architectural components. Section IV presents XPRIDE’s implementation while in Section V we present the testing and results obtained. Section VI analyses the performance of XPRIDE as compared to WS-Security, and in Section VII we provide some conclusions.

II. PREVIOUS WORK

A common way for achieving Web Services security is by relying on a secure transport layer, typically using SSL, which encrypts all data transmitted across a network and can thus prohibit eavesdropping and unauthorized access to Web Services and SOAP messages. This technology ensures point-to-point security by establishing a secure channel on top of TCP, and provides data integrity and confidentiality in addition to authentication. Point-to-point security techniques are appropriate for creating secure connections in which information can be directly transmitted; an example is using SSL to secure online banking transactions where account numbers and client information are transmitted. However, SOAP messages sent over the network may need to travel through numerous intermediaries before reaching their ultimate SOAP receiver. Thus, in a Web Services architecture, intermediaries can manipulate a message on its way to the receiver. But when using a transport layer security technology such as SSL, intermediaries will no longer be able to control the SOAP messages [8], since the message would need to be decrypted by the intermediary before being forwarded to the ultimate receiver using a new encrypted stream. In addition, SSL performs bulk encryption on the whole data - in this case the SOAP messages- as one stream, without separating sensitive data from the insignificant data that causes no threat whatsoever if exposed. Although most of the data in some cases need not be secured [12], SSL encrypts everything, thus causing additional delay and inefficiency in the system performance especially if significant amounts of data are transmitted. The encryption can also be performed at the application level by modifying the code of the application itself to encrypt the content of the SOAP message; this ensures security on a per-message basis. However, integrating security functions into applications requires a significant amount of coding effort and expertise. XML Encryption and XML Digital Signature are also common ways to secure SOAP messages and provide security at the SOAP message level. XML Encryption provides data confidentiality and supports partial encryption of XML files, by encrypting only specific XML elements using XPath expressions to define these elements. XML Digital Signature is used in a similar way to provide integrity, authentication and non-repudiation to the signed data. To combine various security techniques and concepts together, OASIS has standardized the WS-Security Specification that was proposed by Microsoft, IBM and VeriSign. The task was not to invent a new security mechanism, but rather to define ways to use the techniques that already existed in the Web Services world. WS-Security proposes a standard set of SOAP extensions that can be used to insure message content integrity and confidentiality, by relying on existing standards and specifications such as XML Encryption and XML Signature among other techniques. What WS-Security adds to existing specifications is a framework to embed these mechanisms into a SOAP message through additional headers. WS-Security lets applications attach security data to the headers of SOAP messages in what is referred to as security tokens. No specific type of security token is required; the specification is designed to be extensible. It was proposed by OASIS as well to add new standards to be layered on top of WS-Security. One of these standards is WS-SecurityPolicy [9], which specifies how to define security assertions that clearly state a Web Service's preferences and security requirements, such as the digital signature and encryption algorithms used, and the parts of the message (XML elements) that need to be secured. WS-SecurityPolicy builds on the foundation of WS-Policy that defines a general approach to specifying policies of all kinds for Web Services.

III. DESIGN AND ARCHITECTURE

This section gives an overview of the design and architecture of XPRIDE. An abstract view of the major
A. The Security Policy and Scope

The Security Policy controls the overall security behavior of XPRIDE. The policy configuration is stored in an XML-formatted document for each Web Service secured by XPRIDE, and is used by the XPRIDE security engine on the sender side. A reference table that indicates the locations of the encrypted parts is used by the XPRIDE security engine on the receiver side for decrypting and validating the received data. The Policy is divided into two main parts (see Figure 2): the first part (classes) specifies security-related attributes and parameters, while the second deals with how security mechanisms are applied to the SOAP message. The security-related attributes specified in the first part of the policy control the behavior of the filter regarding how confidentiality, integrity and key management are performed. Following is a list of these attributes: 1) encryption algorithm, 2) hashing algorithm, 3) key management algorithm, and 4) session-keys life time. The second part of the Policy is divided into three sections. The first section specifies the security classes that are to be used by the Policy. Each class holds a set of security specifications that can be applied later to different parts of data only by referring to the name of the specific class that holds them without the need to repeat the specifications again. These security classes are characterized by the following attributes: 1) class name, 2) security level, and 3) integrity enforcement. XPRIDE supports three security levels: a High Security level equivalent to an AES [10] 256-bit key length; a Medium Security level which is equivalent to an AES 192-bit key length; and a Low Security level which is equivalent to an AES 128-bit key length. The three keys are derived from one key that is initially generated during the key exchange process. The integrity enforcement attribute in the Policy specifies if data integrity (using digital signatures) needs to be applied. The second and third sections of the policy deal with the SOAP message content. These sections specify the security level and scope for each part of the message, by assigning a certain security class for the part of the message that needs to be secured. The body of a SOAP message leaving a Web Service can be carrying either a set of parameters in case the message is a SOAP request or the response content in case the message is a SOAP response. In order to cover these two cases, and knowing that a policy is assigned to one Web Service, the policy in its second section, contains the information needed to secure the responses leaving the Web Service, while the third section contains the information needed to secure the requests made by the Web Service. Since a Web Service offers a number of methods that can be invoked, the second section (Responses) may contain an entry for each method that is distinguished by name. While the third section (Requests) includes entries for the methods that belong to other Web Services and that may be requested and invoked by this Web Service and whose parameters need to be secured, in this case the request is distinguished by the requested method name and the name of the Web Service it belongs to. Therefore, in case an XPRIDE filter is installed at the client side to secure the outgoing SOAP messages and this client is not a Web Service itself, the policy file at the corresponding XPRIDE filter does not need to include any entries in the Responses section since responses are never sent by this client. The data exchanged between Web Services can be either simple data enclosed in the SOAP message or attached files. Accordingly, data can be identified using the following three mechanisms: 1) filename, 2) regular expression, or 3) filename and regular expression. Filename is used to specify the attached file that needs to be secured; it can be any valid file name along with the extension, an asterisk (*) followed by an extension that specifies all files corresponding to the type identified by this extension, or an asterisk (*) for all files. Regular expressions (REs) are used to match and secure sensitive patterns that may appear in the SOAP message. REs allow for securing specific pattern occurrences in the SOAP messages, and that are embedded inside the XML elements, while WS-Security, which relies on XML Encryption, enables whole XML element encryption only. The filter, when finding a match, encrypts the entire fraction where the match occurred. “StartRegExp” is used to match a pattern occurrence, and is used with “EndRegExp” to match and secure all data between these two regular expressions. When an entry is declared in the policy, but no regular expression is declared in it, it means that the whole entry is to be secured and no content-based matches are applied. File name and regular expressions are used together to find an occurrence of a specific pattern in a file that is being attached, an example is securing a specific match in an HTML file. This way, the security policy enables content-based security application on the attachments appended to SOAP messages as well. All the above identification mechanisms are used to specify the encryption boundaries and security levels that are to be applied when enforcing the policy.
B. Policy Caching

Policy resolving and caching are important features that are performed in XPRIDE and that are handled by the Policy Loader. Caching improves the policy loading time by using a compacted binary copy of the security policy instead of parsing it every time it is requested. However, due to the possible modification of the policy configuration by administrators, a hash is used to make sure that the policy information in the cache is always up-to-date.

C. SOAP Message Confidentiality and Integrity

The Security Engine is the component responsible for providing data confidentiality and integrity based on the sensitivity of the data. To support a flexible encryption scheme, XPRIDE identifies the content to be secured in the security policy and provides a filter through which the Web Service and its client can communicate and interact securely. Application developers need not to be concerned about embedding security functions into the application itself and are only required to supply the Security Engine with identification of the data to be secured, how this data is to be secured and to what degree. The Security Engine filter on the recipient side intercepts the received secure messages, performs the necessary integrity verification and decryption operations based on the reference table sent along with the secured message, reconstructs the original message and delivers it to the recipient.

The following steps summarize the overall request/response model of XPRIDE both on the client side and on the service side.

At the service side, XPRIDE (see Figure 3):
1) receives an incoming request from a client, 2) decrypts and validates the request if secured depending on the attached reference table, 3) forwards the request after reconstructing it to the appropriate Web Service, 4) gets the generated response from the Web Service, 5) parses the policy store to check if a policy exists for the corresponding Web Service, and resolves the security settings, 6) applies confidentiality and integrity settings on the generated response, 7) attaches to the response an encrypted version of the reference table that indicates what parts of the message were encrypted at what locations, and 8) forwards the response to the client side.

On the other hand, XPRIDE on the client side:
1) gets a request from the client, 2) parses the policy store to check if the request made should be secured, and resolves the security settings, 3) applies confidentiality and integrity settings on the request parameters as indicated by the policy, 4) attaches to the request an encrypted version of the reference table, 5) sends the request to the appropriate Web Service, 6) waits for the response to arrive, 7) decrypts and validates integrity of the incoming response depending on the attached reference table, and 8) delivers the plain response to the client.

IV. IMPLEMENTATION

XPRIDE was implemented using the Microsoft.Net 2.0 Framework with Visual Studio 2005, the WSE (Web Services Enhancement) class library 1.0 and Internet Information Services (IIS) 5.1. The tests were performed on a PC with the following specifications: DELL Inspiron 510 with 1.6 GHz Centrino CPU, 512 MB RAM and 40 GB hard drive. The operating system used was Windows XP Professional SP2. The implementation relies on using the SOAP filters provided by the WSE class library to access the raw SOAP messages exchanged over the network between the Web Services and their clients. SOAP output filters intercept the SOAP messages flowing out, while the input filters intercept the incoming

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2007 proceedings.
SOAP messages. Therefore, SOAP filters seemed as the ideal candidate to implement custom security for Web Services. The main web method that was considered for testing takes as an argument the name of a file, fetches the file from the server storage and sends it back to the client which in turn saves it locally.

V. COMPARISON WITH BULK ENCRYPTION

We tested the prototype XPRIDE implementation on sample file data of size 11 MB. In order to compare the performance of content-based security to the performance of bulk encryption (such as SSL or TLS), we implemented separate testing filters, on both the client and the server side. These filters simply encrypt and decrypt the content of every SOAP message to simulate a bulk encryption system.

The first test was performed assuming no encryption at all; the time in this case is considered to be a reference time, which we refer to as $t_0$. The same test was repeated with bulk encryption. The additional time in this case is the difference value $t_1$, due to the bulk encryption operations, and calculated as follows:

$$ t_1 = \text{time taken with bulk encryption} - t_0 $$

XPRIDE content-based encryption was tested next, with policies that gradually increased the percentage of the data that needed to be secured. The percentage was varied as follows: 20%, 30%, 50%, and up to 70% of the data. For each percentage value, the following cases were considered:

1. One regular expression is used in the policy. The data to be encrypted is enclosed between two distinctive words in the expression, which denote the start and end of the sensitive data.
2. To study the effect of increasing the number of regular expressions in a policy on the performance of XPRIDE, the number of regular expressions used in the policy was increased while keeping the overall percentage of the encrypted data constant. The number of regular expressions was increased to 3, 4, and 5.

For every test, a time value $t_2$ was calculated as follows:

$$ t_2 = \text{time taken with content-based encryption} - t_0 $$

Tests were repeated several times and the average values were recorded.

The values of the averaged times ($t_2$) using XPRIDE content-based encryption filters, for the different cases listed above, are summarized in Table 1. Figure 4 (A) compares the bulk encryption time with the time of content-based security for different data percentages, when using a single regular expression. Similarly, Figures 4 (B), (C), and (D) compare the bulk encryption time with the time taken after applying content-based security when three, four or five regular expressions, respectively, are used in the security policy.

By examining the results shown in Table 1 and Figure 4, we can conclude that content-based security, as performed by XPRIDE, remains efficient as long as the percentage of the data that needs to be encrypted in relation to the whole data does not exceed 50%. Since, in most cases less than 20% of the web data traffic is classified as sensitive and needs to be encrypted [7], XPRIDE is very efficient in such cases, and can reduce the encryption overhead time by a factor of 2.5.

VI. COMPARISON WITH WS-SECURITY

Since XPRIDE and WS-Security are defining the parts of data that should be encrypted, they are both exposed to user mistakes when defining these parts, compared to bulk encryption. To compare the performance of XPRIDE with that of WS-Security, we propose the following business scenario. A large company headquarters connects to each of its branches using Web Services to synchronize 11 MB of data and keeping it up-to-date. The Web Services that the business uses are described in Table 2.

Table 1. XPRIDE Time (in Seconds) for Different Encrypted Data Percentage and Varying Number of Regular Expressions

<table>
<thead>
<tr>
<th>Percentage Encrypted</th>
<th>1 Regular Expression</th>
<th>3 Regular Expressions</th>
<th>4 Regular Expressions</th>
<th>5 Regular Expressions</th>
<th>No Encryption</th>
<th>Bulk Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>30%</td>
<td>8.464</td>
<td>9.926</td>
<td>9.512</td>
<td>10.574</td>
<td></td>
<td></td>
</tr>
<tr>
<td>70%</td>
<td>15.468</td>
<td>15.636</td>
<td>19.550</td>
<td>23.029</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4. XPRIDE Results
Table 2. Web Services Description

<table>
<thead>
<tr>
<th>Web Service</th>
<th>Description</th>
<th>% of total traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Service A</td>
<td>Transmits documents such as purchase orders, invoices and reports</td>
<td>60%</td>
</tr>
<tr>
<td>Web Service B</td>
<td>Continuously transmits the currency exchange rates</td>
<td>15%</td>
</tr>
<tr>
<td>Web Service C</td>
<td>Performs human resources related transactions</td>
<td>25%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service</th>
<th>% encrypted WS-Security</th>
<th>% encrypted XPRIDE</th>
<th>% of total traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service A</td>
<td>100%</td>
<td>25%</td>
<td>60%</td>
</tr>
<tr>
<td>Service B</td>
<td>0%</td>
<td>0%</td>
<td>15%</td>
</tr>
<tr>
<td>Service C</td>
<td>35%</td>
<td>30%</td>
<td>25%</td>
</tr>
<tr>
<td>Total% Encrypted</td>
<td>69%</td>
<td>22.5%</td>
<td>25%</td>
</tr>
<tr>
<td>Time Taken</td>
<td>13 sec</td>
<td>4.5 sec</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. XPRIDE and WS-Security Comparison

The Web Services should transmit data securely since they may contain sensitive information. The first scenario is to use WS-Security and WS-SecurityPolicy to secure the traffic of these Web Services. In this scenario, the policy defines the XML elements that should be encrypted statically. Thus, by using the WS-SecurityPolicy, and because WS-Security pays no attention to content, all the traffic sent by Web Service A should be encrypted even though it may contain non-sensitive documents that need no encryption such as images and videos. Also, no part of the traffic sent by Web Service B is to be encrypted because it includes international exchange rates that are public data. Finally, a fraction of 35% of the traffic sent by Web Service C is secured: this percentage represents the XML elements that are defined as sensitive in the policy. As a result the percentage of the encrypted data in the WS-Security scenario will be around 69% (see Table 3.)

XPRIDE, on the other hand, allows the encryption to be dynamically applied depending on the sensitivity of the content, using regular expressions. We assume therefore that the sensitive documents comprise 25% of the overall documents sent by Web Service A, and 30% of the traffic sent by Web Service C. This reduces the fraction of encrypted data in XPRIDE to 22.5% of the overall traffic (see Table 3.)

From Figure 4 (A), which represents the case of one regular expression, we can see that the time taken by XPRIDE (at 22.5%) for 11 MB is 4.5 seconds. The time taken by WS-Security corresponds to a 69% percentage, and is around 13 seconds. Therefore, XPRIDE takes 65% less time to secure the 11 MB of Web Services traffic described above. Using the same approach, we estimated the percentage of time decrease for three, four, and five regular expressions. The results are shown in Figure 5.

VII. CONCLUSION

In this paper we presented XPRIDE, as a policy-driven security architecture for providing confidentiality and integrity for the XML-based SOAP messages exchanged among Web Services and their clients. The security provided by XPRIDE is based on the content relevance and sensitivity of the data in a manner that adds additional features to the WS-Security specifications. XPRIDE is designed to work in a platform-neutral manner and to be seamlessly integrated into existing application servers and proxies. The implementation clearly showed the higher performance of content- and policy-based security, in comparison to traditional bulk encryption protocols and to WS-Security.

REFERENCES