Executive Guide to Web Services Security

Abstract

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

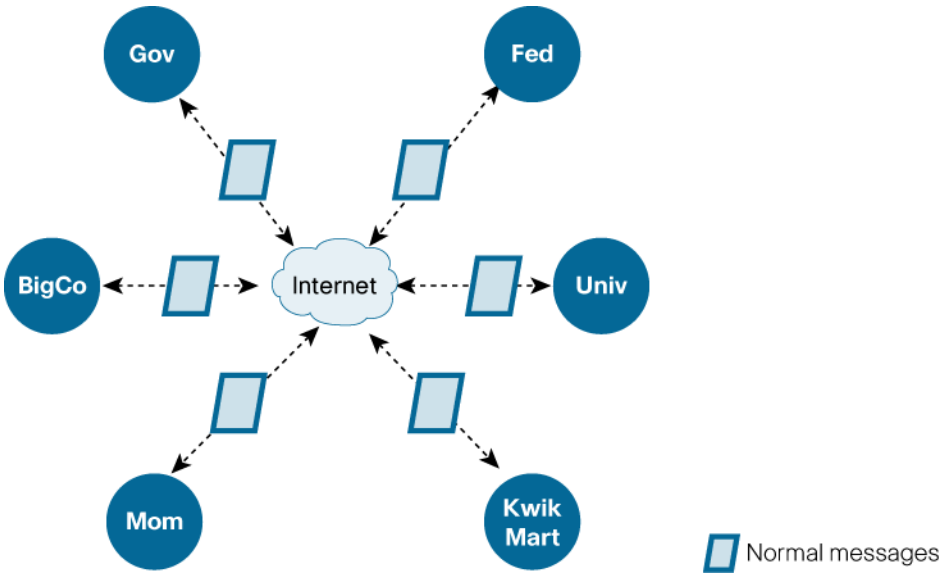
Although these benefits are real, and more and more companies are adopting Web services for that reason, the same characteristics that make Web services quicker and cheaper to deploy, more robust, and more flexible than older methods also make them vulnerable to new kinds of security risks and vulnerabilities.

This document discusses the special security challenges posed by the use of Web services and how to secure networks against them.

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 be able to assume that their messages will be received and understood. When someone sends a message in a standard format using a standard protocol, the protocol helps ensure that the message will be delivered correctly, and the data format helps ensure that the receiver will be able to read it (Figure 1).

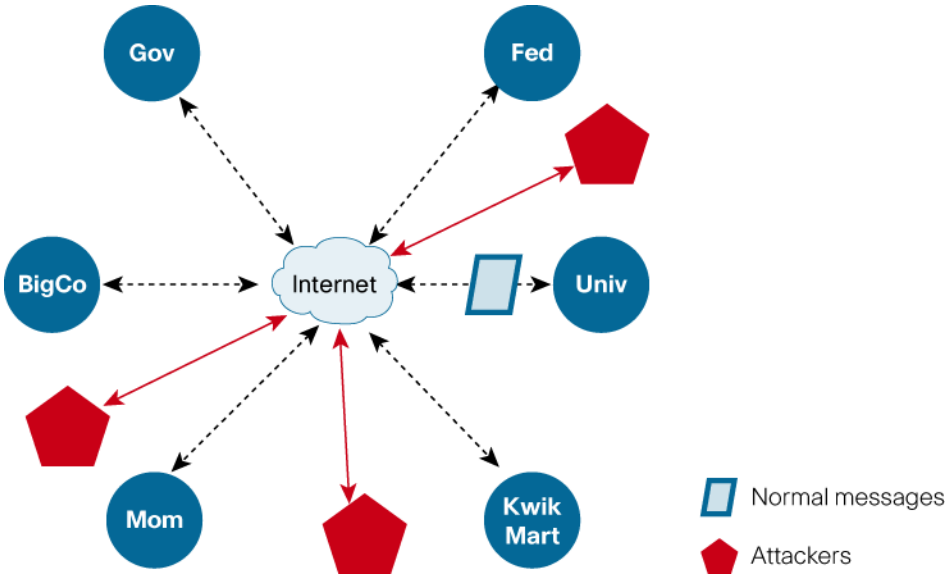
Figure 1. Messages Sent Using Standard Protocols



Unfortunately, these same advantages make Web services and other Internet technologies uniquely vulnerable to attack. Because the Internet reaches everyone, anyone can use it: not just

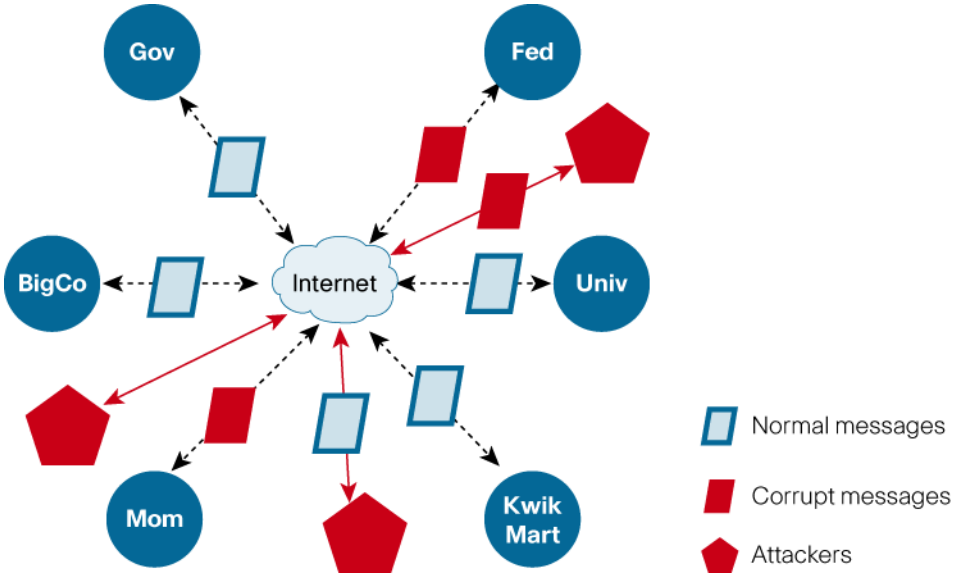
honest people engaged in legitimate business, but vandals, criminals, and other abusers of the network. The universal nature of the Internet enables these unscrupulous users to intercept legitimate communications and connect to others' systems. Similarly, the standardization of Internet protocols and data formats enables them to read, understand, and even forge communications between legitimate users (Figure 2).

Figure 2. Message Intercepted by Attacker



The openness of Extensible Markup Language (XML) and Web services lets you cost-effectively conduct strategic operations with customers and partners. However, openness cuts both ways. Although standards-based solutions claiming to solve "the security problem" are abundant, the problem encompasses more than security. Securing your Web services must take into account multiple connections to individual vendors, strategic partners, and customers (Figure 3). These connections are revenue pipelines, so measures must help ensure security and enable rapid customer acquisition. That is why standards are not enough.

Figure 3. Potential Attacks on Messages



Securing Web services to maximize their benefits requires the following:

- A discriminating approach to support of standards
- The ability to defend against new, potentially 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 service-oriented architecture (SOA) team to seize new business opportunities.

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

Making Sense of Standards

Choosing a Web services security solution that is standards based is appropriate, but only a starting point. There are dozens of standards; some apply to specific industries, and some apply to specific security technologies. Baseline functions and compatibility standards cannot adequately protect all businesses and all Web services. In addition, malicious threats emerge and quickly mutate, potentially mitigating the effectiveness of security measures to protect against a threat. Companies of all sizes expend significant effort in creating services, so it is important that they choose the standards that best support their needs. The answers to these questions can help you begin:

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

The standards described in Table 1 are commonly used in today's Web services to facilitate rapid deployment and interoperability. Organizations deploying Web services should incorporate standards, ranging from HTTPS through XML Signature standards, yet keep their security plans open enough to accommodate the future addition of Security Assertion Markup Language (SAML) and Web Services Trust (WS-Trust) standards.

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 by partners and customers who connect to it. The most widely deployed standard, Secure Sockets Layer (SSL), is a basic security building block. Early Web services were secured only with two-way SSL. Today, supporting only SSL in Web services significantly limits the service's long-term function and overall enterprise security.

Table 1. Well-Established and Emerging Standards for Web Services

Standard	Standard Description	Adoption	
		Today	Trend
HTTPS	An HTTP connection secured between the client and host using SSL and Transport Layer Security (SSL/TLS), a secure pipe that helps ensure the confidentiality of the information transmitted over the public Internet	Very high	Stable
XML	A text markup language for interchange of structured data	Very high	Stable
XML Schema	A language for describing the structure and constraining the contents of XML documents	High	Growing
Simple Object Access Protocol (SOAP)	A standard that defines application-level structure for messages	Moderate	Growing
Web Services Definition Language (WSDL)	Effectively the URL for a specific Web service; expressed in XML, a WSDL definition describes how to access a Web service and what operations it will perform	Moderate	Growing
Web Services Security (WS-Security)	A mechanism for incorporating security information into SOAP messages	Moderate	Growing
XML Encryption	A process for encrypting and decrypting parts of XML documents; a subset of the standard is used by WS-Security to maximize interoperability	Moderate	Growing
XML Signature	A mechanism for validating the origin and integrity of XML documents; a subset is used by WS-Security to maximize interoperability	High	Growing
SAML	A framework for exchanging authentication and authorization information	Moderate	Growing
WS-Trust	A standard for creating networks of federated trust	Low	Growing

Which Standard Fits Your Needs?

Today, SSL secures HTTP connections (HTTPS) and information in transit. This approach is important, but not enough. Relying only on HTTPS creates three problems:

- The Web service must undertake considerable private key and certificate management.
- Message confidentiality and integrity cannot be guaranteed.
- No auditable record of the message, session, or security is 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 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 messages. Applications using this feature are most efficiently deployed on dedicated infrastructure to optimize performance.

The WS-Security specification provides a way to help ensure that messages remain confidential, have not been tampered with, and are actually from senders asserting to have sent them. WS-Security specifies the use of XML 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.

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 is 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 works both ways, however. Widespread use of XML and Web services makes it significantly easier for outside, uninvited parties to integrate systems and invade applications. The results can range from annoying service glitches, to privacy breaches, 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 become more connected to each other over the Internet, the number and severity of attacks rises.

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) attacks, which are new, application-level versions of network-level DoS attacks

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

Defending Against Identity Attacks

Traditional identity threats (Table 2) 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 system 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 2. Traditional Internet Threats Relevant for Web Services

Attack	Countermeasure
• Request authentication attack: An attacker pretends to be a particular authorized user so the service will grant the attacker 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. • Response authentication attack: An attacker can also pose as the service, rather than as the user. For instance, if a legitimate user sends a request to a valid service, but an attacker is eavesdropping, the attacker can then pose as the legitimate service and can request confidential information or payments. Phishing is a variant of this type of attack.	• 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 help ensure that each party is who it claims to be.
• Authorization attack: An authenticated user obtains access that he or she 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, enter 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 work.	• A service that controls access to many different resources should implement a well-designed authorization strategy to help ensure that each authenticated user has access to just the appropriate resources and no others.
• Confidentiality attack: An attacker eavesdrops on a transmission and obtains a copy of the authorization. The attacker then has copies of any confidential information in the authorization: Social Security numbers, account numbers, addresses and phone numbers, private health and medical information, and so on. Confidentiality threats are serious matters; they can result in identity theft, embezzlement, fraud, leakage of trade secrets, and many other serious problems.	• 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. An attacker who succeeds in intercepting an encrypted message gains 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.
• Data integrity attack: If anyone involved in the process of generating, transmitting, or receiving data alters it improperly, the transaction can be fraudulent and dangerous. It might be altered to order the wrong product, or to send it to the wrong address, or to bill the wrong party. Attacks that rely on altered or malicious data are called data integrity threats. There are many ways to launch a data integrity attack: for instance, an attacker may forge a message or intercept and change a legitimate one. For example, the Code Red worm relied on data that was simply too big for the target servers to handle properly.	• The simplest, most effective technique for protecting data integrity is the use of cryptographic tools to protect the data channels and the contents of messages, as explained in the discussion of confidentiality. Cryptographic tools also provide techniques such as digital signatures, which can help guarantee that a message cannot be altered without the receiver's knowing about it. Content-analysis tools can also use technologies such as document type definitions (DTDs) and XML Schemas to analyze the contents of messages to determine whether they meet certain requirements.
• Replay attack: An attacker improperly and continuously resends a legitimate, intercepted or copied request to a service. For example, an attacker who managed to capture a valid purchase order could repeat the order over and over, essentially vandalizing a company's sales process.	• The straightforward way to protect against replay attacks is to attach a serial number or identifier to each message and compare each new incoming message to help ensure that no message is used more than once.

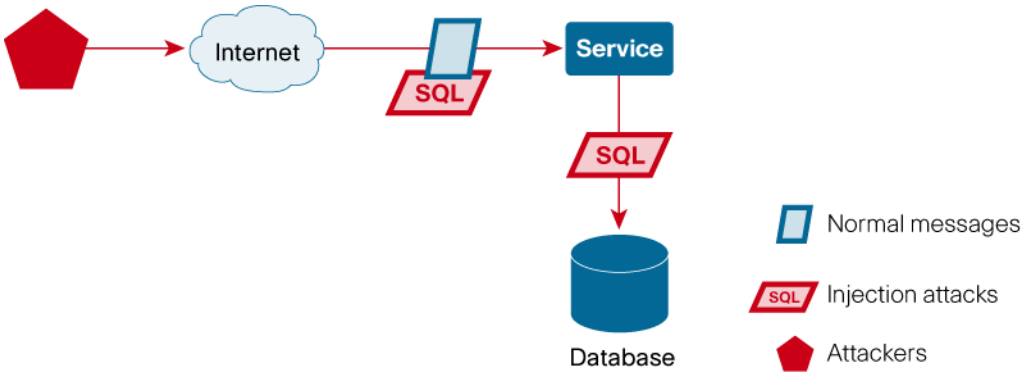
Defending Against Content-Borne Attacks

An excellent feature of the Web is its use of 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 FTP, are guarded more closely. However, 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.

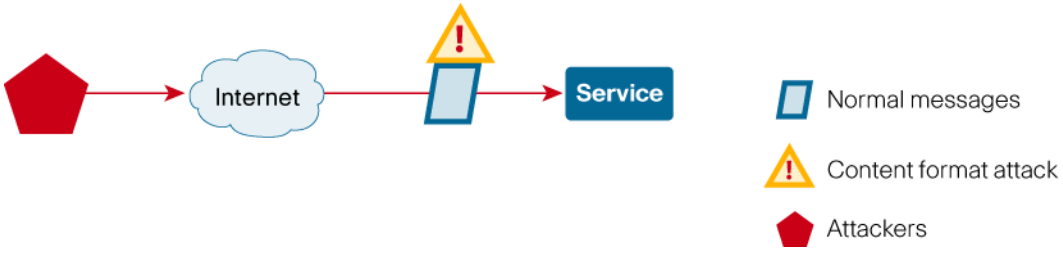
Two examples of content-borne XML exploits are Sequential Query Language (SQL) injection attacks and buffer overflow attacks. SQL injection is the practice of inserting malicious SQL statements into XML to disrupt back-end systems. If a Web service connected to a database does not validate SQL, an incoming XML message containing rogue SQL statements could break out of the expected database query and be used to obtain unauthorized information or destroy data (Figure 4). In fact, SQL injection attacks are a subset of a broader class of attacks known as command injection attacks. As when they use malicious SQL code to attack databases, hackers attempt to tunnel UNIX commands inside XML to exploit any system that has a command-oriented interface.

Figure 4. SQL Injection Attack



Like SQL and command injection attacks, a buffer overflow attack is aimed at the service endpoint and preys on vulnerabilities there, such as a buffer without enough memory set aside to handle a large variety of inputs: for example, a Web service designed to take in phone numbers.

Figure 5. Content Format



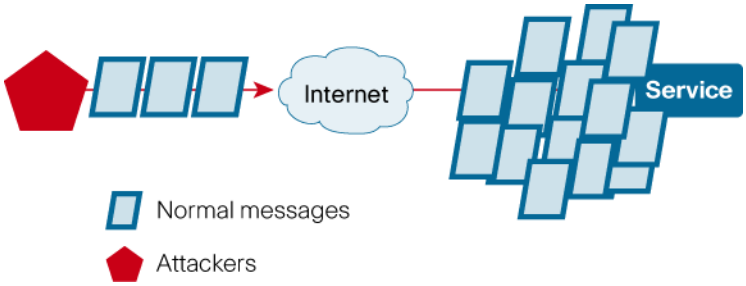
Another example of an XML virus or content-borne attack is a content format attack that exploits vulnerabilities in the way that services read content formats (document types, element names, attribute names, etc.) before they examine the actual content (Figure 5). Web services integration relies on standards 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 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 insertion of extremely large element or attribute names into an XML document in an attempt to overload a parser's resources.

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 the use of schema validation in conjunction with a second, more sophisticated pattern matcher that detects suspicious patterns such as SQL statements and commands. Services should process only content that successfully passes through both validation steps.

Defending Against XDoS Attacks

The third type of XML and Web services attacks are XDoS attacks (Figure 6). These attacks tend to make services unusable for everyone. These attacks are difficult to distinguish from legitimate traffic, making selectively servicing only legitimate requests difficult. New XDoS attacks have similarly thorny issues. Defending against XDoS attacks requires detection of an attack based on a combination of metrics that signify an attack, not just one metric viewed in isolation.

Figure 6. XDoS Attack



One of the first widespread XDoS attacks was the entity expansion attack, where unprivileged users used 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 XDoS attacks can occur as the result of simple human error, such as a programmer's mistakenly sending 100 requests per second instead of 10 or accidentally coding an infinite loop.

XDoS and certain authentication attacks can be detected only with configurable heuristics. For example, there may be from three to eight indicators that XML traffic is actually an XDoS attack. These signals are not generated only 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 help ensure that abnormalities are noticed and handled. A sophisticated approach uses a graduated response to handling abnormalities, with actions ranging from alerts, to throttling, and finally to 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 are 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 these factors seriously impede your ability to quickly provision new partners and revenue-generating services.

Instead, look for solutions that do the following:

- Let you securely connect Web services with internal or external business partners quickly, reliably, transparently, and manageably
- Enable centrally defined coarse and fine-grained security policies (different users in different groups can specify a scalable Web services security solution, and it will employ intelligent policy coordination for consistent enforcement)
- Optimize the processes that your Web services team has to do all the time: create and provision services and connections; create, approve, and record policies; migrate services and policies between environments; and transactionally deploy policy
- Enable any-to-any integration for platform, protocol, and standards mediation with a deny-by-default architecture that helps ensure that only trusted messages reach your services, an approach that provides highly reliable security and extends the longevity of your Web services architecture while reducing testing time in heterogeneous environments
- 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
- Provide comprehensive, flexible support for failover, load balancing, and capacity planning

The Importance of Logging

Many of the problems 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, after an attacker has defeated an authentication or authorization scheme and gained access to sensitive resources, the attacker can exploit those resources repeatedly. Similarly, after an attacker discovers how to create a forged message that gets effective results, the attacker can send it over and over. In addition, the task of debugging is much more complicated with encrypted messages, which need to be considered as part of the troubleshooting process. Finally, in this era of scrutiny and compliance, a secure record of the security enforced, the policy enforced, and the messages themselves is crucial to compliance, and all these functions must be 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 XDoS attacks because the only reliable way to identify the threat is to detect a sudden increase in the 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, lower the cost of business

communication, and 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.

Cisco® provides the critical XML infrastructure products used by enterprises to realize the promise of Web services. The Cisco ACE Application Control Engine Extensible Markup Language (XML) Gateway enables businesses to secure, implement, and operate XML Web services more efficiently and effectively, accelerating time-to-market for their products and gaining competitive advantages in their businesses. For more information about the Cisco ACE XML Gateway, visit http://www.cisco.com/go/ace.

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Asia Pacific Headquarters
Cisco Systems, Inc.
168 Robinson Road
#28-01 Capital Tower
Singapore 068912
www.cisco.com
Tel: +65 6317 7777
Fax: +65 6317 7799

Europe Headquarters
Cisco Systems International BV
Haarlerbergpark
Haarlerbergweg 13-19
1101 CH Amsterdam
The Netherlands
www-europe.cisco.com
Tel: +31 0 800 020 0791
Fax: +31 0 20 357 1100

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at www.cisco.com/go/offices.

Printed in USA C11-410359-00 6/07

