Doc. ID: ZTF-WS113-1
Released: June 20, 2002

zapthink
focus report

# XML Security Technology Landscape

ZapThink, LLC • 11 Willow Street, Suite 200 • Waltham, MA 02453 • info@zapthink.com • www.zapthink.com

# XML SECURITY TECHNOLOGY LANDSCAPE

JUNE 20, 2002

*Analyst: Jason Bloomberg*

**Abstract**

Security is the immediate roadblock facing widespread implementation of Web Services technologies across the enterprise. As a result, many software vendors are throwing their hat into the XML and Web Services security ring, offering a broad and confusing number of solutions to a variety of real and perceived problems. However, much of this effort amounts to jostling for defensible market positioning ahead of a solid demand for enterprise-class XML and Web Security products and services. As a result, ZapThink believes that the emerging market for XML and Web Services security solutions will be characterized by a period of turbulence, as companies struggle to clarify their messages and shake the kinks out of their product offerings.

## *Key Points:*

### ✦ Market Overview

- Security is the major roadblock to Web Services adoption.
- There is tremendous confusion over implementing secure Web Services due to conflicting, overlapping, and incomplete security offerings.

### ✦ Facts & Figures

- The XML and Web Services security market will reach $4.4 billion in 2006, growing over 300% annually.

### ✦ Analysis

- The lack of robust security and manageability solutions inhibits the ability for companies to effectively utilize Web Services for integration with business partners.
- Emergent startups have remarkably robust XML/WS security solutions due to adequate funding, solid business models, seasoned management teams, and high quality engineering staff.
- Companies that have deep technical knowledge of application level security coupled with a solid customer base will be best poised for success in the XML and Web Services security space.

### ✦ Future Trends

- Demand for XML/WS security solutions will spike within the next 12 months.
- Web Services will play limited role in transactional environments until 2003.
- By 2006, most security products from existing vendors will support or provide XML and/or Web Services security.

### ✦ Decision Points

- Securing a company's Web Services outside of the context of an overall security strategy provides a false sense of security.
- Enterprises must institute policies that apply to their entire extended enterprise and administer that security in a hierarchical fashion.
- Next-generation firewalls must be capable of looking at and securing the content of XML streams .

All Contents Copyright © 2002 ZapThink, LLC. All rights reserved. Reproduction of this publication in any form without prior written permission is forbidden. The information contained herein has been obtained from sources believed to be reliable. ZapThink disclaims all warranties as to the accuracy, completeness or adequacy of such information. ZapThink shall have no liability for errors, omissions or inadequacies in the information contained herein or for interpretations thereof. The reader assumes sole responsibility for the selection of these materials to achieve its intended results. The opinions expressed herein are subject to change without notice. All trademarks, service marks, and trade names are trademarked by their respective owners and ZapThink makes no claims to these names.

# Table of Contents



Copyright © 2002, ZapThink, LLC 

Copyright © 2002, ZapThink, LLC 

## I. Report Scope

*Locking 19 entrances to a building does not provide security if there are 20 entrances.*

Fundamentally, this report covers the overlap of two large areas of general interest: IT security, and the combination of XML and Web Services. Therefore, the scope of this report must be put into the context of each of these areas. IT security solutions address how to mitigate risks in the enterprise at all levels. Locking 19 entrances to a building does not provide security if there are 20 entrances, and the same is true for IT. Therefore, discussions of security must cover all aspects of security, including human, physical, network, and application security.

Within the XML and Web Services world there are two main challenges: how to secure Web Services (and in general, any kind of XML message), and how XML and Web Services-based solutions can provide security to the enterprise. Each challenge must also be placed within the overall context of IT security; after all, a hacker doesn't care if a particular vulnerability is Web Service related.

This ZapThink report, *XML and Web Services Security*, covers the various products and services on the market today that focus on meeting the requirements for providing security to enterprises that use Web Services, as well as products and services that use Web Services to provide security to enterprises. The report identifies benefits, challenges, key market drivers, and determines the sizing and growth of the market for these products and services.

This report covers:

- The context for XML and Web Services security within both the Web Services model for distributed computing and the IT security landscape
- The technology landscape for XML and Web Services security, including:
  - The layers of IT security
  - Precursors to XML and Web Services security, including PKI, Kerberos, and SSL
  - Key ongoing efforts in XML security, including XML signatures, XML encryption, and XKMS
  - Key ongoing efforts in Web Services security, including SAML and WS-Security
- Segmentation of the XML and Web Services security market
- The current state of the XML and Web Services security market, including the advantages and disadvantages of various approaches to security, and the return on investment (ROI) for implementation of the various products and services in this space
- Business and technology trends in the XML and Web Services security market, including:
  - ZapThink's predictions for market growth
  - inhibitors to growth
  - Market sizing and consolidation
- Profiles of vendors offering XML and Web Services security solutions.

This report does not cover:

- Global identity services, namely the Liberty Alliance and Microsoft Passport and TrustBridge. These will be covered in ZapThink's upcoming *Global Identity Services* report.
- Digital Rights Management products and services.
- Directory services (LDAP, Active Directory, etc.)
- Web Service infrastructure and management vendors who do not have a specific security offering. Web Service infrastructure and management

Copyright © 2002, ZapThink, LLC 

will be covered in the upcoming ZapThink Report *Web Services Infrastructure and Management*.

ZapThink hopes that this report will provide a fundamental understanding of the security issues surrounding XML and Web Services, and will provide a clear picture of the current state as well as the future of this dynamic, emerging market.

## II. Context for Security in the Web Services Model

ZapThink believes that Web Services are an evolutionary step in the development of distributed computing characterized by open standards, loose coupling, and dynamic description and discovery. (For our complete definition of Web Services, please see the ZapThink report *The Pros & Cons of Web Services*. For a broad look at the Web Services market, refer to the ZapThink report *Web Services Technologies & Trends*). Nevertheless, to say that Web Services are evolutionary rather than revolutionary is not meant to diminish their importance. It is important to point out, however, that the *promise* of Web Services is still quite different from today's reality. Therefore, it is a goal of this report—and truthfully, part of ZapThink's mission—to provide our subscribers with a clear roadmap, that places today's implementations of Web Services in context, and then provides a rational picture of where we feel Web Services will take distributed computing and the whole of IT in the future.

### 2.1 The ZapThink Web Services Roadmap

To understand the context of security in the discussion of Web Services, it is important to understand how Web Services will affect distributed computing

**TAKE CREDIT FOR READING ZAPTHINK RESEARCH!**


ZapThink is an IT market intelligence firm that provides trusted advice and critical insight into XML, Web Services, and Service Orientation. We provide our target audience of IT vendors, service providers and end-users a clear roadmap for standards-based, loosely coupled distributed computing – a vision of IT meeting the needs of the agile business.

This document provides just a small glimpse of the intelligence ZapThink offers. To get the full picture, please visit our Web site at www.zapthink.com. You'll find information about the range of our research on XML, Web Services, and SOAs and more of our market insight. You'll also be able to sign up for our popular biweekly ZapFlash newsletter that can deliver our market-leading intelligence directly to your inbox.

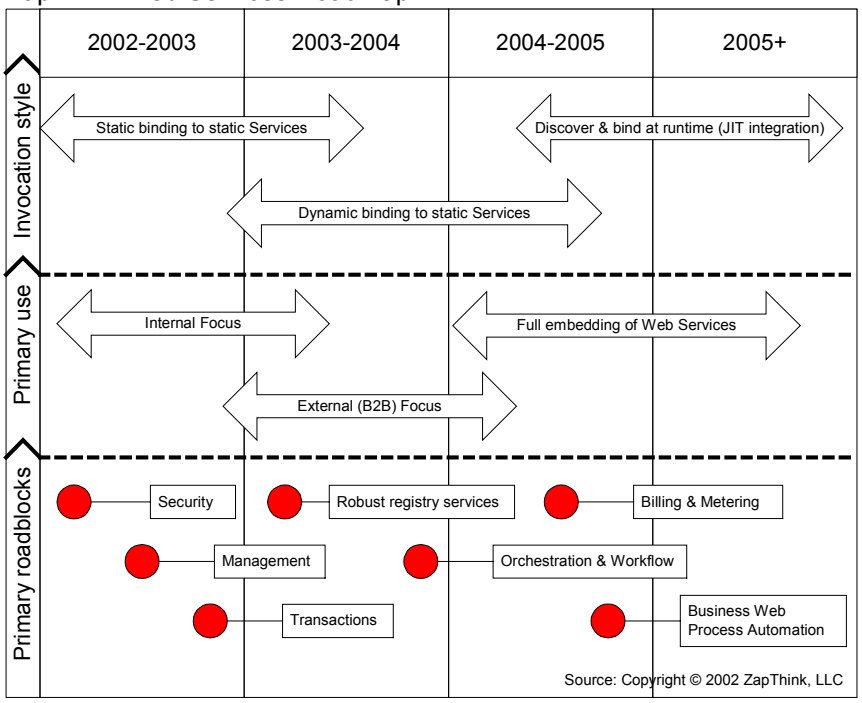
Also, Take Credit for reading ZapThink research! Visit www.zapthink.com/credit and enter the code SECLAND. We'll reward you with ZapCredits that you can use to obtain free research, ZapGear, and more! If you purchased this document, Taking Credit for it entitles you to free updates. If this document was free, then we'll notify you when updates are available if you Take Credit for it.

We hope that this document and our Web site help you understand the XML, Web Services, and Service Orientation marketplace better. However, our research is only a part of the value we offer our customers. For personal advice, press support, and competitive intelligence, subscribe to our ZapAccess research subscription service. Become a ZapThought Leader – let ZapThink help you understand the market-changing impact of standards-based, loosely coupled distributed computing, and use that understanding for competitive advantage.

For more information, please call us at +1-781-207-0203, or drop us an email at info@zapthink.com.

over the next five or more years, as shown in the ZapThink Web Services Roadmap in Figure 2.1.

**Figure 2.1: ZapThink Web Services Roadmap**




*The primary use for Web Services today is for internal integration.*

The primary use for Web Services today is for internal integration, where the invocation style is for Web Service consumers to bind to static Web Services at design time – basically, simply replacing existing, proprietary RPC protocols with open standards-based protocols. However, in the 2003-2004 timeframe, ZapThink sees more companies taking advantage of Web Services' dynamic binding capabilities. This invocation style will generally coincide with a greater use of Web Services outside the firewall between business partners. Finally, in the 2004-2005 timeframe, we believe that companies will be capable of discovering and binding to Web Services at runtime, in a mechanism known as "Just-in-time (JIT) integration". JIT integration enables enterprise software vendors to rearchitect their suites as loosely coupled collections of Web Services. At that time Web Services will be fully embedded into the software development process.

**Decision Point**

*The next roadblock on the path to Web Services adoption is security. Security is today's key enabler for Web Services.*

However, this vision of the future of Web Services is not without its roadblocks. In fact, there are seven major roadblocks that promise to inhibit or even derail progress as shown on the Web Services roadmap. Adoption of Web Services will progress up to the point that the next roadblock begins to impact the further acceptance of Web Services. Only when the relevant issues are resolved and the roadblock disappears will Web Services continue on its path toward widespread adoption.

### 2.2 Security: The Key Enabler for Web Services

When Web Services is used for static communication between two internal, controlled systems, security is not a major issue since existing security

Copyright © 2002, ZapThink, LLC 

techniques (firewalls, SSL, etc.) usually suffice to meet an organization's risk mitigation requirements. However, for any application of Web Services that provides access outside the corporate firewall, or intra-enterprise applications where the Web Services consumer is not fully controlled by the IT department, security issues are clearly the first and foremost issue that IT must resolve. Therefore, security is today's key enabler and roadblock for Web Services.

Of all the concerns on today's executives' plates, security is the most complex. Security is essentially the mitigation of risk, and risk is something every company has and wants to rid itself of. Risk, however, is a slippery concept, because it deals in possibilities: the possibility of a break-in, or a virus, or a loss of confidentiality, for example. Lowering risk means lowering the probability that such future events might occur; therefore, the ROI on security investments can only be expressed in terms of how much a security failure would have cost if it had happened, even though it hasn't. Such ROI calculations are particularly difficult, because the greatest risk is one that is completely unexpected—one that by definition wasn't predictable.

The other main reason that security is such a complex issue is the "twenty doors" problem. Simply securing one part of a system, or a network, or a building complex provides only false security. All possible risks must be considered, including those unpredictable ones that are potentially the most costly. Therefore, it is critical that this report be placed into the context of an overall security strategy. If a company increases its risk by not securing its Web Services, then there's no question that this door must be closed and locked. However, simply securing all of a company's Web Services alone only provides a false sense of security.

*This report must be placed into the context of an overall security strategy. Simply securing all of a company's Web Services alone only provides a false sense of security.*

Therefore, while this report necessarily must focus on securing XML and Web Services, all of the discussions herein must be considered within the context of the greater security landscape. The next five sections place the discussion of securing XML and Web Services in the greater context of IT security and related markets, in order to provide readers with the "big picture" of IT security as it pertains to XML and Web Services.

### 2.3 Context: Security Products & Services

From the broadest perspective, security encapsulates three major categories: human, physical, and IT. The human side of security involves hiring policies, training, password and physical key policies, and various cultural issues. Physical security includes lock and key systems, surveillance technologies, alarms, and other such protection systems. IT security is the final category and is concerned with securing computer hardware and software, networks, devices, etc.

The IT security market can be segmented as follows:

- Hardware security solutions, divided into:
  - Firewall appliances
  - Other security appliances (secure routers, etc.)
  - Authentication hardware
- Professional services solutions for security.
  - Protection services
  - Security audit
  - Intrusion detection services
- Software security solutions, segmented into:
  - *Authentication, authorization, and administration* ("3A").

Copyright © 2002, ZapThink, LLC

- Encryption, a smaller market that often overlaps with 3A. For the purposes of this report, therefore, we will consider encryption to be part of the 3A IT security services segment.
- Software that runs on firewalls and other security devices.
- Anti-virus software.

*Most XML and Web Services security offerings on the market fall within the 3A (authentication, authorization, and administration) segment of the IT security market.*

Most XML and Web Services security offerings on the market fall within the 3A segment, but there are several important exceptions to this generalization. We will consider each of them:

- Many hardware firewall vendors are developing support for XML and Web Services. In addition, there are vendors such as **Sarvega** and **Forum Systems** who are developing new kinds of hardware network appliances in order to provide different levels of XML and Web Services security. ZapThink's upcoming report on *XML and Web Services Network Appliances* will provide detailed coverage of these segments, including both the firewall hardware and the software that runs on them.

- Anti-virus software does not protect XML or Web Services *per se*. However, major anti-virus software vendors, including **Symantec** and **McAfee**, have been offering their desktop anti-virus products via Web-based service interfaces for a while now. Both companies are considering moving their currently proprietary service infrastructures to the open Web Services standards, which would make them *bona fide* Web Services. However, providing anti-virus software via a Web Service is an example of an application of Web Services for security, rather than an example of securing XML and Web Services.

- IT security professional service providers like **TruSecure**, **ISS**, and divisions of **IBM** and **EDS** offer a range of professional services options to their customers, which increasingly include Web Services security professional services. For these companies, Web Services initiatives are just one more kind of risky project their customers might undertake. Each service provider is thus updating their procedures to include the proper advice, configuration management, intrusion detection, and auditing services to reflect the new protocols and programming techniques that Web Services introduce to the enterprise. Broadly speaking, therefore, IT security professional services are not being fundamentally changed by Web Services; instead, securing Web Services is just one more wrinkle in the complex set of protective services such companies offer their customers.

**Vendor Focus**

***Forum Systems***
***Sarvega***
***Symantec***
***McAfee***
***TruSecure***
***ISS***
***IBM***
***EDS***

*Most 3A products will be Web Service-enabled by 2005.*

So, while this report will touch upon the above segments, the core of the story about XML and Web Services security falls within the 3A security software segment. As we will show in section 6, the interesting story is not just how Web Services lead to new kinds of products within the 3A segment, but also how Web Services technologies will gradually propagate throughout the entire segment, where most 3A products will be Web Service-enabled by 2006.

**Vendor Focus**

***Bowstreet***
***Cape Clear***
***Iona***
***Systinet***
***AmberPoint***
***Flamenco Networks***
***Microsoft***
***IBM***
***HP***

### 2.4 Context: Web Services Management and Infrastructure Products

There are several startups and emerging companies fighting over the Web Services infrastructure space, including **Bowstreet**, **Cape Clear**, **Iona**, and **Systinet**, to name just a few. Each of these companies' product offerings includes security functionality in some form. In addition, there is now a burgeoning Web Services management segment, populated by the likes of **AmberPoint** and **Flamenco Networks.** Naturally, **Microsoft** also provides

Copyright © 2002, ZapThink, LLC

management capabilities within its .Net framework, and **IBM** is revamping its Tivoli system management product to provide robust Web Services management capabilities. It is also likely that **HP** is doing something similar with OpenView, although at this time HP's strategy is still up in the air due to their recent merger with Compaq. Managing the security aspects of Web Services are part of what each vendor in this category does, as well.

As shown in figure 2.1, Web Services management is the next major roadblock to Web Services adoption after Web Services security. It follows, therefore, that many companies looking to provide solutions in the Web Services management space must also resolve many of the issues surrounding Web Services security. For the purposes of this report, however, we will only be considering Web Services management and infrastructure vendors insofar as their XML and Web Services security products are important to the market. ZapThink covers the emerging Web Services Management market in its *Web Services Infrastructure and Management* report.

### 2.5 Context: Global Identity Services

Vendor Focus

***Microsoft***
***Sun Microsystems***

Another market segment closely associated with the XML and Web Services security space is the global identity services segment, populated most notably by **Microsoft**'s Passport and recently announced TrustBridge initiatives, and the **Sun Microsystems**-led Liberty Alliance. Each party in this brewing feud promises to offer both consumers and businesses a universally usable way of managing user identity and authentication mechanisms. However, since the Liberty alliance has yet to announce many aspects of their recommendations, Zapthink is publishing this report a few months too early to cover this segment in depth. Therefore, ZapThink will cover this area in its future report on *Global Identity Services*.

### 2.6 Context: Digital Rights Management Technologies

Vendor Focus

***ContentGuard***
***Microsoft***
***InterTrust***

The Digital Rights Management (DRM) space is also closely related to the XML and Web Services security space as well. Originally, DRM applied primarily to media assets (audio and video), but now increasingly applies to XML formatted content, as well. Vendors are using DRM technologies to apply usage metering and permission systems for the use of Web Services as well as the protection of the XML content from inappropriate usage or distribution. A primary source of overlap between the DRM and XML security areas results from the development of the Extensible Rights Markup Language (XrML), developed by **ContentGuard** and championed primarily by **Microsoft**.

*XrML-based DRM technologies will not form an independent market by themselves. Instead, the DRM space will eventually merge with the work in XML and Web Services security.*

XrML provides a universal method for specifying and managing rights and conditions associated with digital content as well as services, including Web Services. At this time, XrML is the primary rights language being used in working DRM solutions, and in particular, those from Microsoft. The reason that XrML overlaps XML and Web Services security is largely a byproduct of its complexity. In fact, XrML has been growing increasingly more complex as it has moved to the current version 2.0. For example, XrML 1.3 defined 18 different rights that can be conferred on content. XrML 2.0 expands this number to 24. The newly added rights include those that have to do with the issuance of digital signatures and the read/write/execute rights that are a key part of authorization mechanisms.

While Digital signatures are an important part of the XML security story, the question remains whether it is really necessary to define digital signatures as part of a DRM specification. The fact that XrML is overly complex, has relatively few supporters, and overlaps the more widely accepted efforts in the XML security space leads us to predict that XrML-based DRM technologies will not

Copyright © 2002, ZapThink, LLC 

form an independent market by themselves. It is more likely that with the assistance of DRM leaders like **InterTrust** and others, the DRM space will eventually merge with the ongoing work in XML and Web Services security.

### 2.7 Context: Directory Servers

**Vendor Focus**

***Novell***
***Sun Microsystems***
***Microsoft***

Ever since the unwieldy X.500 directory standard was supplanted by its simpler and more flexible cousin LDAP, the directory server market has taken off. Leaders in the LDAP directory server space include **Novell** and **Sun Microsystems**, while **Microsoft** provides an alternative through its more feature-rich Active Directory. Directory servers appear at the heart of most enterprises, managing hierarchical repositories of information on people and physical resources. LDAP is also capable of working with security constructs like digital signatures and certificates, making directory servers the favored user access management tool across enterprises worldwide.

Naturally, most of the products that this report covers are capable of accessing LDAP and/or Active Directory servers, either as a core part of their solution, or as one of the supported interfaces. The directory server story, therefore, is an important part of the XML and Web Services security story. Nevertheless, the directory server market is by no means XML-specific, and is generally considered to be a market in its own right.

## III. Technology Landscape

To understand the XML and Web Services security market, it is important to have a high-level understanding of software security, as well as the precursor technologies that provide the basis for XML and Web Services security. It is also important to understand what makes XML and Web Services security different than typical Web or network security.

There are two basic types of IT security:

- *Perimeter network security* focuses on which machines are on what networks, what protocols will be allowed to pass across these networks, and which TCP/IP ports on each device will allow traffic from particular sources.
- *Application level security*, on the other hand, focuses on which services users can request, based upon what roles those users have in the system. Application security also provides for the confidentiality and integrity of the transmitted data, both in transit as well as in storage.

*Application level security focuses on which services users can request, based upon what roles those users have in the system, and also provides for the confidentiality and integrity of the transmitted data.*

XML and Web Services security is primarily concerned with application level security, because Web Services send messages through firewalls typically using TCP/IP ports 80 (standard Web traffic) and 443 (SSL-secured Web traffic), thereby bypassing typical firewall restrictions. Earlier RPC (remote procedure call) technologies like CORBA and Java RMI communicate on different ports, which ironically has limited the usefulness of those protocols. Public-key encryption rollouts have also been affected by this problem, because firewalls typically block LDAP directory lookups, which use port 389.

**Decision Point**

*Next-generation firewalls must be capable of looking at the content of XML streams, and the security mechanisms for such data must be part of that content.*

Therefore, XML and Web Services circumvent the security offered by firewalls that simply deny network traffic based on particular TCP/IP port numbers. Next-generation firewalls must be capable of looking at the content of XML streams, and the security mechanisms for such data must be part of that content. The security mechanism must be sophisticated enough, however, to verify the

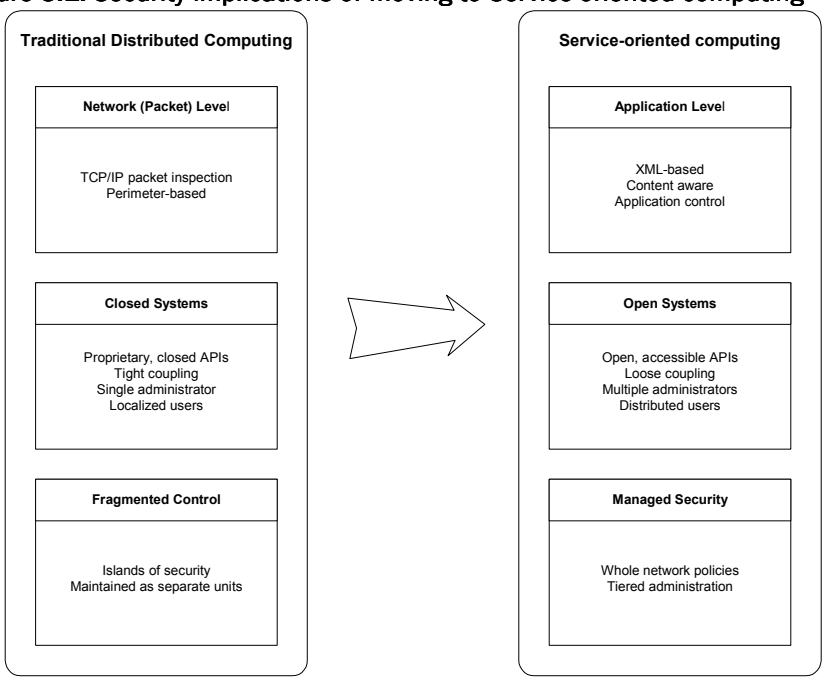
Copyright © 2002, ZapThink, LLC 

security of the message without compromising that security. This issue is fundamental to securing XML and Web Services.

### 3.1 XML security and the shift to Service-oriented computing

The shift from focusing on packet level network security to application level security that is aware of the contents of messages is one of the many changes facing an enterprise as it implements Web Services. As shown in figure 3.1, there are many levels of change facing IT in the enterprise, and each type of change has security implications associated with it. In addition to the need for firewalls to be application and content aware, there are changes at the system level as closed, proprietary systems give way to open, loosely coupled systems. Closed systems are relatively straightforward to secure; an administrators only needs to set up the users and their privileges, and the work is mostly complete.

**Figure 3.1: Security implications of moving to Service-oriented computing**




Source: Copyright © 2002 ZapThink, LLC

**Decision Point**

*Enterprises must institute policies that apply to their entire enterprise network (including participants invited from outside), and administer that security in a hierarchical fashion.*

Securing open, loosely coupled systems requires a much more sophisticated security approach, involving multiple administrators that support distributed users. Different systems now have different policies and possibly different security mechanisms. As a result, administrators must manage security much more actively than was necessary in the closed model.

Traditional distributed computing security was modeled by *islands of security*, which describe systems and users on isolated networks or subnetworks. The network acted as an island, with its own perimeter security, but users within the network were considered to be trusted. This “trusted vs. untrusted” dichotomy breaks down in a Service-oriented model, because users can access Services located on systems across one or more enterprises. The concept of trusted groups no longer has meaning; instead, enterprises must institute policies that apply to their entire enterprise network (including participants invited from outside), and administer that security in a tiered, or hierarchical fashion.

Copyright © 2002, ZapThink, LLC 

Departments or other organizational groups may then have their own administrators, but those administrators may in turn be administered by a more senior admin at a higher level within the enterprise.

### 3.2 Principles of Application Security

In order to fully understand how security can be provided and managed in the Service-oriented enterprise, it is important to first understand the principles of application level security.

TCP/IP forms the basis for all Internet-based communications. It allows systems to send information from one computer to another through a variety of intermediate computers and separate networks before it reaches its destination. TCP/IP's great flexibility has led to its worldwide acceptance as the basic Internet and intranet communications protocol, but the fact that it allows information to pass through intermediate computers makes it possible for a third party to interfere with communications. Securing these points of attack form the basis for application level security in a TCP/IP environment:

- *Eavesdropping* -- Information remains intact, but an unauthorized person compromises its privacy. For example, someone could learn a credit card number, record a sensitive conversation, or intercept classified information.

- *Tampering* – An unauthorized person changes or replaces information in transit to the recipient. For example, someone could alter an order for goods or change a person's resume.

- *Impersonation* -- Information goes to a person who poses as the intended recipient. Impersonation can take two forms:

    - *Spoofing* -- A person can pretend to be someone else. For example, a person can pretend to have someone else's email address, or a computer can identify itself as a Web site it is not.

    - *Misrepresentation* -- A person or organization can misrepresent itself. For example, a site might pretend to be a bookstore when it is really just a site that takes credit-card payments but never sends any goods.

These potential security breaches lead to the following requirements for application level security.

#### *3.2.1 Application level security requirements*

Application level security contains five basic requirements, expressed in terms of the messages sent between parties. Such messages include any kind of communication between the sender (party who wishes to access an application) and the recipient (the application itself). The five requirements for application level security are:

- *Authentication*, The recipient of the message must be able to confirm the identity of the sender of the message.
- *Authorization*. The sender of a message must be authorized to send the message.
- *Confidentiality*. The contents of messages must not be available to unauthorized parties.

Copyright © 2002, ZapThink, LLC 

- *Data integrity*. The recipient of a message must be able to guarantee that a message hasn't been tampered with in transit.
- *Nonrepudiation*. The sender must be able to guarantee that the recipient received the message, including the time the message was sent and the fact the recipient received only a single copy.

This report will take each of these five requirements in turn, and explain the issues involved in satisfying each requirement within an XML and Web Services security context.

*3.2.2 Authentication*

A system provides authentication if it allows access only to those users who can provide the proper identity credentials. Systems commonly handle authentication with a user ID and password as the identity credentials. However, user ID/password security offers only a relatively low level of protection. Other authentication schemes use software tokens as identity credentials. PKI, for example, uses a digital certificate as the identity credential, and Kerberos uses a ticket. Only the person who owns a security token, whether it's a certificate or a ticket, can use the token.

One of the biggest issues with authentication is the bootstrap problem: to obtain a security token a user must prove their identity to the issuer of such tokens, but how can they do that without a token? The answer is that there must be an identification mechanism outside of the software system that provides the initial confirmation of identity—a user can only get their first security token by providing a driver's license to the token issuer, for example. Once the user obtains one security token, then it is possible to use that token to provide authentication to multiple systems.

*There must be an identification mechanism outside of the software system that provides the initial confirmation of identity.*

*3.2.3 Authorization and Access Control*

Authorization determines whether a user is allowed to perform the functions it requests or access requested data. Authorization is particularly important because of the need for tiered security administration in Service-oriented environments, where security administrators delegate their administrative functions to other administrators. At the simplest level, systems handle authorization with access control lists (ACLs) that list which users are entitled to perform certain operations (e.g., read, write, delete) on particular resources. However, ACLs are generally insufficient to handle the real-world security policies required at many enterprises, because Web Services provide programmatic interfaces that are difficult to monitor for suspicious activity. Take, for example, an HR application that has a Web Service interface. A request for Mary's salary would raise immediate suspicion from a human HR representative, but access to an improperly protected SOAP interface to the HR system would be more difficult to detect.

This situation is even more complex when multiple, heterogeneous systems are involved, either within an enterprise or across two or more companies. Every company will likely have its own security policies, in addition to its own authorization technology. Therefore, the ability to provide and administer authorization across multiple systems is a difficult problem that many of the vendors covered in this report are trying to solve.

*The ability to provide and administer authorization across multiple systems is a difficult problem that many of the companies covered in this report are trying to solve.*

*3.2.4 Confidentiality*

Confidentiality means that an unauthorized person cannot view or interfere with a communication between two parties. Trust infrastructures like the *Public Key Infrastructure* (PKI) and Kerberos use encryption to ensure that messages are kept confidential. PKI in particular can use encryption to protect the

Copyright © 2002, ZapThink, LLC 

confidentiality of data both in transit and in storage. Virtual Private Networks (VPNs) and *Secure Sockets Layer* (SSL) can protect the confidentiality of messages between two endpoints, but neither secures the data in storage or across intermediaries, because both SSL and VPNs are point-to-point techniques. Therefore, an SSL-encrypted message, for example, would have to be unencrypted at an intermediary, which opens a security hole.

*Intermediaries are especially important in the context of Web Services, because the SOAP protocol is designed to support intermediaries that can forward or reroute SOAP messages.*

The problem of intermediaries is especially important in the context of XML and Web Services, because the SOAP protocol is designed to support one or more intermediaries that can forward or reroute SOAP messages based upon information either in the SOAP header or the HTTP header. Therefore, there must be a way for the intermediary to read the part of the message that tells it what to do, without compromising the confidential payload of the message. However, technologies such as SSL prevent the effective functioning of these intermediaries.

*3.2.5 Data Integrity*

Data integrity comprises two requirements: first, the data received must be the same as the data sent. In other words, the message did not change in transit, either by mistake or on purpose. The second requirement for data integrity is that at any time in the future, it is possible to prove whether different copies of the same document are in fact identical.

PKI, for example, uses a technique called a *message digest* or *one-way hashing* to ensure data integrity. Hashing is an algorithm that takes any message and calculates a much shorter string of characters, known as the message digest, in such a way that performing the same algorithm on the same message again always yields the same result. The subsequent chance of two different messages yielding the same message digest is astronomically small. If the message digest is the same when a message is sent and when it is received, and the digest itself was kept secure, then data integrity is guaranteed. Likewise, a message can be hashed at two different points in time to determine if it has been altered.

*3.2.6 Non-Repudiation*

When secure messages are sent, the recipient often requires that the sender can't repudiate the message, or claim that the message wasn't sent at particular date and time. Likewise, a sender would like to guarantee that a given message was received. The most common way to provide non-repudiation is through the use of digital signatures. With digital signature technology, senders can both provide evidence that a document is valid while simultaneously logging the message transactions into signed audit logs. Once an audit log has been signed it cannot be surreptitiously modified.

### 3.3 IT Security Fundamentals

It is also helpful and important to understand the fundamentals of the existing application level security technologies in place. Much of this technology is based on a security construct known as a "*key*". A key is a string of characters that acts as a code that can be used to encrypt messages, and then decrypt them at the other end. There are two basic approaches to key encryption: *symmetric* and *asymmetric* key encryption. Symmetric key techniques, including Kerberos and SSL, use a specially generated secret key that is used as a shared secret between two communicating parties. If both parties have the same key, the message is secure. PKI, however, uses asymmetric key encryption, where there is a *public key* and a *private key*. The private key is used to encrypt a message, and the public key decrypts the message. Therefore, private keys are secrets kept with their owners, while public keys are truly public.

Copyright © 2002, ZapThink, LLC

### 3.3.1 Encryption and Decryption

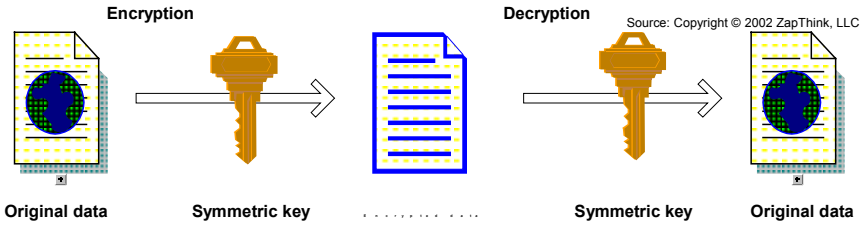
*Encryption* is the process of transforming information so it is unintelligible to anyone but the intended recipient. *Decryption* is the process of transforming encrypted information so that it is intelligible again. A *cryptographic algorithm* is a mathematical function used either for encryption or decryption. In most cases, systems use two related algorithms: one for encryption and the other for decryption.

With most modern cryptography, the ability to keep encrypted information secret is not based on the cryptographic algorithm itself, which is widely known, but on the key that a system must use in conjunction with the algorithm to produce an encrypted result or to decrypt previously encrypted information. Decryption with the correct key is a simple, quick process. Decryption without the correct key, however, should be impossible – in reality, requiring significant effort.

### 3.3.2 Symmetric-Key Encryption

The simplest form of key encryption is known as *symmetric-key encryption*. With this form of encryption, systems can calculate the encryption key from the decryption key and vice versa. With most symmetric algorithms, systems use the same key for both encryption and decryption, as shown in Figure 3.2.

**Figure 3.2: Symmetric-key encryption**




Symmetric-key encryption can be highly efficient and also provides a degree of authentication, since information encrypted with one symmetric key cannot be decrypted with any other symmetric key. As long as both parties keep the symmetric key secret, each party can be sure that it is communicating with the other.

Symmetric-key encryption is only effective, therefore, if both parties keep the symmetric key secret. If anyone else discovers the key, then the communication loses both confidentiality and authentication. A person with an unauthorized symmetric key not only can decrypt messages sent with that key, but also can encrypt new messages and send them, thus spoofing either party.

### 3.3.3 Public-Key Encryption

Public-key encryption is a form of *asymmetric* encryption, because it uses a pair of keys—a public key and a private key—for each entity that wants to authenticate its identity, sign data, or encrypt data. The entity publishes its public key, but keeps the corresponding private key secret. If a system uses a public key to encrypt data, that data can only be decrypted with the corresponding private key. Figure 3.3 illustrates public-key encryption.

**Figure 3.3: Public-key encryption**

Copyright © 2002, ZapThink, LLC 

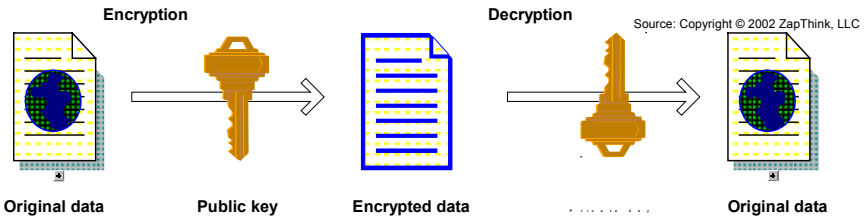



Figure 3.3 shows that a user can freely distribute a public key, but only that user will be able to read data encrypted using this key. In general, to send encrypted data to someone, the user encrypts the data with the intended recipient's public key, and the person receiving the encrypted data decrypts it with their corresponding private key. Compared with symmetric-key encryption, public-key encryption requires more computing effort. SSL solves this efficiency limitation by using public-key encryption to send a symmetric key, which it then uses to encrypt additional data.

The process illustrated in figure 3.3 works in reverse when the goal is to digitally sign data. If a user encrypts a message with their private key, then anyone can decrypt it, but recipients can confirm that the user who sent the message actually signed it. Such digital signing also confirms that an unauthorized party hasn't tampered with the message since it was signed.

*3.3.4 Digital Signatures*

Encryption and decryption address the problem of eavesdropping, but do not alone address the issues of tampering and impersonation. Tamper detection and related authentication techniques rely on message digests. A message digest has the following characteristics:

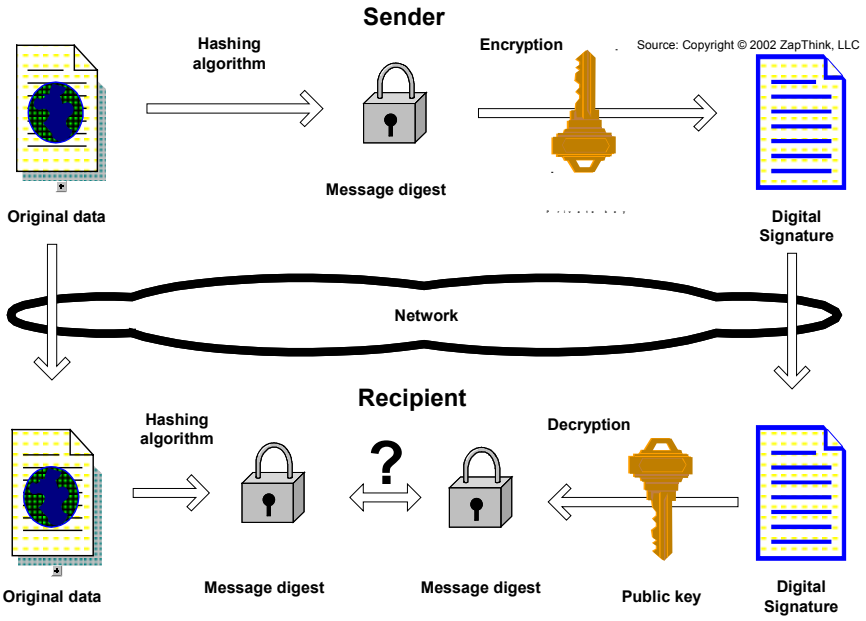
- The value of the digest is unique for the hashed data. Any change in the data, such as deleting or altering a single character, results in a different value.
- No one can deduce the content of the hashed data from the digest, which is why this technique is called "one-way."

As pointed out above, it is possible to use a private key for encryption and the corresponding public key for decryption in order to digitally sign data. In practice, however, signing software first creates a message digest of the data, and then uses the private key to encrypt the digest. The encrypted digest is called a *digital signature*.

Figure 3.4 illustrates how digital signatures validate the integrity of signed data.

Copyright © 2002, ZapThink, LLC 

**Figure 3.4: Using a digital signature to validate data integrity**




In the above diagram, the recipient receives the original data and the digital signature. To validate the integrity of the data, the receiving software first decrypts the digest with the signer's public key. It then uses the same hashing algorithm that generated the original digest to generate a new message digest of the same data, which it compares to the original digest. If the two digests match, the integrity of the message is confirmed, and the recipient can be sure that the public key the recipient has corresponds to the private key that the sender used to create the digital signature. Confirming the actual identity of the signer, however, requires the additional step of confirming that the public key really belongs to a particular person or other entity, which requires a certificate.

### *3.3.5 Digital certificates*

A *certificate* is an electronic document that can identify an individual, a server, a company, or any other entity. The certificate associates that identity with a public key. Like a driver's license, a certificate provides generally recognizable proof of a person's identity. Public-key cryptography uses certificates to address the problem of impersonation.

The issuing of certificates works pretty much the same way as driver's licenses and other familiar forms of identification. *Certificate authorities* (CAs) are entities (typically companies, but they could also be internal to enterprises) that validate identities and issue certificates. The certificate the CA issues binds a particular public key to the name of the entity the certificate identifies (such as the name of a person or a server). Certificates help prevent the use of fake public keys. Only the public key that the certificate certifies will work with the corresponding private key that the entity the certificate identifies possesses.

In addition to a public key, a certificate always includes the digital signature of the issuing CA, which allows the certificate to function as a "letter of introduction" for users who know and trust the CA but aren't familiar with the entity the certificate identifies.

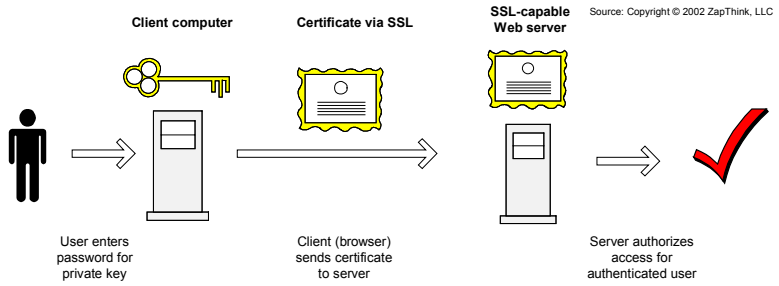
Copyright © 2002, ZapThink, LLC 

### *3.3.6 Authentication with certificates*

Digital certificates support several different types of authentication, including the ability to authenticate the client (for example, a person using a browser), or a server (for example, an SSL-secured Web site). In addition, certificates authenticate digital signatures. In addition to authentication, digital signatures ensure nonrepudiation: a digital signature makes it difficult for the signer to claim later not to have sent the signed message.

Figure 3.5 shows how client authentication works using certificates and SSL. To authenticate a user to a server, a client digitally signs a randomly generated piece of data and sends both the certificate and the signed data across the network. The server authenticates the user's identity based on the information in the digital signature. This example assumes that the user has already decided to trust the server and has requested a resource (typically a Web page), and that the server has requested client authentication in order to decide whether to grant access to the requested resource.

**Figure 3.5: Using a certificate to authenticate a client to a server**




*Certificate-based authentication is more secure than simple password-based authentication, because the user must have the private key as well as the password.*

Certificate-based authentication is more secure than simple password-based authentication, because the user must have the private key as well as the password. However, neither password-based authentication nor certificate-based authentication addresses the security issues inherent in physical access to individual machines or passwords.

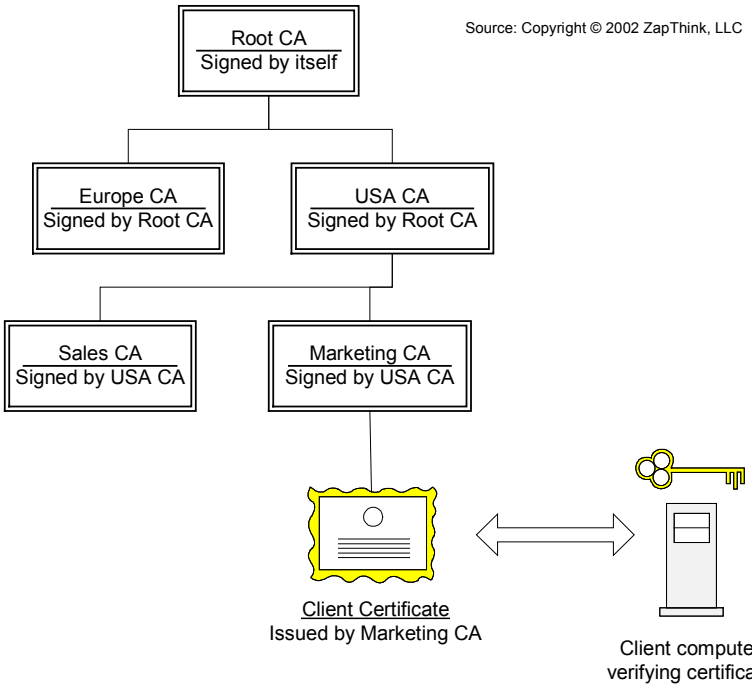
### *3.3.7 How CA Certificates Establish Trust*

*Certificate authorities* (CAs) validate identities and issue certificates. They can be either independent third parties or organizations running their own certificate-issuing server software. Any client or server software that supports certificates maintains one or more trusted CA certificates. These CA certificates determine which other certificates the software can validate—in other words, which certificate issuers the software can trust. In the simplest case, the software can validate only certificates issued by one of the CAs for which it has a certificate. Trusted CA certificates may also be part of a chain of CA certificates, each issued by the CA above it in a certificate hierarchy.

CAs can delegate certificate-issuing responsibilities to subordinate CAs. The ANSI X.509 standard (which is the standard for certificates) includes a model for setting up hierarchies of CAs. In this model, the root CA is at the top of the hierarchy. The root CA signs its own certificate, and then signs the certificates that belong to the CAs that are directly subordinate to the root CA, and the process cascades down the certificate chain. A *certificate chain* is series of certificates issued by successive CAs. Figure 3.6 shows a certificate chain leading from a root CA down through an enterprise's internal CA hierarchy.

Copyright © 2002, ZapThink, LLC 

**Figure 3.6: A certificate chain**




To verify that a certificate is valid and trustworthy, the verifying party climbs the certificate chain until it identifies a certificate it can trust. If the verifier trusts the issuer's certificate, the verification is successful. Otherwise, the verifier checks the issuer's certificate to make sure it is properly signed by the next CA up the chain. The verifier then checks that CA's certificate, and repeats the process as many times as necessary until it encounters a CA certificate it can trust.

*3.3.8 Managing Certificates*

*PKI includes the processes for issuing and storing certificates and keys, renewing and revoking certificates, and procedures for delegating the certificate issuing responsibility.*

The *Public Key Infrastructure* (PKI) refers to the set of standards and services that facilitate the use of public-key cryptography and X.509 certificates in a networked environment. PKI includes the processes for issuing and storing certificates and keys, renewing and revoking certificates, and procedures for delegating the certificate issuing responsibility.

All certificates are valid for a certain time period. Attempts to use a certificate before or after its validity period will fail. Therefore, CAs must be able to renew certificates. In addition, there is always the possibility that the private key used to create a certificate can become compromised, in which case the CA must be able to revoke the validity of that certificate. CAs typically handle certificate revocation by publishing a *certificate revocation list* (CRL), which is a list of revoked certificates, to a set location, and then checking the list as part of the authentication process.

*3.3.9 Kerberos*

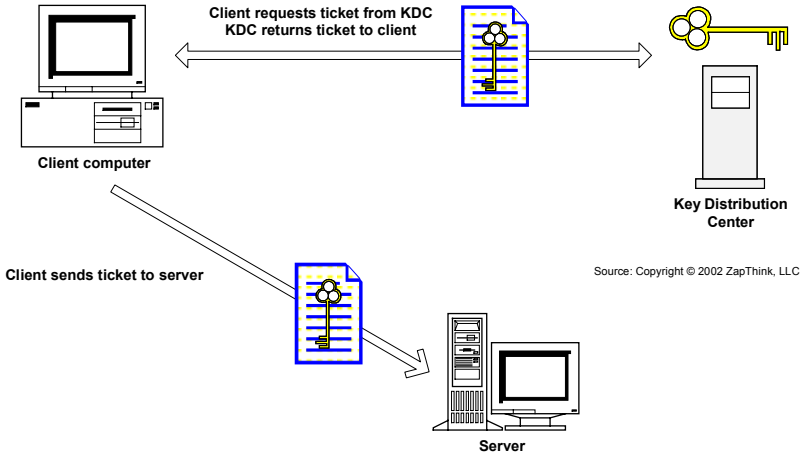
Like public-key encryption, *Kerberos* is an authentication protocol that identifies principals (including users and services) by requiring them to present proof of identity. Kerberos provides for mutual authentication: not only do users of a service identify themselves to the service, but users can challenge the service to prove its identity. To accomplish mutual authentication, Kerberos makes use of *trusted third-party authentication*. Trusted third-party authentication requires that the Kerberos server know who all the parties are as well as how they prove their identity (in particular, their passwords).

Copyright © 2002, ZapThink, LLC 

The concentration of secret information makes the Kerberos server (technically known as the *Key Distribution Center* or KDC) a very important resource. In particular, companies who use Kerberos must guarantee the physical security of the KDC, and the actual computers housing the KDC may actually be more vulnerable to unauthorized access than the network protocols that shield the KDC.

The Kerberos protocol assumes that all network traffic is vulnerable to capture, examination and substitution. In such an environment, authenticating (or "logging in") to Kerberos is a complex procedure, because the user cannot simply send the password directly to the server, or risk compromising the password. Kerberos solves this problem with encryption technology. The Kerberos client software uses the user's password to generate a symmetric encryption key. After a few exchanges with the server, the KDC returns information to the user, known as a *ticket*, that only software on the workstation that knows the temporary encryption key can use, as shown in figure 3.7. When users wish to contact a Kerberos-protected service, they first contact the Kerberos ticket-granting service and ask for a ticket to the service. A ticket is an encrypted message digest that proves the user's identity to the service.

**Figure 3.7: Kerberos authentication**




To increase the security Kerberos can provide, it makes use of temporary keys wherever possible. When a user and a service are interacting, they are doing so with a temporary key that the KDC specially generated just for this particular interaction. Such keys expire within a relatively short period of time. Furthermore, Kerberos also applies encryption technology to guarantee data integrity. Assuming that the client and service have authenticated as described above and each have the current temporary key, Kerberos is able to guarantee both confidentiality and data integrity following the digital signature process explained in section 3.3.4.

*3.3.10 Basic Security in HTTP*

HTTP addresses client authentication in a couple of ways. The HTTP Specification defines an authentication mechanism known as *basic authentication*. In basic authentication, a client passes its credentials (its name and password) to the HTTP service (typically a Web server) in the authentication header of the HTTP request. The HTTP service can authenticate the user or it can pass the credentials to another system for authentication. A second HTTP authentication mechanism, known as *digest authentication*, attempts to strengthen basic authentication by making the client compute a message digest containing the

Copyright © 2002, ZapThink, LLC 

client credentials. Both HTTP approaches have limited value, because neither digest authentication nor basic authentication provides for integrity, confidentiality, or nonrepudiation. For those reasons, browsers use SSL when the user requires more security than HTTP security can provide.

*3.3.11 Secure Sockets Layer (SSL)*

The *Secure Sockets Layer (SSL)* protocol governs server authentication, client authentication, and encrypted communication between servers and clients. SSL is the most popular security technology used on the Internet, especially for e-Commerce and other interactions that involve exchanging confidential information between a browser and a Web server.

SSL requires a server SSL certificate, at a minimum. As part of the initial handshake process, the server presents its certificate to the client, which the client uses to authenticate the server's identity. The authentication process uses PKI and digital signatures to confirm that the server is in fact the server it claims to be. Once the client has authenticated the server, the client and server use symmetric-key encryption to encrypt all the information they exchange for the remainder of the session, as well as to detect any third-party tampering that may occur during the session.

Servers may also require client authentication as well. In this case, after the client has authenticated the server, the client presents its certificate to the server for it to use to authenticate the client's identity before it establishes the encrypted SSL session.

**Decision Point**

*There are two main limitations to SSL. First, SSL does not provide audit trails, and second, SSL is point-to-point.*

There are two main limitations to SSL. First, SSL does not provide audit trails. It is difficult to prove that an SSL session occurred, or even to prove that a recipient received a piece of data sent via SSL. Therefore, users who wish to provide for non-repudiation typically use digital signatures in addition to SSL. The second problem with SSL is that it is point-to-point, meaning that security is guaranteed directly between the two communicating parties. For communications between a browser and a Web server, this limitation is not an issue, but in the more general case where messages can hit several intermediaries between the sender and the recipient, SSL does not provide end-to-end security.

### 3.4 XML Security Efforts

*The principles of application level security, key encryption, digital signatures, PKI, SSL, and Kerberos form the basis for all work in XML security.*

The principles of application level security, key encryption, digital signatures, PKI, SSL, and Kerberos form the basis for all work in XML security. XML presents two questions within this existing framework: how should systems secure XML messages, and how can the extensible and open principles of XML apply to the existing security technologies? The two essential XML security efforts that address these issues are XML Signature and XML Encryption.

*3.4.1 XML Signature*

The *XML Digital Signature (xml-dsig)* standard is a joint initiative of the IETF (Internet Engineering Task Force) and the W3C (World Wide Web Consortium). XML Signature describes a set of XML elements and attributes that store information about the hashing and encryption algorithms that software uses to generate digital signatures, as well as the signatures themselves. In addition, the public key that systems use to verify the digital signature either appears within the XML signature itself, or alternatively the XML signature can include the URL of the public key directory that includes the public key.

Copyright © 2002, ZapThink, LLC 

*XML Signature brings the well-known benefits of XML to digital signatures, making them human-readable, easily parsed, platform independent, and simpler to implement.*

While XML Signature defines how to render digital signatures in XML, it's not the case that XML Signature is solely intended for digitally signing XML documents. In fact, it is for signing any kind of digital content. XML Signature brings the well-known benefits of XML to digital signatures, making them human-readable, easily parsed, platform independent, and generally simpler to implement than earlier digital signature standards. In addition, XML Signature allows for one signature to reference multiple documents, and mandates that software must also sign information about the relevant encryption algorithms.

On the surface, XML Signature seems like a relatively simple extension of the existing work on digital signatures. However, bringing XML into the fray has led to some subtle, but complex issues. For example, the xml-dsig standard mandates that systems should sign only what the user sees. If a system uses a stylesheet to render the XML on a screen, then the server software must sign the visual representation of the data, since this is what the user actually sees. In addition, the xml-dsig standard specifies that the data must be signed along with whatever filters, stylesheets, client profiles or other information that might affect its presentation.

Another tricky problem with XML Signatures is that parties to a message must protect the *document itself*—not just the contents of the XML tags—so that no changes happen to it in transit that could invalidate the signature. The problem with this requirement has to do with the one-way hashing algorithms that systems use to produce the message digests that form a critical part of a digital signature. An XML document typically contains some white space between tags, for example, and this white space may change when a DOM or SAX processes the XML. Furthermore, the order in which tags or attributes occur in an XML document may change when it's loaded into a DOM or SAX processor. Therefore, when the application computes the message digest, then the hash will not match the original hash (because the white space or tag order is different) and so the signature will not be valid. To solve these problems, there are ongoing efforts with specifications known as *XML Canonicalization*. XML Canonicalization defines a standard way to normalize XML information among applications and operating systems. The resulting canonical XML should be application and platform-neutral, so that white space, tag order, and other minor differences caused by different XML parsing techniques disappear.

### *3.4.2 XML Encryption*

XML Encryption is currently a proposal that is working its way through the W3C. In addition to being a means to encrypt XML, XML Encryption also expresses meta-information about signed digital documents, so that processors of those documents are aware of what algorithms were used to encrypt them. XML Encryption also allows for the encryption of entire XML documents or parts of XML documents, allowing systems to use the unencrypted data even when they cannot decrypt the confidential data.

Systems use XML Encryption and XML Signature together when both signing and encrypting a document. In order to check the signature of the data, a system must first decrypt the signed document with a decryption algorithm. This decryption algorithm is also working its way through the W3C. The decryption algorithm proposal notes that when a document is first signed and then encrypted, the XML Signature reveals the message digest value of the signed resource, which is useful information for an attacker.

There are additional problems with XML Encryption. As with general encryption, there's no problem digitally signing an XML document as a whole. However, systems will have difficulty when signing parts of a document, perhaps by

Copyright © 2002, ZapThink, LLC

*XML Encryption is not yet ready for prime-time since many lingering issues remain to be resolved.*

different people, especially when each party wants to encrypt their part of the message. Combine these multi-party issues with the XML formatting issues that led to XML Canonicalization, and it is clear that much work remains to resolve the issues behind XML Encryption.

### 3.5 Web Services Security Efforts

Web Services security efforts fall into three basic categories:

- XML Signature services (xml-dsig), as covered in section 3.3.4
- XML authentication and authorization services (SAML and XACML)
- XML key management services (XKMS)
- WS-Security, which provides a comprehensive Web Services security model that supports and integrates the other efforts to provide broad interoperability

#### *3.5.1 SAML*

*SAML (the Security Assertion Markup Language*) is a standard for exchanging authentication and authorization information between systems or domains. SAML is a reasonably mature specification (for a version 1.0 standard).

The SAML specification includes three types of assertions:

- Authentication assertions (facts that users have proven their identities)
- Attribute assertions (information about the user such as credit limits)
- Authorization decision assertions (whether the user is authorized to buy an item, for example)

*SAML concerns itself only with authentication and authorization. Its main goal is to provide a standard procedure for enabling single sign-on across organizational boundaries using Web Services.*

SAML concerns itself only with authentication and authorization. Its main goal is to provide a standard procedure for enabling single sign-on across organizational boundaries using Web Services. The SAML protocol describes how systems request and retrieve assertions, using SOAP over HTTP (in SAML 1.0; additional wire protocols should appear in future versions of the specification). The SAML protocol then defines the request and response messages and the simple choreography for using them. SAML allows for 10 subject authentication mechanisms, including user name/password, Kerberos ticket, SSL certificates, PKI public keys, XKMS public keys and XML Signatures.

A client starts a typical SAML exchange by sending a Web Service request containing both a SAML authentication request and the resource the client wants to access. The client making the request can be a person, a server or a Web Service. If the SAML server successfully authenticates the client, it will return a digitally signed security token to the client (using the xml-dsig standard). The token is valid only for a certain amount of time and can restrict the user to particular levels of access, such as read, write or delete. The client can then use the SAML token to make further requests of any system or Web Service that trusts the SAML server, even across company boundaries. This single-sign-on capability allows systems to chain Web Services together, where the same client must authenticate to multiple Services without having to recheck user credentials.

OASIS declared SAML 1.0 final in April 2002, and is expected to ratify it as a standard specification this summer. Vendors such as **Baltimore Technologies**, **Quadrasis**, **Netegrity**, **Systinet**, **Waveset** and **VeriSign** have based products on SAML.

Copyright © 2002, ZapThink, LLC

### 3.5.2 XACML

*XACML (XML Access Control Markup Language),* also developed by OASIS, allows systems to express access control policies in XML. XACML is a specification for capturing authorization and entitlement policies for resources. When a Web Service receives a SAML assertion, the Web Service sends a request to a SAML PDP (Policy Decision Point), which then checks an XACML policy via a PRP (Policy Retrieval Point). Using XML for access control allows systems to replicate policies from various access control products easily, using XACML as a common data format. The XACML initiative defines a structured entitlement or policy language, something the SAML committee chose to leave out in its first iteration.

XACML addresses fine-grained control of authorized activities (e.g., read, write, copy, etc.) based on access requester characteristics, the protocol over which the request is made, and the authentication mechanism. It is important to point out that a compliant Policy Decision Point (PDP) may choose a representation entirely different from XACML for its internal evaluation and decision-making processes. That is, XACML may be regarded simply as a policy interchange format, with any given implementation translating the XACML policy to its own local policy language at some point prior to evaluation.

### 3.5.3 XKMS

*XKMS (XML Key Management Protocol),* a W3C specification, provides a means for using Web services to simplify a number of complex PKI protocols and processes. XKMS incorporates X-KISS (XML Key Information Service Specification), which provides a means to query the trustworthiness of a user's digital certificate. XKMS also incorporates X-KRSS (XML Key Registration Service Specification), which enables systems to register digital certificates with an XKMS service.

*XKMS brings the advantages of XML to PKI. XKMS simplifies the integration of PKI with a wide variety of XML-aware applications. XKMS shifts essential PKI processes away from the desktop to the PKI server.*

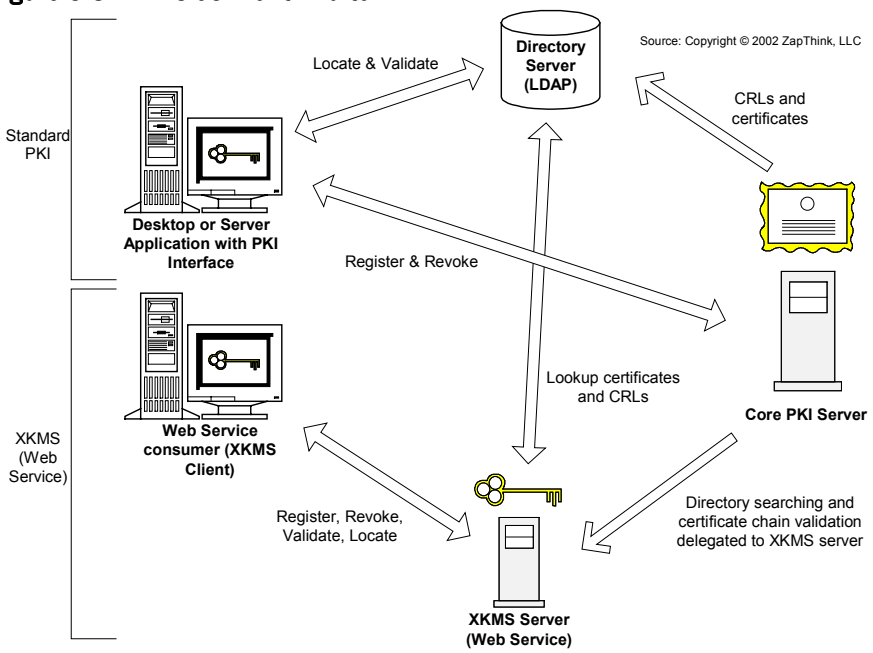
XKMS brings the advantages of XML to PKI. XKMS uses XML to augment PKI in order to outsource key registration, validation, and other trust processes to third party systems. The standard gives developers a common interface for such processes, letting them ignore the underlying PKI. Therefore, developers will be free to focus on Web Services applications rather than the complex tasks associated with PKI.

XKMS simplifies the integration of PKI with a wide variety of XML-aware applications. XKMS deals with the key management aspects of PKI, including the issuing of digital certificates and reporting on their status, but not the cryptographic functions of PKI, namely signing and encryption. XKMS provides a common communication channel for applications that consume PKI services, and shifts essential PKI processes away from the desktop to the PKI server. Companies use cryptographic capabilities in their operating system or existing applications in conjunction with XKMS to support their security requirements. In particular, such companies no longer need to deploy additional PKI toolkits to support multiple PKI protocols.

Because XKMS addresses issues associated with PKI integration, it acts as a front end to PKI to streamline the PKI development effort, as shown in figure 3.8 below. XKMS is an appropriate front end to PKI, because PKI resolves the issues that arise in heterogeneous security environments. In particular, PKI can be invisible to the Web Services implementation, while administrators are still able to use mature PKI products.

Copyright © 2002, ZapThink, LLC

**Figure 3.8: XKMS as Front End to PKI**




## Vendor Focus

*VeriSign*
*Microsoft*
*WebMethods*
*Baltimore Technologies*

**VeriSign**, **Microsoft**, and **webMethods** created XKMS and submitted it to the W3C. **Baltimore Technologies** also has a leadership role in the development of XKMS. It contains two parts:

- The *XML Key Information Service Specification* (X-KISS) allows a client to delegate to a third-party Trust Service some or all of the tasks necessary to process an XML Signature. This ability is useful for developers who do not want to implement signature checking themselves, or who simply want to delegate this functionality to an Application Service Provider or other third party who might better be able to act as a Trust Service.

- The *XML Key Registration Service Specification* (X-KRSS) is an XML-based replacement for existing binary PKI file formats for when a user applies for a digital certificate.

### *3.5.4 X-KRSS*

X-KRSS supports servers that register, revoke, and recover public and private key pairs. These keys enable a system to authenticate a user, either as part of PKI or another secret key encryption scheme. An X-KRSS registration server acts as an intermediary between the client and the PKI service that stores key and user information. X-KRSS servers serve a similar role to certificate authorities in PKI.

X-KRSS registration supports the following modes:

- The client generates the public and private key pair and sends the public key to the registration server. The server can then ask the client for *proof of possession* (POP) of the matching private key.

- The X-KRSS registration server can store client user names and other attributes that are bound to the client's public key. In this mode, the X-

Copyright © 2002, ZapThink, LLC 

KRSS server can either generate the certificate or delegate the certificate generation process to another PKI service.

- The X-KRSS server generates the key pair and sends the private key to the client using the XML Encryption. This mode would be useful if the client loses its private key.

*3.5.5 X-KISS*

The X-KISS protocol provides for assertion servers that can locate and validate keys from an XKMS registration service. By using features in the XML Signature specification, X-KISS can support three tiers of service that describe key location and validation functions that both clients and assertion servers are able to perform:

- Tier 0: The client locates and validates keys.
- Tier 1: The client delegates locating keys to the assertion server, but performs the key validation itself.
- Tier 2: The client delegates both the location and validation of the keys to the assertion server.

Clients operating at tier 0 can support many PKI and digital certificate tasks, as well as XML-related security tasks. For example, if a Web Services client receives a document digitally signed following the XML Signature specification, the client can look inside the signature to determine the URL where it can find the certificate containing the public key. On tier 1, the server locates information describing the public key and sends it to the client, which then validates that information locally. Tier 1 is especially useful in situations where the client doesn't fully trust the server. Finally, tier 2 allows the server to take on both tasks itself, in situations where the client fully trusts the server. Tier 2 is especially suited for clients that do not have the power to perform the key location and validation functions themselves.

*3.5.6 WS-Security*

Finally, *WS-Security (Web Services Security)* provides a broad set of specifications that cover security technologies that apply to authentication, authorization, privacy and confidentiality, trust, delegation, and auditing (non-repudiation) across a broad range of existing security frameworks and technologies, including both PKI and Kerberos. The WS-Security specification defines a comprehensive Web Service security model that supports and integrates the security models and technologies discussed above at a level of abstraction that enables multiple, heterogeneous systems to interoperate securely. Currently, WS-Security's creators, **IBM**, **Microsoft**, and **VeriSign,** are driving the development of WS-Security. These vendors published the WS-Security 1.0 specification in April 2002, and **IBM** so far is the only company that has released prototype software based on the specification. Microsoft has also announced their TrustBridge initiative which they are basing on WS-Security and Kerberos, but they have not yet released any related software.

**Vendor Focus**

***IBM***
***Microsoft***
***VeriSign***

WS-Security applies to Service-oriented architectures at a higher level than SAML in that it doesn't establish a universal framework to express user policies, nor does it provide a way to pass security credentials among security domains to build single-sign-on systems. However, WS-Security does support data encryption. SAML, in contrast, currently requires the use of encrypted transport mechanisms like HTTPS (SSL over HTTP) to provide message security.

Because WS-Security works at a level of abstraction above the nuts-and-bolts of application level security, WS-Security does not ensure security by itself, and it

Copyright © 2002, ZapThink, LLC

**Decision Point**

*WS-Security does not ensure security by itself, and it doesn't provide a complete security solution. Instead, companies must use WS-Security in combination with other Web Service security protocols.*

doesn't provide a complete security solution. Instead, companies must use WS-Security in combination with other Web Service security protocols to accommodate a variety of security models and encryption technologies. WS-Security explicitly builds upon certain foundational technologies, including SOAP, WSDL, XML Signatures, XML Encryption and SSL. The intention is to provide a framework that allows Web Service providers and consumers to develop solutions that meet their own individual security requirements.

The WS-Security specification includes a broad set of specifications including authentication, authorization, privacy, trust, integrity, confidentiality, secure communications channels, federation, delegation and auditing. In addition, WS-Security operates across a broad range of applications and architectures. WS-Security provides an extensible and flexible framework that its creators have especially designed to enable companies to leverage existing investments in security technologies, even when the other parties they wish to securely communicate with use different technologies.

The three vendors who are driving WS-Security are taking a phased approach to rolling out WS-Security. The current effort, which they released in April 2002, consists of a roadmap and an initial "seed" specification. This early draft defines the core facilities for protecting the integrity and confidentiality of messages, as well as mechanisms for associating security-related claims with messages. WS-Security's creators then hope to work with the industry and one or more standards bodies to flesh out the follow-on issues of policy, trust and privacy The roadmap also discusses additional issues, including firewall processing, privacy, browsers and mobile clients, access control, delegation, and auditing.

The initial specifications that make up the current draft of WS-Security include:

- *WS-Security* describes how to attach signature and encryption headers to SOAP messages. It also describes how to attach security tokens to messages, including digital certificates and Kerberos tickets. The term "WS-Security" formally refers to this particular specification as well as referring informally to the overall group of specifications.
- *WS-Policy* describes the capabilities and constraints of both security and general business policies on both message intermediaries and endpoints. For example, WS-Policy describes required security tokens, supported encryption algorithms, and privacy rules.
- *WS-Trust* describes a framework for trust models that enable Web Services to interoperate securely.
- *WS-Privacy* describes a model for how Web Services and requesters can state their own privacy preferences as well as organizational privacy statements.

The follow-on specifications that IBM, Microsoft, and VeriSign are looking to the industry to help develop include:

- *WS-SecureConversation* describes how to manage and authenticate message exchanges between parties, including security context exchanges and establishing and creating session keys.
- *WS-Federation* describes how to manage and broker the trust relationships in a heterogeneous federated environment, including support for federated identities.
- *WS-Authorization* describes how to manage authorization data and policies.

Copyright © 2002, ZapThink, LLC

The basic goal of WS-Security is similar to that of SAML, which is to provide interoperable, XML-based mechanisms for providing access to Web Services. Unlike SAML, however, WS-Security doesn't provide for the transfer of security authorization from one system to another, which is necessary to support single sign-on capabilities, because WS-Security doesn't separate the action of authentication from the Web Service request itself. Systems that support WS-Security will either accept or reject the request as a whole, and since every request must include authentication, WS-Security takes a much less efficient approach to authentication than SAML does.

*WS-Security takes a much less efficient approach to authentication than SAML, but WS-Security has broader applicability than SAML.*

WS-Security also goes further than SAML in providing ways to seal the entire SOAP message against tampering, both by signing it to ensure sender identity and encrypting it to provide confidentiality. SAML provides guarantees of sender identity and confidentiality only for the SAML security tokens themselves, not for the rest of the Web Service request.

WS-Security accomplishes message signing and encryption by using the XML Signature and XML Encryption standards. However, WS-Security does not depend on HTTP to provide these protections, while SAML does, so at this point, WS-Security has broader applicability than SAML. It remains to be seen, however, whether a future version of the SAML specification will either remove this limitation, or move toward a specification that is unified with WS-Security.

## IV. Conclusions

Security is the next major roadblock impeding Web Services adoption. Therefore, many vendors are focusing their resources to develop solutions to the multitude of security-related problems that companies will face as they seek to implement XML and Web Services solutions across the enterprise, and in particular, among business partners.

Nevertheless, XML and Web Services security is on the front edge of an emerging market, and as such, the entire segment is facing a period of dramatic turbulence. Business models will continue to shift, competitors will enter and leave the market, and companies will face mergers, acquisitions, and failures. Much of this turbulence will occur in a period before there is solid, well-understood demand for the products and services themselves.

*The best positioned companies to be profitable in the XML and Web Services security space are those companies that already have deep technical knowledge of application level security technologies, coupled with a solid customer base.*

The best positioned companies to be profitable in the XML and Web Services security space are those companies that already have deep technical knowledge of application level security technologies, coupled with a solid customer base. As those customers demand Web Services security, the established security vendors must be able to rise to the occasion.

However, we are not saying that there aren't opportunities for startups and other new entrants to the XML and Web Services security marketplace. As Web Services move beyond the initial "horseless carriage" phase, and enterprises realize that they can leverage open standards-based, loosely coupled Web Services to transform the way they use IT, there will continue to be opportunities for smaller, nimble players who have the resources and courage to push the envelope.

### 4.1 Key Notes

- *Locking 19 entrances to a building does not provide security if there are 20 entrances.*

Copyright © 2002, ZapThink, LLC

- *The primary use for Web Services today is for internal integration.*
- *The next roadblock on the path to Web Services adoption is security. Security is today's key enabler for Web Services.*
- *Most XML and Web Services security offerings on the market fall within the 3A (authentication, authorization, and administration) segment of the IT security market.*
- *Most 3A products will be Web Service-enabled by 2005.*
- *XrML-based DRM solutions will not form an independent market by themselves. Instead, the DRM space will eventually merge with the work in XML and Web Services security.*
- *Application level security focuses on which services users can request, based upon what roles those users have in the system and also provides for the confidentiality and integrity of the transmitted data.*
- *There must be an identification mechanism outside of the software system that provides the initial confirmation of identity.*
- *The ability to provide and administer authorization across multiple systems is a difficult problem that many of the companies covered in this report are trying to solve.*
- *Intermediaries are especially important in the context of Web Services, because the SOAP protocol is designed to support intermediaries that can forward or reroute SOAP messages.*
- *Certificate-based authentication is more secure than simple password-based authentication, because the user must have the private key as well as the password.*
- *PKI includes the processes for issuing and storing certificates and keys, renewing and revoking certificates, and procedures for delegating the certificate issuing responsibility.*
- *The principles of application level security, key encryption, digital signatures, PKI, SSL, and Kerberos form the basis for all work in XML security.*
- *XML Signature brings the well-known benefits of XML to digital signatures, making them human-readable, easily parsed, platform independent, and simpler to implement.*
- *XML Encryption is not yet ready for prime-time.*
- *SAML concerns itself only with authentication and authorization. Its main goal is to provide a standard procedure for enabling single sign-on across organizational boundaries using Web Services.*
- *XKMS brings the advantages of XML to PKI. XKMS simplifies the integration of PKI with a wide variety of XML-aware applications. XKMS shifts essential PKI processes away from the desktop to the PKI server.*
- *WS-Security takes a much less efficient approach to authentication than SAML, but WS-Security has broader applicability than SAML.*
- *Some of the vendors in this report may be selling solutions that nobody will ever want to buy.*
- *Web Services are particularly suited for building access and policy management solutions, because they are coarse grained and loosely coupled.*
- *Web Services offer great potential for B2B communication and integration, but the lack of robust security and manageability solutions currently inhibit the ability for companies to conduct business with each other via Web Services over the Internet.*
- *The solutions that are appearing are quite sophisticated and broadly applicable to customer scenarios that are for the most part still on the drawing board.*
- *The combination of adequate funding, solid business models, seasoned management teams, and high quality engineering staff leads startups to offer surprisingly robust XML and Web Services security solutions.*

Copyright © 2002, ZapThink, LLC

- *There is a market for XML security solutions for companies looking to address current risks unrelated to Web Services.*
- *There will be a spike in demand for Web Services security solutions within the next 12 months.*
- *Web Services will not play a major role in transactional environments in 2002-2003.*
- *The 2003 timeframe won't see many multiple-company B2B Web Services, because companies will choose to implement B2B Web Services on a point-to-point basis.*
- *The XML and Web Services security market will reach $4.4 billion in 2006, which will represent 65% of the total 3A security market. This growth represents an average CAGR of 300%.*
- *Existing 3A security vendors will incorporate XML and Web Services into their product lines, so by 2006, most 3A security products will support or provide XML and/or Web Services security.*
- *The best positioned companies to be profitable in the XML and Web Services security space are those companies that already have deep technical knowledge of application level security technologies, coupled with a solid customer base.*

### 4.2 Decision Points

- *This report must be placed into the context of an overall security strategy. Simply securing all of a company's Web Services alone only provides a false sense of security.*
- *Next-generation firewalls must be capable of looking at the content of XML streams, and the security mechanisms for such data must be part of that content.*
- *Enterprises must institute policies that apply to their entire enterprise network (including participants invited from outside), and administer that security in a hierarchical fashion.*
- *There are two main limitations to SSL. First, SSL does not provide audit trails, and second, SSL is point-to-point.*
- *WS-Security does not ensure security by itself, and it doesn't provide a complete security solution. Instead, companies must use WS-Security in combination with other Web Service security protocols.*
- *There is no way to accurately calculate the ROI of a security solution.*
- *Companies planning on using Web Services across the firewall will necessarily have to resolve the resulting security issues first.*

### 4.3 Figures



Copyright © 2002, ZapThink, LLC 

## V. Vendor Profiles

### 5.1 Web Services Security Platforms

*Westbridge Technology*
Please see ZapNote ZTZN-0612

*Quadrasis*
Please see ZapNote ZTZN-0304

*Baltimore Technologies*
Please see ZapNote ZTZN-1008

### 5.2 Secure Integration Vendors

*Actional*
Please see ZapNote ZTZN-0280

### 5.3 Global Trust Services

*VeriSign*
Please see ZapNote ZTZN-1102

*Entrust*
Please see ZapNote ZTZN-1038

### 5.4 Identity Management/Authorization/Single Sign-On Vendors

*Netegrity*
Please see ZapNote ZTZN-1069

*Oblix*
Please see ZapNote ZTZN-1073

*OpenNetwork*
Please see ZapNote ZTZN-1075

*Entegrity*
Please see ZapNote ZTZN-1037

*OneName*
Please see ZapNote ZTZN-1074

### 5.5 Access & Policy Management Vendors

*Waveset*
Please see ZapNote ZTZN-1106

### 5.6 Software XML Firewalls

*Vordel*
Please see ZapNote ZTZN-0238

Copyright © 2002, ZapThink, LLC

### 5.7 PKI Vendors

*RSA Security*

Please see ZapNote ZTZN-0610

### 5.8 Enterprise Security Services

*TruSecure*

Please see ZapNote ZTZN-0613

Copyright © 2002, ZapThink, LLC

## A. Related Research

***Reports***

- *Web Services Technologies and Trends* Report (ZT-WEBSRV)
- *Pros and Cons of Web Services* Report (ZTR-WS101)
- *Service-Oriented Integration* Report (ZTR-WS103)
- *XML Proxies* Report (ZTR-DI101)
- *Service-Oriented Management* Report (ZTR-WS106)

***ZapNotes***

- *BEA* ZapNote (ZTZN-1009)
- *ContentGuard* ZapNote (ZTZN-0213)
- *Forum Systems* ZapNote (ZTZN-0212)
- *Grand Central* ZapNote (ZTZN-0215)
- *IBM* ZapNote (ZTZN-1050)
- *InterTrust* ZapNote (ZTZN-0265)
- *IONA* ZapNote (ZTZN-0140)
- *Microsoft* ZapNote (ZTZN-1066)
- *Novell* ZapNote (ZTZN-1071)
- *Sarvega* ZapNote (ZTZN-0271)
- *SeeBeyond* ZapNote (ZTZN-0279)
- *Sun Microsystems* ZapNote (ZTZN-1093)
- *Systinet* ZapNote (ZTZN-1096)
- *TIBCO* ZapNote (ZTZN-1100)
- *webMethods* ZapNote (ZTZN-1107)

## B. Supporting Resources

XKMS www.w3.org/TR/xkms
XML Signatures www.ietf.org/html.charters/xmldsig-charter.html
XML Encryption www.w3.org/Encryption
SAML www.oasis-open.org/committees/security/index.shtml
W3C www.w3.org
OASIS www.oasis-open.org
IBM's XML Security Suite www.alphaworks.ibm.com/tech/xmlsecuritysuite

Copyright © 2002, ZapThink, LLC

## Trademark Notice and Statement of Opinion

All Contents Copyright © 2002 ZapThink, LLC. All rights reserved. Reproduction of this publication in any form without prior written permission is forbidden. The information contained herein has been obtained from sources believed to be reliable. ZapThink disclaims all warranties as to the accuracy, completeness or adequacy of such information. ZapThink shall have no liability for errors, omissions or inadequacies in the information contained herein or for interpretations thereof. The reader assumes sole responsibility for the selection of these materials to achieve its intended results. The opinions expressed herein are subject to change without notice. All trademarks, service marks, and trade names are trademarked by their respective owners and ZapThink makes no claims to these names.

## About ZapThink, LLC

ZapThink is an IT market intelligence firm that provides trusted advice and critical insight into XML, Web Services, and Service Orientation. We provide our target audience of IT vendors, service providers and end-users a clear roadmap for standards-based, loosely coupled distributed computing – a vision of IT meeting the needs of the agile business.

ZapThink's role is to help companies understand these IT products and services in the context of SOAs and the vision of Service Orientation. ZapThink provides market intelligence to IT vendors who offer XML and Web Services-based products to help them understand their competitive landscape and how to communicate their value proposition to their customers within the context of Service Orientation, and lay out their product roadmaps for the coming wave of Service Orientation. ZapThink also provides implementation intelligence to IT users who are seeking guidance and clarity into how to assemble the available products and services into a coherent roadmap to Service Orientation. Finally, ZapThink provides demand intelligence to IT vendors and service providers who must understand the needs of IT users as they follow the roadmap to Service Orientation.

ZapThink's senior analysts are widely regarded as the "go to analysts" for XML, Web Services, and SOAs by vendors, end-users, and the press. They are in great demand as speakers, and have presented at conferences and industry events around the world. They are among the most quoted industry analysts in the IT industry.

ZapThink was founded in October 2000 and is headquartered in Waltham, Massachusetts. Its customers include Global 1000 firms, public sector organizations around the world, and many emerging businesses. ZapThink Analysts have years of experience in IT as well as research and analysis. Its analysts have previously been with such firms as IDC and ChannelWave, and have sat on the working group committees for standards bodies such as RosettaNet, UDDI, CPExchange, ebXML, EIDX, and CompTIA.

Call, email, or visit the ZapThink Web site to learn more about how ZapThink can help you to better understand how XML and Web Services impact your business or organization.

**ZAPTHINK CONTACT:**
ZapThink, LLC
11 Willow Street
Suite 200
Waltham, MA 02453
Phone: +1 (781) 207 0203
Fax: +1 (786) 524 3186
info@zapthink.com

ZapThink, LLC • 11 Willow Street, Suite 200 • Waltham, MA 02453 • info@zapthink.com • www.zapthink.com