Executive Guide to Web Services Security

A Reactivity White Paper

October 2004

ABSTRACT
Businesses are rapidly adopting Web services to provide new levels of integration between applications. By comparison with earlier data-communications techniques, Web services are faster and cheaper to develop, quicker to deploy, and easier to adapt to emerging business needs.

Although these benefits are real, and more and more companies are adopting Web services for that reason, the same characteristics that make Web services quicker and cheaper to deploy, more robust and more flexible than older methods, also make them vulnerable to new kinds of security risks and opportunities.

This paper discusses the special security challenges posed by the use of Web services, and how to secure networks against them.
Reactivity, Inc.
The Executive Guide to Web Services Security

Contents
Special Advantages, Special Risks ................................................................. 3
Making Sense of Standards ........................................................................ 4
Standard Description .................................................................................... 5
  Which standard fits your need? ................................................................. 6
Trust and Threats in the Web Services Paradigm ........................................... 6
  Malicious Intent or Human Error? .............................................................. 7
  Defending Against Identity Attacks .......................................................... 7
  Understanding XML Virus Attacks ............................................................. 9
  Defending Denial of Service Attacks ........................................................ 10
Making the Architecture Work ..................................................................... 11
  The Importance of Logging ...................................................................... 11
Conclusion ..................................................................................................... 12
About Reactivity .............................................................................................. 12

Figures
Figure 1: Well-established and emerging standards for web services ............... 5
Figure 2: Traditional Internet threats relevant for Web services ...................... 7-9
Special Advantages, Special Risks

The great advantage of the Internet is that it is universally accessible. Because it consists of thousands of freely-communicating networks all over the world, the Internet provides a communication infrastructure that reaches everyone, an infrastructure that a business can use without significant new capital investment.

Similarly, Internet standards define communication protocols and data formats that enable anyone to make network connections and transmit data, and rely on the fact that their messages will be received and understood.

When someone sends a message in a standard format using a standard protocol, the protocol ensures that the message can be delivered correctly, and the data format ensures that the receiver can read it.

Unfortunately, these very advantages make Web services and other Internet technologies uniquely vulnerable to attack. Because the Internet reaches everyone, anyone can use it—not just honest people engaged in legitimate business, but vandals, criminals and other abusers of the network.

The universal nature of the Internet enables these unscrupulous users to intercept legitimate communications and connect to others’ systems. Similarly, the standardization of Internet protocols and data formats enables them to read, understand, and even forge communications between legitimate users.

The openness of XML and Web services lets you cost-effectively conduct strategic operations with customers and partners. Openness cuts both ways, however.

While standards-based solutions claiming to solve “the security problem” flood the market, the problem encompasses more than security. Securing your Web services must take into account multiple connections to individual vendors, strategic partners, and customers. These connections are revenue pipelines, so measures must assure security and enable rapid customer acquisition. That’s why standards aren’t enough.
Securing Web services to maximize their benefits requires:

- A discriminating approach to supporting standards
- The ability to defend against new, crippling XML threats while connected to many different types of services and networks
- A scalable foundation that enables both rapid and repeated provisioning and optimizes the Web services or SOA team to capture new business opportunities.

Only when these three critical elements are incorporated into a Web services architecture or SOA can you reliably secure Web services and capture the flexibility and cost savings they promise.

Making Sense of Standards

Saying a Web services security solution is “standards-based” is helpful, but incomplete. There are dozens of standards. Some apply to specific industries; some apply to specific security technologies. Baseline functionality and compatibility standards can’t adequately protect all businesses and all Web services. In addition, malicious threats emerge and quickly mutate, making it difficult to ensure that security measures are always up to the threat. Companies of all sizes spend significant effort in creating services; it’s important that you choose the standards that best support your needs. The answers to these questions can help you begin:

- Which standards are the most established and reliable? Approved or still emerging?
- Which standards are most beneficial to support for our company, partners and customers?
- Which standards are required for our industry in terms of compliance or operability?
- Which standards enable rapid deployment of new services and interoperability?
- Can emerging standards be easily added to our Web services architecture?

The standards described in Figure 1 are commonly used in today’s Web services for enabling rapid deployment and interoperability.

Organizations deploying Web services should incorporate standards, ranging from HTTPS through XML Signature, yet keep their security plan open enough to accommodate the future addition of Security Assertion Markup Language (SAML) and WS-Trust.

A Web service interface is an exposed, standards-based integration point for your applications. It must be able to accommodate a wide range of security sophistication from partners and customers who connect to it. The most widely deployed standard, Secure Socket Layer (SSL), is a basic security building block. Early Web services were secured only with two-way SSL. Today, only supporting SSL in Web services significantly limits the service’s long-term functionality and overall enterprise security.
## The Secure Web Services Deployment System

<table>
<thead>
<tr>
<th>Standards Stack</th>
<th>Standard Description</th>
<th>Adoption Today</th>
<th>Trend</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPS</td>
<td>An HTTP connection secured between the client and host using SSL/TLS - a &quot;secure pipe&quot; that ensures the confidentiality of the information transmitted over the public Internet.</td>
<td>Very High</td>
<td>Stable</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language. A particular text markup language for interchange of structured data.</td>
<td>Very High</td>
<td>Stable</td>
</tr>
<tr>
<td>XML Schema</td>
<td>A language for describing the structure and constraining the contents of XML documents.</td>
<td>High</td>
<td>Growing</td>
</tr>
<tr>
<td>SOAP</td>
<td>A standard that defines application-level structure for messages.</td>
<td>Moderate</td>
<td>Growing</td>
</tr>
<tr>
<td>WSDL (Web Services Definition Language)</td>
<td>Effectively the &quot;URL&quot; for a specific web service. Expresed in XML, a WSDL definition describes how to access a web service and what operations it will perform.</td>
<td>Moderate</td>
<td>Growing</td>
</tr>
<tr>
<td>WS-Security (Web services Security)</td>
<td>A mechanism for incorporating security information into SOAP messages.</td>
<td>Moderate</td>
<td>Growing</td>
</tr>
<tr>
<td>XML Encryption</td>
<td>A process for encrypting and decrypting parts of XML documents. A subset of the standard is used by WS-Security to maximize interoperability.</td>
<td>Moderate</td>
<td>Growing</td>
</tr>
<tr>
<td>XML Signature</td>
<td>A mechanism for ensuring the origin and integrity of XML documents. A subset is used by WS-Security to maximize interoperability.</td>
<td>High</td>
<td>Growing</td>
</tr>
<tr>
<td>SAML (Security Assertions Markup Language)</td>
<td>A framework for exchanging authentication and authorization information.</td>
<td>Moderate</td>
<td>Growing</td>
</tr>
<tr>
<td>WS-Trust</td>
<td>A standard for creating networks of federated trust.</td>
<td>Low</td>
<td>Growing</td>
</tr>
</tbody>
</table>

Figure 1: Well-established and emerging standards for web services
The Secure Web Services Deployment System

Which standard fits your need?

Today, SSL secures HyperText Transfer Protocol (HTTP) connections (i.e., HTTP over SSL [HTTPS]) and information in transit. It’s important, but not enough. Simply relying on HTTPS creates three problems:

- The Web service must undertake considerable private key and certificate management.
- There are no guarantees of message confidentiality or integrity.
- There’s no auditable record of the message, session, or security enforced.

Additional standards should be included in Web services security architecture, too. For example, an increasing number of Web services and Web applications are written using Simple Object Access Protocol (SOAP). SOAP specifies how to encode HTTP headers and XML files so applications running on different systems can successfully pass information back and forth. Web services designed to communicate with partners and customers increasingly use SOAP so they can communicate with programs anywhere.

Web services applications must be able to verify XML digital signatures and quickly encrypt and decrypt. Applications using this functionality are most efficiently developed and deployed on dedicated infrastructure to optimize performance.

The WS-Security specification provides a way to ensure that messages remain confidential, haven’t been tampered with, and are actually from senders asserting to have sent them. WS-Security specifies the use of XML Digital Signature and XML Encryption within SOAP, enabling the application developer to insert a security token that identifies the original sender and optionally captures information about intermediate destinations of the XML message. Security tokens can be as simple as a name, IP address, and password; more complex, such as a Public Key Infrastructure (PKI) certificate; or as comprehensive as a SAML assertion.

Finally, SAML is used for user identity assertions and for asserting actions performed by various elements of an enterprise infrastructure. For example, if a Web services security gateway performs the necessary authentication, authorization, encryption, digital signature, and other security functions, it can insert a SAML token that’s accepted by a Web service, asserting that it can accept and process the message.

You should regularly and rigorously test your implementations of whatever standards you decide to support. Standards continue to evolve and their implementation can vary considerably. Consider, for example, PKI. It predates Web services by 10 years and the standards for PKI still require significant interoperability efforts.

Trust and Threats in the Web Services Paradigm

The openness of XML and Web services lets you cost-effectively conduct strategic operations with customers and partners. Openness cuts both ways, however. Widespread use of XML and Web services makes it significantly easier for outside, uninvited parties to integrate with systems and invade applications. The results can range from annoying service glitches to privacy breaches all the way to catastrophic system failures and data loss.

Determining who to trust and creating a comprehensive XML defense model is vital. Your Web services architecture must be flexible enough to manage different levels of defenses and security sophistication among your connection partners.
Malicious Intent or Human Error?

As systems are more connected to each other over the Internet, the number and severity of attacks rises. Ray Wagner of Gartner Group predicted in October 2003 that, by 2005, Web services will have reopened 70 percent of the attack paths closed by network security infrastructure.

New XML and Web services expose critical corporate assets to customers and business partners. For example, worms and viruses have the potential to create disastrous business conditions. Combining easy access with human-readable data formats and open integration standards creates an almost irresistible attraction for malicious hackers. Malicious Web services threats typically fall into one of three categories:

- Identity threats, which are new XML versions of traditional identity threats such as authentication attacks and eavesdropping
- Content-borne threats, which are attacks with elements in the actual XML payload, such as XML viruses
- XML Denial of Service (XDoS), which are new, application-level versions of network-level DoS attacks.

In addition, inexperienced developers often err, producing situations that resemble outside attacks, but, in fact, are simply accidents. These mistakes, though benign, still cost downtime, require IT remediation, and can disrupt revenue-generating services.

Defending Against Identity Attacks

Traditional identity threats include authorization and authentication attacks where hackers steal identities, attempt to “spoof” the service itself, or attempt to use permitted access to reach restricted resources. Eavesdropping attacks enable a hacker to read and potentially alter messages flowing between you and your business partners. In attacks such as these, an attacker can either access your systems or redirect and collect messages between you, your customers, and partners. The use of standards such as WS-Security and SSL can reduce the likelihood of identity attacks.

<table>
<thead>
<tr>
<th>Attack</th>
<th>Countermeasure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Authentication</strong></td>
<td>Prove the identity of each user of a system. Many systems demand usernames and passwords to authenticate requests, but this method may not be very secure. A more secure solution is to use a cryptographic technology such as SSL to establish a secure connection between the user and the service, and then exchange digital certificates to ensure that each party is who it claims to be.</td>
</tr>
<tr>
<td>• Request authentication attack - an attacker pretends to be an authorized user, the Service will grant him the same access and privileges as that authorized user. The attacker can then use the service and any information or other resources it provides, using these privileges.</td>
<td></td>
</tr>
<tr>
<td>• Response authentication attack - An attacker can also pose as the service, rather than as the user. Imagine that a legitimate user sends a request to a valid service, but an attacker is eavesdropping. The attacker can pose as the legitimate service and can request confidential information or payments. Phishing is a variant of this attack.</td>
<td></td>
</tr>
<tr>
<td>Attack</td>
<td>Countermeasure</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Authorization</td>
<td>An authenticated user obtains access that he should not have to services, data, or other resources. If the service allows the access, the attacker can then collect all accessible confidential data, access sensitive systems, issue dangerous commands, and so on. For example, attackers often use compromised machines to launch attacks on other systems, covering their tracks by using someone else’s systems to do their dirty work.</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>Cryptographic tools provide the most effective protection against loss of confidentiality, enabling networks to transmit sensitive data in an encoded form that is useless to attackers. If an attacker succeeds in intercepting an encrypted message he has gained nothing because the message is unreadable without the keys needed to decode it. Encryption technologies such as SSL enable systems to encrypt individual messages, or to encrypt communications channels so that every bit of data that passes from one system to another is encrypted. The most secure solutions use both methods—encrypting data channels so that no outsiders can eavesdrop on communications, and encrypting the individual messages so that they are unreadable even by unauthorized insiders.</td>
</tr>
<tr>
<td>Data Integrity</td>
<td>The simplest, most effective technique is to use cryptographic tools to protect the data channels and the contents of messages, as explained above in the confidentiality section. Cryptographic tools also provide techniques such as digital signatures, which can guarantee that a message cannot be altered without the receiver knowing about it. Content-analysis tools can also use technologies such as DTDs and XML Schemas to analyze the contents of messages to determine whether they meet certain requirements.</td>
</tr>
</tbody>
</table>

Confidentiality threats are serious matters; they can result in identity theft, embezzlement, fraud, leakage of trade secrets, and many other serious problems.
### Understanding XML Virus Attacks

A brilliant feature of the Web is that it uses standard ports for all communications—generally port 80 for all HTTP traffic. Port 80 is typically opened to the world, while other ports, such as File Transfer Protocol (FTP), are guarded more closely. The "port 80 problem" is that viruses and malicious content can be included in innocuous legitimate content and tunneled through port 80 to reach inside an organization. Content-borne attacks are generally intended to affect the actual applications that run Web services after tunneling unnoticed through the security infrastructure. Content-borne attacks are also known as XML viruses or XML worms.

Two examples of content-borne XML exploits are SQL injection attacks and buffer overflow attacks. SQL injection is the practice of inserting malicious SQL statements into XML to disrupt back-end systems. If a Web service connected to a database doesn’t validate SQL, an incoming XML message containing rogue SQL statements could break out of the expected database query and be used to obtain unauthorized information or destroy data. In fact, SQL injection attacks are a subset of a broader class of attacks known as command injection attacks. Like using malicious SQL code to attack databases, hackers attempt to tunnel Unix commands inside XML to exploit any system that has a command-oriented interface.

Like SQL and command injection techniques, a buffer overflow attack is aimed at the service endpoint and preys on vulnerabilities there, such as not setting aside enough memory to deal with a large variety of inputs. One example would be a Web service designed to take in a list of phone numbers.

Another example of a "XML Virus" or content borne attack is a Content format attack that exploits vulnerabilities in the ways that services read content formats (document types, element names, attribute names, etc.) before they examine the actual content. Web services integration relies upon standards to structure interactions between parties. To exchange information, applications format content in their requests and responses according to supported standards.

One such attack, **entity expansion**, exploits a capability in DTDs that allows for the creation of custom macros, or entities, that can be used throughout a document. By recursively defining a set of custom entities at the top...
of a document, an adversary can overwhelm parsers that attempt to completely resolve the entities by forcing them to iterate indefinitely on these recursive definitions. Other attacks include inserting extremely large element or attribute names into an XML document in an attempt to overload a parser’s resources.

**Countermeasures**

Protection against content attacks requires robust parsing and XML Schema validation capabilities. Before passing content to a service, the security solution’s parser checks for abnormal conditions such as unusually large element and attribute names. In addition, the parser should either detect recursive entity definitions or expand entities only partially before signaling failure. A good solution involves using schema validation in conjunction with a second, more sophisticated pattern matcher that detects suspicious patterns such as SQL statements and commands. Services should only process content that successfully passes through both validation steps.

**Defending Denial of Service Attacks**

The third type of XML and Web services attacks are DoS attacks. These attacks tend to result in services being unusable for everyone. Because these attacks are difficult to distinguish from legitimate traffic, it becomes doubly difficult to selectively service only legitimate requests. New XDoS attacks have similarly thorny issues. The art of defending against XDoS is to detect an attack based on a combination of metrics that signify an attack, not just one metric viewed in isolation.

One of the first widespread XDoS attacks was the “entity expansion” attack. Here, unprivileged users could use completely “correct” entity declarations in an XML message to wreak havoc on unprotected XML 1.0 standard-compliant parsers. When a vulnerable parser encountered such a message, recursive entity declarations caused the parser to shut down with an out-of-memory error or to use an inordinate amount of processor cycles. Inadvertent DoS attacks can occur as the result of simple human error, such as a programmer on the other end mistakenly sending 100 requests per second instead of 10, or doing something as simple as accidentally coding an infinite loop.

XDoS and certain authentication attacks can be only detected with configurable heuristics. For example, there may be anywhere from three to eight indicators that XML traffic is actually an XDoS attack. These signals aren’t only generated by traffic from outside the enterprise but also from the response rate of Web services within the enterprise. You must be able to monitor those signals in real-time, over time, to ensure that abnormalities are noticed and handled. A sophisticated approach uses a graduated response to handling abnormalities. These range from alerts to throttling and finally, IP blocking—all accompanied by secure, sophisticated logs that let administrators trace events.
Making the Architecture Work

Choosing your supported standards and building an XML threat defense model are good first steps. However, many architectures fall short when it comes to deploying a workable, repeatable process. Many let you successfully secure a single Web service and program all the code necessary for standards, threat defense, and security policy. However, as Web services are connected to heterogeneous environments, they’re subject to many requirements in addition to security requirements. Services based on these “code-it-in” architectures quickly become inefficient. All security processing must be done in the Web service itself. Each new Web service requires new programming. Older services require reprogramming and upgrading to successfully defend against new or evolved XML threats; all of which seriously impede your ability to quickly provision new partners and revenue-generating services.

Instead, look for solutions that:

1. Let you securely connect Web services with internal or external business partners quickly, reliably, transparently, and manageably.

2. Enable centrally defined coarse and fine-grained security policies. (Different users in different groups can specify a scalable Web services security solution, and it employs intelligent policy coordination for consistent enforcement.)

3. Optimize the processes that your Web services team has to do every time - create and provision services and connections, create, approve, and record policies, migrate services and policies between environments and transactionally deploy policy.

4. Enable any-to-any integration for platform, protocol, and standards mediation with a deny-by-default architecture that ensures only trusted messages reach your services. This provides the most reliable security and extends the longevity of your Web services architecture while reducing testing time in heterogeneous environments.

5. Provide detailed, configurable, and collaborative event and message logs that help you instantly identify and anticipate issues such as the need to check an expired certificate.

6. Provide comprehensive, flexible support for failover, load balancing, and capacity planning.

The Importance of Logging

Many of the issues that arise when deploying and scaling secure Web services can continue for some time undetected, doing damage to the affected services the whole time. For example, once an attacker has defeated an authentication or authorization scheme and gained access to sensitive resources, the attacker can exploit those resources repeatedly. Similarly, once an attacker discovers how to create a forged message that gets effective results, he can send it over and over. In addition, the task of debugging is infinitely more complicated with encrypted messages that needed to be considered as part of the troubleshooting process. Finally, in this era of scrutiny and compliance, a secure record of the security enforced, policy in force, and the messages themselves is crucial to compliance. And all of this functionality is delivered through searchable, policy aware logs.

Logging is an important diagnostic and compliance tool for managers of business networks. Services, and the Gateways that protect them, must keep accurate logs of the kinds of traffic that pass through them, and if possible the contents of the messages. By examining logs, network administrators can quickly identify and
diagnose potential problems, and take steps to prevent or correct damage. Logs are important in protecting services from denial-of-service because the only reliable way to identify the threat is to detect a sudden increase in volume of messages from one or a few addresses. Sophisticated security products, such as service Gateways that perform content analysis, can even examine logs and alert network managers to potential problems.

**Conclusion**

The growing adoption of Web services in business represents a great opportunity for those businesses to improve their time-to-market with new services, to lower the cost of business communication, and to offer new services to customers and partners at modest cost. These benefits are so compelling that even the threat of serious security breaches has not prevented the adoption of Web services, but it has prevented businesses from enjoying the full benefit of those services. An informed and comprehensive approach to threat prevention, detection, and correction is essential before the full benefit of Web services can be realized.

Is there more to securing Web services than standards? Yes. Are there solutions that offer a more comprehensive approach? Yes, again. Variously called XML firewalls, secure Web services gateways, or security gateways, new dedicated products address the security risks, policies, and standards associated with Web services and are optimized for the deep content inspection this effort requires.

Companies that rely on high-value transactions conducted across Web services must consider more than standards. Now, you can more easily evaluate solutions that help you reliably deploy a secure Web services infrastructure and reinforce security policy—even as complexity and risks increase.

**About Reactivity**

Reactivity provides the leading secure deployment infrastructure used by enterprises to realize the promise of Web services. Reactivity’s family of products enables businesses to secure, provision and operate XML and Web services more efficiently and effectively, accelerating their time-to-market and gaining competitive advantage in their businesses. Reactivity has unique experience with the largest and most demanding Global 2000 enterprises in the financial services, publishing, travel and other industries. Reactivity is located in Belmont, CA and can be reached at [www.reactivity.com](http://www.reactivity.com), +1-650-551-7800 or [info@reactivity.com](mailto:info@reactivity.com).