Global Information Assurance Certification Paper

SOAP, Web Services and Security
Cameron J. Michelis
GSEC Version 1.4 option 1

ABSTRACT

A Web Service provides an XML interface to a network computing resource in a heterogeneous environment such as the Internet. SOAP is the framework for this communication with the major web services implementations. In this paper I will discuss the security aspects of web services, the current risks involved and the state the technology.

This document requires a basic understanding of XML concepts and cryptography. Knowledge of specific Web Service implementations such as the Microsoft .NET or Sun One projects is not required.

1. Introduction

The general principle behind web services is that you have standard methods of finding, describing and using distributed computing resources on an application level. That is, you can develop an application that will connect to the public network, locate a service that it needs, get a description for an interface to the service and then communicate with the service using XML. This can be done while developing your web service or client application using any development tools you need.

Currently the protocols required for this communication are SOAP, WSDL and UDDI. These protocols combined with the use of XML as a data transfer mechanism provide the framework for a truly distributed computing environment.

This collection of protocols can be used to communicate across standard transport mechanisms such as HTTP, SMTP, FTP and even NNTP. At a high level the protocols that enable applications to find and use web services are considered the framework and the individual services are the building blocks for a distributed computing network.

Commercial Web Service projects such as the Microsoft .NET initiative, The Sun One Project and the IBM Web Services toolkit are developing technologies that promise to revolutionize B2B and B2C transactions with Internet Web Services. These vendor technologies are implementations for web services specifications designed by the W3C committee.

2. Web Service Architecture

2.1 UDDI Protocol

The UDDI protocol is used to advertise a web service. It is not specific to any business category, the protocol simply specifies the format for service providers to use to announce its existence to a service registry. This protocol is central to the success of web service and service central computing in general.

UDDI uses the traditional web service standards such as SOAP and XML on top of common, widespread protocols such as HTTP, SMTP and TCP. The use of these open standards and automatic interfaces allows developers to write software that programmatically discovers the required web service.

Companies and formal business networks can define their own private UDDI network if there is a business case to target a specific set of authorized users and exclude others. This may be useful for certain B2B exchanges such as private auction networks.

2.2 WSDL protocol.

The Web Service Description Language (WSDL) Protocol is used to define how a service subscriber will interface with the defined service. In a nutshell, WSDL uses XML and SOAP to describe the web service interface. This description is delivered in the form of an XML document referred to as a WSDL file. There are a few important parts of the WSDL file worth mentioning. This protocol tells applications how to talk to it.

- The Types section. This section defines the data types used in the service. This section is critical so the client can allocate the proper memory structures to establish a SOAP conversation.
- The Messages Section. This section defines the messages used for the SOAP conversation and associates data types with them.
- The PortTypes Section. This section associates internet communication protocols with messages. An example would be to associate a message defined in the messages section with an HTTP-GET command.
- The Bindings Section. This section tells how to encapsulate information associated with a method.
- The Services Section. This section tells how to associate what protocols can be used with which service and it provides URI access information.

2.3 SOAP Messaging

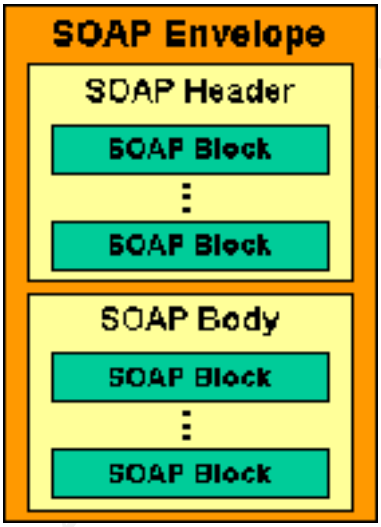
The SOAP message is the standard for communication with a web service, around which SOAP containers are built and interfaced to the application providing the service. A SOAP container will convert the SOAP request into the

language of the underlying components. There exists existing SOAP containers for most programming languages. This availability allows a rapid development cycle for converting existing source code into a Web Application.

The SOAP request takes the form of an XML document with a header, an envelope and a body containing the payload. The header is optional and is used to provide metadata about the transaction and the envelope defines the namespaces of the transaction while the body contains the payload.

2.4 SOAP Message Hierarchy

A SOAP message is an XML document formatted as follows:



(Image Note: Gudgin, Martin et al)

The role of the SOAP header envelope is encapsulate the message and define it as a SOAP message for other SOAP nodes that will receive and process it. The SOAP header serves as additional information for SOAP nodes delivered in the form of metadata. The SOAP Body is the payload containing the user transaction data. Essentially, the header defines the specific delivery mechanisms for the SOAP nodes and the body declares the content.

Software at the SOAP node must be written to understand the SOAP header block. The developer is free to define SOAP headers as necessary as long as they are well formed. Header definition should comply with the W3C extensibility guidelines.

A SOAP message originates at one SOAP node and terminates at another SOAP node. In many web service deployments a SOAP message will pass through multiple SOAP nodes known as SOAP intermediaries. Each intermediary can process the SOAP message and write data to the header as

needed. The requirement of access to the SOAP Envelope by all nodes and intermediaries cripples the potential benefits of Transport Layer Security technologies such as SSL and IPSEC in a Web Service deployment with more than 2 communication tiers.

2.5 Example message with above elements

```
<env:Envelope xmlns:env="http://www.w3.org/2002/06/soap-envelope">
    <env:Header>
        <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
            <n:priority>1</n:priority>
            <n:expires>2001-06-22T14:00:00-05:00</n:expires>
        </n:alertcontrol>
    </env:Header>
    <env:Body>
        <m:alert xmlns:m="http://example.org/alert">
        <m:msg>Pick up Mary at school at 2pm</m:msg>
        </m:alert>
    </env:Body>
</env:Envelope>
```

(code: Gudgin, Martin et al.)

Notes regarding code example SOAP message:

- Notice the header defines the version of SOAP used in the definition. This will tell the SOAP nodes in the transmission network how to process the data.
- The alertcontrol tag again references its definition for what the alert structure means. Without this definition, the intermediary SOAP nodes would not know what alert standard to apply to the message
- This message has an expiration date that the soap nodes will use for processing. This is critical in many stateless messaging frameworks since you do not have a guarantee that the message was received without a stateful session based framework.
- Code snippet taken from: Gudgin, Martin, et al. “SOAP Version 1.2 Part 1: Messaging Framework”, W3C, June 2002

2.6 SOAP Transactions

2.6.1 SOAP and State

The SOAP messaging model is a stateless protocol. In order for state to be built into a transaction the developer can build session control mechanisms into the SOAP header. The SOAP specification allows for asynchronous communication that can be artificially stateful at best and therefore prone to errors and vulnerabilities associated with session-key management.

State can be built using traditional methods familiar to web programmers experienced programming in a stateless environment. Setting a session attribute in the header tag of the SOAP envelope can simulate an HTTP session cookie. Cookies can be used explicitly as well if HTTP is used at the Transport layer.

2.6.2 Transmission

A Soap message will be transmitted across multiple soap nodes. The SOAP sender is a soap node, the SOAP receiver is a soap node and each SOAP intermediary is both a SOAP sender and a SOAP receiver. This model allows for dynamic routing across any transport layer protocol. The primary transport layer protocol implemented today is HTTP. This distributed transaction model diminishes the benefit of transport layer security and increases the possibility of man-in-the-middle attacks.

HTTP does not fit all scenarios. HTTP is stateless, it is a relatively insecure protocol and it requires that the server and client both be online during communication. Other protocols can be used to transport a SOAP message to accommodate different service level objectives. Other protocols will definitely need to be used if you require an asynchronous transport mechanism.

3. Security Risks

3.1 Firewall Management

The fact that Web Services piggy back on protocols traditionally used for applications such as internet mail and web pages allows web services to communicate across firewalls without explicit permission. An attacker could configure covert application based transactions across firewalls without raising alerts.

This is an important issue. All transactions occur as straight text passed across a transport layer traditionally used for another protocol. Binary messages can be encapsulated in a SOAP envelope with the binary data type.

3.2 Confidentiality

The Web Services Architecture can not rely on Transport layer security because a SOAP intermediary node may need to process the document before forwarding it on to the endpoint. This means that SSL and IPSEC can only be deployed in peer to Peer environments.

A distributed transaction will not be adequately protected by transport layer encryption techniques such as SSL do to multiple connection start/stop points. In the above scenario where you have 2 nodes, that is a sender and a receiver this will work because only the originating and terminating nodes need access to the

SOAP Envelope. In the scenario where you have multiple nodes and intermediaries, you must design message layer security into your deployment.

The important thing to note is that the number of potential SOAP intermediaries increases as the deployment scale increases. This decreases the likelihood of encrypting your data successfully at the transport layer, increases the amount of access to each SOAP envelope and increases the possibility of a passive sniffing attack against your text based SOAP messaging framework. Message layer encryption and digital signatures are a must for a secure deployment.

3.3 Availability

SOAP is an open standard, therefore you have various levels of vendor implementations including an arbitrary number of Home Grown SOAP deployments.

One recommendation of the W3C XML-ENC specification is that any implementation of XML Encryption be able to handle an arbitrary level of recursion of the <`EncryptedKey`> tag. Section 6.4 of the XML-ENC specification lays out the following scenario as an DoS attack against a service processing encrypted XML documents:

> This specification permits recursive processing. For example, the following scenario is possible: EncryptedKey **A** requires EncryptedKey **B** to be decrypted, which itself requires EncryptedKey **A**! Or, an attacker might submit an EncryptedData for decryption that references network resources that are very large or continually redirected. Consequently, implementations should be able to restrict arbitrary recursion and the total amount of processing and networking resources a request can consume.
>
> (Quote: Imamura, Takeshi et al)

3.4 Integrity

How do you know a client transaction is authorized? XML documents can be digitally signed, this is a design decision on the application level. We have seen that transport layer security measures such as Ipsec or SSL are inadequate solutions if you need require end-to-end security in a distributed environment. A Secure web services deployment will require strong message layer security that does not vulnerable to attacks against insecure soap-nodes between the two communication endpoints.

Message layer security should be implemented according to the specifications and standards defined in the W3C documents detailing Digital Signatures and XML Encryption.

4. SOAP Security Extensions defined by the W3C

The W3C specifications for SOAP Security extensions offer a framework for providing Encryption, Digital Signatures and Authorization. These features are implemented in the form of header tags. We are talking about integrating message layer security tags into the SOAP framework in order to provide confidentiality and integrity.

4.1 XML Encryption

SOAP encryption is provided by the XML encryption standard and can be applied at to any element in the SOAP message. This is not unique to SOAP, but applies to SOAP since soap extensibility inherits all features of XML extensibility. The software at each node in the chain must understand the Encryption framework if it is going to process an encrypted attribute.

It is possible to encrypt the entire XML document and it is possible to encrypt elements of the XML document. Encryption of the entire XML document is done via an Octet sequence. It is possible to encrypt data defined by <encrypteddata> elements. This is known as Super Encryption but it should be noted that encryption of <encrypteddata> sub-elements individually is not possible.

As of the writing of this document, the W3C includes specifications for the following algorithms to encrypt XML: Block Encryption, Stream, Key Transport, Key Agreement, Symmetric Key Wrap, Message Digest and Message Authentication.

4.2 SOAP Digital Signature

Again, SOAP benefits from inheriting the extensibility of XML. The XML Digital encryption standard applies to SOAP messages. Implementing a Digital Signature is a requirement in order to guarantee the source of your message in a distributed environment. A SOAP deployment across an uncontrolled distributed environment requires source verification to prevent man-in-the middle and replay attacks.

According to the W3C XML digital signature specification document, an XML digital signature can fall in one of three categories:

- Wrapped Signature
- Detached Signature
- Enveloped Signature

A wrapped signature is very similar to S/MIME or open PGP. A wrapped signature places the signature and the signed content in a single package. A detached signature separates the signature from the signed document but the

signature always contains a URI to the signed document. An Enveloped signature embeds the signature within the signed document. The trick to this is that you have to be able to embed the signature without modifying the hash for the document. An enveloped signature can only be used with XML signed content.

4.3 SOAP Authorization

SOAP Authorization is currently in the definition stage, however any implementation will require the above two(2) components. Consider the scenario where a service provider offers multiple service levels based on an access control list. If the developer were to require every transaction to contain only a digitally signed authorization string she would be deploying a system vulnerable to a replay attack.

The alternative to this is to generate a unique session key and digitally sign the non-predictable session key with the message. In this scenario the encrypted authentication information will be unique with every payload.

Traditional authentication techniques such as SSL, Cookies, Basic HTTP Authentication and Proxy Authentication will work as well. Authentication techniques specific to deployment tools can be used but you should beware when deploying a non-standards based approach. For example, the perl SOAP:Lite module documents the embedding of a username:password string in the url when you connect.

5. Vendor plans and other Proposals

5.1 The evolution of standards

The development of web services is no different than most other cooperative standards based development process. The politics of the matter can not be avoided, I don't think it is accidental that Microsoft and IBM, the co-developers of the proposal in section 5.2 did not participate in the proposal of SAML as discussed in 5.3.

Developers of Web Services should keep this in mind when using vendor tools to develop standards based web services.

5.2 The Microsoft/IBM Hybrid Security Proposal

Microsoft and IBM produced a joint paper addressing deployment details for a vendor neutral secure web services infrastructure.

The following quote was taken from the abstract of the joint proposal put together by Microsoft and IBM. This proposal is titled "Security in a Web Services World:

A Proposed Architecture and Roadmap". There is a link to this document in the reference section of this paper.

> *This document describes a proposed strategy for addressing security within a Web service environment. It defines a comprehensive Web service security model that supports, integrates and unifies several popular security models, mechanisms, and technologies (including both symmetric and public key technologies) in a way that enables a variety of systems to securely interoperate in a platform and language-neutral manner. It also describes a set of specifications and scenarios that show how these specifications might be used together.*
>
> *(Della-Libera, Giovanni, et al)*

The Microsoft/IBM paper proposes a practical vs. ideal combination of existing technologies to secure web services today. The proposal combines the use of the classical technologies and current technologies. This document is a roadmap that will guide the developer interested in combining the following items: SOAP, WSDL, XML Digital Signatures, XML Encryption and SSL/TLS.

The paper addresses the fact that Transport layer security is not adequate unless there are only two nodes, but does not abandon the concept either. The model describes hybrid approaches moving from the simple peer-to-peer to the sophisticated multi-node mobile clients.

5.3 SAML 1.0

Many major players in the industry developed the Security Assertion Markup Language (SAML 1.0). Contributors include sun, Cisco, Verisign, BEA Systems, Entrust and Netegrity.

SAML 1.0 discusses a number of methods of using Security tokens based on a variety of objects. Examples include hardware-based tokens, PKI, SSL, Kerberos and many others.

5.4 WS-Security

The WS-Security specification is a joint specification proposed by IBM, Microsoft and Verisign. This document defines methodology for message level authentication and describes a methodology of associating messages with security tokens.

5.5 XAML

XAML is an initiative taken on by XAML Initiative, Bowstreet, Hewlett-Packard Company, IBM, Oracle Corporation and Sun Microsystems in order to improve

transaction support for Web Services. It is a good idea, but XAML as a spec has not made it very far. As of the writing of this document XAML has not been submitted to any governing body such as W3C or OASIS.

6. Limitations and other Considerations

6.1 Web Services are a collection of evolving standards

There is no typical web service deployment. Two organizations utilizing the same Web Service vendor are likely to use a unique combination of SOAP, XML Encryption, XML Digital Signatures, WSDL, UDDI, XAML XKMS or XFS depending on the use case requirements in question.

The diverse array of standards and technologies means that the Web Services developer will require a high level knowledge of the standards and technologies not just the specific vendor deployment. For example, a developer using Microsoft .NET or Sun One could easily not realize there is a standard for XML based key management deployed by Verisign and Entrust. A developer without this knowledge could find themselves developing there own key management system which will be far less secure and reliable than one that undergoes public scrutiny.

6.2 There is no single solution for securing web services

The current state of the standards, specifications and technology makes the production of hardening guides and security best practices in the near future an unlikely event.

Securing a Web Service will have to be designed into any development project, based on the current state of the technology. The reliance on open standards means no one vendor will have the final answer. This will need to be an item on a development project plan and a secure solution will need to be custom built into any deployment based on the use case considerations in question.

7. Conclusion

Web Services and the technology behind them are certain to have a profound impact on the computing community in the years to come. We have already experienced the industry acceptance of XML as a self-describing data transfer mechanism for heterogeneous computing systems. This proven technology coupled with emerging Standards such as SOAP, UDDI, and WSDL demonstrate that we are very close to full blown deployment.

It is the opinion of the author that we will see the majority of Web Service development happen in B2B exchanges where companies are managing supply chain and partner relationships before we see companies make web services

available to the general public on any significant scale. It is also the opinion of the author that secure web service development is non-trivial at best.

8 References

1. Nielsen, Henrik F., and Satish Thatte. "Web Services Routing Protocol (WS-Routing)", Microsoft, October 2002
   URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-routing.asp

2. Atkinson, Bob, et al. "Web Services Security (WS-Security)" ,Microsoft and IBM April 2002
   URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-security.asp

3. Farrell, Stephen, et al. "Assertions and protocol for the Security Assertion Markup Language", OASIS, May 2002
   URL: http://www.oasis-open.org/committees/security/docs/cs-sstc-core-01.pdf

4. Della-Libera, Giovanni, et al. "Security in a Web Services World: A Proposed Architecture and Roadmap", Microsoft and IBM, April 2002
   URI: http://msdn.microsoft.com/webservices/building/security/default.asp?pull=/library/en-us/dnwssecur/html/securitywhitepaper.asp

5. Ford, Warwich et al. "XML Key Management System" , W3C, March 2001
   URL: http://www.w3.org/TR/2001/NOTE-xkms-20010330/

6. Imamura, Takeshi et al. "XML Encryption, Syntax and Processing", W3C, August 2002,
   URL: http://www.w3.org/TR/2002/CR-xmlenc-core-20020802/

7. Kulchenko, Paul. "The SOAP::LITE Cookbook", Aug 2002
   URL: http://cookbook.soaplite.com/

8. Brown, Allen et al. "SOAP Security Extensions: Digital Signature", W3C, February 2001
   URL http://www.w3.org/TR/2001/NOTE-SOAP-dsig-20010206/

9. Gudgin, Martin et al. "Simple Object Access Protocol (SOAP) 1.1", W3C, May 2000 URL
   http://www.w3.org/TR/2001/WD-soap12-20010709/

10. Christensen, Eric et al. "Web Services Description Language 1.1 (WSDL), W3C, March 2001
    URL: http://www.w3.org/TR/2001/NOTE-wsdl-20010315

11. CERT "Oracle 9iAS SOAP components allow anonymous users to deploy applications by default" , Feb 2002, URL: http://www.kb.cert.org/vuls/id/736923

12. Bellwood, Tom, et al. "UDDI Version 3.0" , OASIS, July 2002
    URL: http://www.uddi.org/pubs/uddi-v3.00-published-20020719.htm

13. Gudgin, Martin, et al. "SOAP Version 1.2 Part 1: Messaging Framework", W3C, June 2002
    URL: http://www.w3.org/TR/2002/WD-soap12-part1-20020626/

14. XML.Com "XAML (Transaction Authority Markup Language)", Oct 2000
    http://www.xml.com/pub/r/811

15. Oracle, “Oracle Security Alert #22”, November 2001,
URL: http://otn.oracle.com/deploy/security/pdf/ias_soap_alert.pdf

16. Robert Taylor, “Microsoft .NET XML Web Services” , SAMS Publishing, December 2001