

# The SOA Magazine

## Feature Article



### SOA Security 101: Patching the Firewall Hole

by Atif Ghauri

Published: August 13, 2008 (SOA Magazine Issue XXI: August 2008)

[Download this article as a PDF document.](#)

*Abstract: Service-oriented architectures have opened and connected “black box” software implementations across enterprises, resulting in a new set of interoperable heterogeneous solutions with the common thread of standard protocols. While this level of integration is unprecedented for enterprise systems, it further muddies the water for application security. The objective of this article is to first introduce the new threats associated with service-oriented solutions, and then provide fundamental design considerations to mitigate the risks resulting from these threats.*

#### Introduction

It's 2008 and it's become clear that SOA is here to stay. The first to implement are realizing the benefits of an open architecture as well as the complexity involved with mapping business functionality to services. Security, however, is often still an afterthought. If considered at all, it is approached with conventional application security mechanisms, such as network and host system controls, identity and access management controls, and data encryption during storage and transit. Is this enough to mitigate the security risks in an SOA?

Well, if the security considerations that applied to traditional application architectures applied to SOAs, then the aforementioned controls would be sufficient. But unfortunately service-oriented solutions are different. Unlike the interoperability challenges that stunted the growth of Common Object Request Broker Architecture (CORBA), today's SOAs are interoperable, relying often on Web services based on standards such as XML for encapsulation and HTTP/HTTPS for transport. It is in fact this interoperability and interdependence between entities that has influenced a new frontier of security threats.

Unlike conventional application architecture, service-oriented solutions that expose Web services to external users and business partners are more susceptible to unauthorized access. Traditionally application architectures allowed limited external access with adequate security controls in place to do so. Message-level security was unnecessary because all processing was carried out internally, within the application. Instead the emphasis of security was on how the message was transported. The contents of the messages were assumed to be valid since they were coming from one trusted source within the organization.

In a service-oriented infrastructure, a solution invites all HTTP/HTTPS traffic through its firewalls, and does little to differentiate between a malicious or valid message. A typical organization's border and internal firewalls are packet filtering and “allow” all HTTP/HTTPS traffic back to the application server from an uncontrolled zone to a DMZ to a secure zone without inspection. Thus, the service-oriented solution has introduced a hole in the firewall.

#### Understanding the Risks of Exposure

To offer a rudimentary understanding of the new threats, we'll consider them in the context of a sample scenario with “Big Bank”. Suppose Big Bank is developing an application to issue checks and has implemented a Web service called “cut\_check”. The cut\_check service may be accessed by other Big Bank organizations and business partners on

disparate networks across the world. Following best practices, the application will have its Web server in the DMZ and application server (containing business logic) in a separate security zone, away from the DMZ, protected by packet filtering.

Conventional application security architecture would not allow direct access to the cut\_check service running on the application server, but instead would only accept requests through a proxy, such as an HTTP Web server in a DMZ. The cut\_check service would process each transaction at the application server and would then present the results upon completion of each transaction to the Web server to display back to the user.

As illustrated in the top of portion of Figure 1, the user makes a request to the application, the application makes a request to the service, and the response is provided back to the user. However, as shown on the bottom portion of Figure 1, a service-oriented solution introduces new methods of invocation. The service may be directly exposed to the user, invoked by another Web service, or it may invoke a service directly. The service-to-service invocation can result in a chaining or composition of services to perform a specific operation.

For example, at Big Bank the clerk may issue cut\_check which may rely on another service such as calc\_taxes (that determines the amount of tax to be drawn prior to issuance). Consequently, the service-to-service use cases also require security controls similar to user-to-service scenarios.

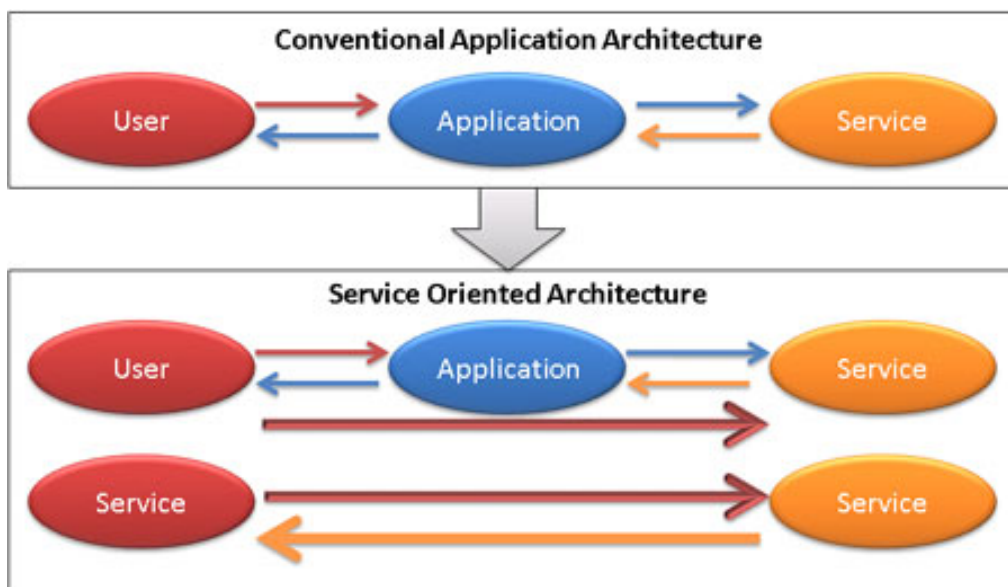


Figure 1: The shifting architectural paradigms introduced by SOA.

Furthermore, SOA-based applications are incubating a new breed of security threats to Web based applications. There are numerous attack scenarios around Web services already being exploited and the list keeps growing. The types of attacks include, but are not limited to: denial of service, replay, message alteration, breach of confidentiality, man in the middle, and spoofing attacks.

In particular, XML attacks tend to be the most pervasive through well-formed SOAP messages embedded in HTTP/HTTPS protocols. Overall, there are four broad classifications of threats associated with XML, as summarized in Table 1.

## Common XML Attack Categories

<b>XML Denial of Service (xDOS)</b>	<p>xDOS attacks will exhaust a web service so that valid service requests are hampered or denied. Examples include:</p> <p><b>Jumbo Payloads</b> – Sending a very large XML message to exhaust memory &amp; CPU on the target system</p> <p><b>Recursive Elements</b> – XML messages that can be used to force recursive entity expansion or other repeated processing to exhaust server resources</p> <p><b>XML Flood</b> – Sending thousands of otherwise benign messages per second to tie up a Web Service</p>
<b>Unauthorized Access</b>	<p>These attacks attempt to gain unauthorized access to a web service or its data. Examples include:</p> <p><b>Replay Attack</b> – Re-sending a previously valid message for malicious effect, possibly where only parts of the message (such as the security token) are replayed</p> <p><b>Dictionary Attack</b> – Guessing the password of a valid user using a brute force search through dictionary words.</p> <p><b>Falsified Message</b> – Faking that a message is from a valid user, such as by using Man in the Middle to gain a valid message and modifying it to send a different message.</p>
<b>Data Integrity &amp; Confidentiality</b>	<p>These attacks strike at data integrity of web service responses, requests, or underlying databases. Among the most common rely on a replay attack which may use a signed message to replay a message unless it uses time stamps that are cached and signed by a validated certificate authority. Examples include:</p> <p><b>Message Tampering</b> – Modifying parts of a request or response in flight, most dangerous when undetected; also known as "Message Alteration".</p> <p><b>Message Snooping</b> – A direct attack on data privacy by examining all or part of the content of a message. This can happen to messages being transmitted in the clear, transmitted encrypted but stored in the clear, or decryption of messages due to stolen key.</p> <p><b>SQL Injection</b> – Modifying SQL in XML to obtain additional data than what the service was designed to return.</p>
<b>System Compromise</b>	<p>These attacks corrupt the web service itself or the hosting server. Examples include:</p> <p><b>XML Virus (X-Virus)</b> - Using SOAP with attachments or other attachment mechanisms to transmit malicious executables such as viruses or worms.</p> <p><b>Malicious Include</b> – Causing a Web service to include invalid external data in output or return privileged files from the server file system. For example, using embedded "file:" URLs to return password files or other privileged data to the attacker.</p> <p><b>Memory Space Breach</b> – Accomplished via Stack Overflow, Buffer Overrun or Heap Error, allowing execution of arbitrary code supplied by the attacker with permission of host process.</p>

Table 1 - Common types of XML attacks.

The astute architect might suggest that transport layer security mechanisms such as Secure Sockets Layer (SSL) or

Transport Layer Security (TLS) addresses the XML threat. However, security at the transport layer of the OSI model is just not enough. It doesn't address the context of SOA scenarios because it doesn't protect against malicious content included in the message. It simply encrypts the malicious code embedded in the XML message written by a, perhaps, authorized user with access the service. Once the packet is opened by the application it is too late.

Thus network and host system controls, identity and access management controls, and data encryption controls during storage and transit must be supplemented with additional controls to protect the message payload. This additional control is required given that the application is now open to a broader audience and is accessed with standard protocols instead of proprietary extensions.

## SOA Security Design Considerations

A layered security architecture provides the best defense for a service-oriented architecture. The most effective architectures certainly use strong network and host system controls, identity and access management controls, and data encryption during transit and storage, but also rely on Web service security. Two fundamental design considerations used to secure Web services focus on packet and message-level security.

### Packet Inspection

Since SOAP messages containing XML payloads traverse across runtime service activities using the HTTP/HTTPS protocols, packet filtering firewalls and routers do not provide adequate security controls. Packet filtering devices will not be able to differentiate authorized well-formed XML messages with malicious messages. However, XML firewalls can bridge this gap.

First, XML firewalls provide packet level inspection of all inbound and outbound traffic to the back-end web application server. For inbound traffic, the device tears open packets to scan its contents against a catalog of threats before it forwards it on to the application for processing. For outbound traffic, the device may apply response filters to inspect packets for sensitive information inadvertently included in the response. Response filters can serve as a critical component to prevent confidential information from leaving an organization through Web services.

Next, the XML firewall must be strategically placed on the network. Depending on the application requirements, it is most typically placed in a DMZ security zone in front of the web application server, as illustrated in Figure 2. Coupled with the traditional packet filtering firewalls, placement in the DMZ offers the ideal position for the XML firewall to protect the business logic harvested in the application server. As the services become exposed to external networks beyond the host organization's control, additional XML firewalls may be deployed to add more depth of security across the network.

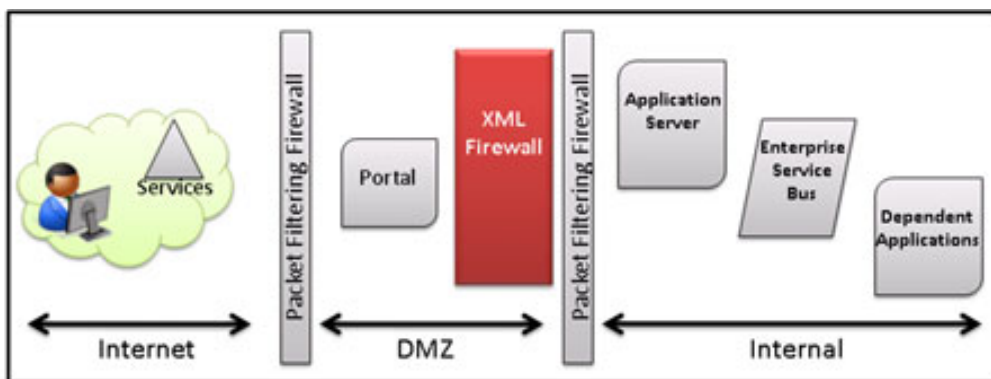


Figure 2: A typical XML firewall architecture.

Furthermore, many of the major application servers, such as IBM WebSphere and BEA WebLogic, offer embedded packet inspection services. These value-add features may be appropriate depending on the amount of messages the server processes and the performance requirements of the application. Considering the processing overhead of packet inspection, the use of these embedded features may introduce significant overhead as the application scales. The industry best practice is to offload this security processing into a hardware-based XML firewall, such as IBM Data Power, Intel Sarvega, or Cisco Reactivity. Using a hardware device also adds an additional layer of protection into the overall security architecture.

## Message-level security

In addition to packet inspection from an XML firewall, Web services may also be protected using message-level security. The use of standards offers a consistent framework for developers to implement security at the message level, especially since the information security community has helped mature Web services security standards over the past few years. Figure 3 describes the security mechanisms available in reference to layers of the Open Systems Interconnection (OSI) model.

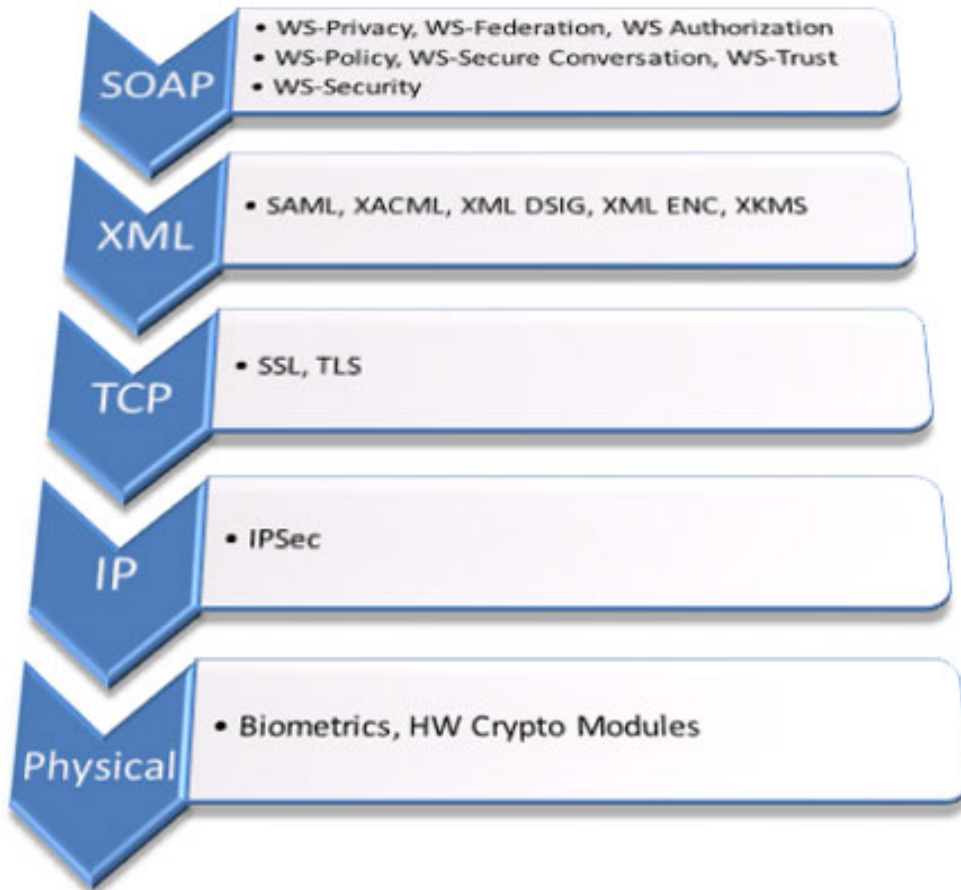


Figure 3: Security protocols categorized by OSI layer.

Most notable in Figure 3 is the SOAP security stack highlighted by WS-Security. The abstract from the WS-Security standards document states that WS-Security is an enhancement to SOAP messaging. The intent is to describe security extensions that may be added to SOAP messages to provide improved security in three areas: message integrity, message confidentiality, and message authentication. First, message integrity ensures that changes to messages are detected. Message confidentiality provides protection against unauthorized disclosure of message contents. Finally, single message authentication restricts the use of Web services to authorized users only.

A key feature of the WS-Security standard is that it makes use of existing technologies instead of inventing new ones. Specifically, WS-Security calls upon the following proven security techniques:

- XML Digital Signatures - Used to ensure that a message is not altered while in transit.
- XML Encryption - Used to encrypt messages to make them unreadable except by those possessing the proper keys.
- Authentication Tokens - A mechanism to restrict access to Web services to only those users that are authorized with supported authentication types: usernames and passwords, X.509 digital certificates, Security Assertion Markup Language (SAML).

While there are considerable advantages to the use of message-level security, it must be used with caution. The use of standards requires additional due diligence because they certainly do not provide a “one-stop shop” for security architects. Standards are, after all, standards and often don’t include implementation specifics that must be extracted

from the application requirements.

For instance, the WS-Policy specification offers security governance for encryption but it does not specify all of the key exchange properties necessary to make this work in the real world. The exchanging parties must still determine and agree upon on a key size, cipher, and algorithm. Also, Web security standards provide little support beyond SOAP and HTTP/HTTPS. Some solutions may still require proprietary specifications and technologies to fully protect their perimeters.

Next, if the technology infrastructure is not sized correctly, the advantages provided by message-level security may result in detrimental performance. Despite these inconsistencies, an architecture based upon standards is still far superior than relying on proprietary methods.

## Conclusion

The proliferation of Web services has pushed the business case for SOA, for they have enabled unparalleled interoperability among a heterogeneous set of enterprise applications. Along with the increased integration, there has been a paradigm shift on how to secure an application given the new frontier of attacks on interdependent enterprise solutions. The new generation of attacks categorically include threats based upon XML denial of service, unauthorized access, data integrity, data confidentiality, and system compromise.

As the proverbial saying goes, "Security is only as good as its weakest link." The external exposure of Web services has warranted inspection of the transaction payload and the use of message-level security. Thus conventional security controls must be amended to include Web service security mechanisms for service-oriented solution designs. The inclusion of XML firewalls for packet inspection and message-level security are therefore critical for the protection of SOA.

Industry standards are available and continue to evolve, but do not provide a comprehensive solution. The onus is on the security architect to use the standards as a baseline and devise appropriate security controls applicable to the specific implementation of the service-oriented solution.

THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL



[Home](#) [Past Issues](#) [Contributors](#) [What is SOA?](#) [SOA Glossary](#) [SOA School](#) [SOA Books](#) [About](#) [Legal](#)

Copyright © 2006-2008  
SOA Systems Inc.