zapthink white paper

CONTEXT & IDENTITY

THE LINCHPINS OF WEB SERVICES SECURITY

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

CONTEXT & IDENTITY: THE LINCHPINS OF WEB SERVICES SECURITY

November 2003

Analyst: Jason Bloomberg

Abstract

Enterprise identity and access management capabilities are a fundamental prerequisite for the implementation of mission-critical Web Services, because such Services provide a layer of abstraction that hides the complexity of underlying technology while at the same time providing increased value to the business user of those Web Services. Such an abstraction layer, however, can lead to the loss of security context, where the information about the identity and access privileges of a requester of a particular Service may be lost to the underlying applications that provide the data and functionality being requested.

Identity-based Web Services security solutions like Netegrity TransactionMinder restore and maintain this security context as an extension of their policy-based approach to enterprise identity management. By supporting the new Web Services security standards like SAML and WS-Security, TransactionMinder can provide the necessary identity management infrastructure for Web Services conversations that extend beyond the enterprise to users at business partners and other companies.

Table of Contents

I. What are Web Services?

Web Services are little more than standards-based interfaces to software functionality.

Fundamentally, Web Services are little more than standards-based interfaces to software functionality. Unfortunately, Web Services are both poorly named and vaguely defined. As a result, there is still a good measure of confusion in the marketplace about exactly what Web Services are, what their importance is, and where they fit into the overall IT picture. Nevertheless, the phrase "Web Services" has, for better or worse, become the *de facto* term for a range of capabilities.

A strict definition of Web Services is *encapsulated, loosely coupled, contracted software objects offered via standard protocols*. Essentially, Web Services are interfaces to application functionality residing on systems that accept requests from other systems locally or across the Internet by means of lightweight, vendor-neutral communications technologies. Breaking down the definition should make it clearer:

- *Encapsulated* means that the implementation of each Web Service is invisible from outside the Web Service. Its functionality is known only by the interface it exposes.
- Web Services can be *loosely coupled*. Loosely coupled means that Web Services and the programs that invoke them (known as Web Service consumers) can be changed independently of each other, instead of requiring a redesign of the involved components.
- *Contracted* means that the Web service's behavior, as well as how to connect, or bind to it, and its input and output parameters, are available to those consumers who are able to access it.
- Web Services are built upon *standard protocols*, including XML (eXtensible Markup Language), as well as SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) and UDDI (Universal Description, Discovery, and Integration), which are all standard protocols based upon XML.

Web Services' loosely coupled, contracted nature provides a layer of abstraction that masks the underlying complexity of heterogeneous technology while providing business-oriented Web Services to users.

This strict definition, however, only tells a part of the story. On the one hand, because Web Services are standards-based interfaces, they reduce the cost and complexity of integration. On the other hand, their loosely coupled, contracted nature provides a layer of abstraction that masks the underlying complexity of heterogeneous technology while providing business-oriented Web Services to users. To achieve the benefits that the Web Services layer of abstraction can provide, however, requires that the IT organization address many issues, including management, data integration, business process implementation, and most importantly, security.

Getting Started with Web Services

Companies often get started with Web Services when small project teams leverage the benefits of Web Services to solve point-to-point integration problems, typically under the radar of executive management—what might be called "grass roots" adoption of Web Services. Such grass roots projects typically use Web Services as an inexpensive means for connecting heterogeneous systems.

As companies build expertise in Web Services techniques and an understanding of their value, they build individual Services that serve critical roles. This broad application of Web Services to complex integration projects includes n-tier projects like portals and eCommerce engagements, supply chain management projects, and EAI projects that use Web Services extensively throughout the project. It is at this phase that the issues of security and identity management become critical to the rollout of Web Services.

Once Web Services have achieved a certain level of acceptance in the organization, architecture teams or other key IT personnel should initiate pilot projects leveraging the dynamic discovery, location independence, and loose coupling capabilities of Service-Oriented Architectures (SOAs). As companies complete successful pilot projects, then, they typically look to leverage their assets and expertise to build out comprehensive, *enterprise SOAs* that abstract IT functionality both within the enterprise and among business partners, enabling Service-oriented processes.

The power of abstraction

For many enterprises, Web Services have moved beyond the hype stage and are now a reality.

For many enterprises, Web Services have moved beyond the hype stage and are now a reality. Throughout the past two years, numerous companies have begun to take steps towards implementing Web Services in ways that provide incremental improvements over existing technologies. Hundreds of companies of different sizes and industries have built Web Services pilot projects, proving that this most recent evolution of distributed computing technology can reduce integration and development costs substantially. ZapThink predicts that the market for Web Services software will surpass $21 billion by 2008. Forward-looking enterprises are now looking to take the next step and figure out how to leverage the power of Web Services strategically across the enterprise.

This next step means moving beyond simple point-to-point applications of Web Services to a broad application of Web Service technologies both within the enterprise and increasingly among business partners. This transition requires more than a simple change in programming practices. This broad application of Web Services technologies requires a *layer of abstraction* that masks the complexity of the technology in an organization. This abstraction can present functionality via loosely coupled Services that offer clear business value, independent of the underlying technology that supports them. This layer of abstraction hides the complexity of existing architectures.

Layers of abstraction find their roots in the very earliest software. The first programmable digital computers dealt in the world of zeroes and ones—that is, *only* zeroes and ones. Programs were zeroes and ones. Output consisted

of zeroes and ones. As a result, programming was very difficult and programs were quite opaque. Into this black-and-white world came programs called compilers that let programmers work with English-like languages like COBOL. The compiler then took the COBOL code, crunched it, and spit out the zero-and-one object code that the computers actually understood. The COBOL compiler, therefore, *virtualized* the object code, creating a layer of abstraction that both masked the underlying complexity of the technology while at the same time provided more power to the users of that technology.

As computers grew more powerful and complex, such virtualization techniques continued to provide additional levels of abstraction. Timesharing mainframe computers allowed users to have virtual control of the machines. Another example is the graphical user interface, which provided virtual access to underlying system resources. Component architectures also provided virtual representations of distributed computing infrastructures. At every step, software allowed people to work with relatively simple tools that accessed complex systems behind the scenes. That's the power of abstraction: enabling the tools people use to get simpler as they become more powerful.

Web Services, then, are currently enabling an evolutionary step in this inexorable progression to the next level of abstraction for distributed computing. For want of a better term, this level of abstraction is called a *Web Services network*. A Web Services network is a logical construct (as opposed to a physical network) that describes the available Web Services, their relationships to each other, and to the consumers of those Services. The promise of Web Services networks is unquestionably appealing, but there are also substantial challenges that companies must face to achieve the full benefit of Web Services. The first such challenge that companies must address is security.

A Web Services network is a logical construct that describes the available Web Services, their relationships to each other, and to the consumers of those Services.

The role of the user

To understand why security is the first challenge facing companies as they seek to implement mission-critical Web Services, it's important to keep in mind that Web Services are interfaces for software-to-software communication. In other words, unlike the World Wide Web that gave Web Services their name, Web Services are not for user-to-software interaction. The Web Service consumer may or may not have its own user interface; in some cases, the Service consumer is several steps removed from the human being who initiates a particular request.

The second point to keep in mind is that unlike the Web, Web Services are typically asynchronous. User requests may be one part of a long-running process, where the user action is separate in time from the business logic execution that takes place as a result of that action. Furthermore, Web Services support event-driven processes, which are not necessarily associated with a user request at all.

User requests may be one part of a long-running process, where the user action is separate in time from the business logic execution that takes place as a result of that action.

If a company's IT functionality was intended to be publicly available, and their data were meant to be freely available for all to see, then Web Services wouldn't provide a serious security concern. In reality, of course, data are usually confidential, and functionality is only meted out to authorized users. Web Services, therefore, exacerbate the problem of IT security, because the consumers of those Services are themselves software. Furthermore, standards-based interfaces are by definition easier to access than

proprietary, closed interfaces. In other words, we're unlocking our doors *and* looking the other way with Web Services.

II. Context and identity: The challenges of Web Services security

The fundamental problem with securing Web Services is one of *context*. Security context is a set of information about the user of a Service, including the rules and policies that apply to that user, as well as information about the business process or transaction the user is currently participating in. When the user is separated from the Service, that context is lost. The challenge with securing Web Services is therefore to maintain this context, even when the user is many steps removed from the Service. When Web Services provide a layer of abstraction, it is also important to maintain the security context across the layer of abstraction. In this case, however, a single Web Service might expose functionality from several systems and applications, and in some cases, those applications may change over time. Therefore, the security context must be maintained across the entire IT infrastructure to ensure the appropriate security to the Web Services network.

The security context must be maintained across the entire IT infrastructure to ensure the appropriate security to the Web Services network.

Identity management: the key to maintaining security context

To get a handle on the problem of Web Services security, it makes sense to start with the users. Because a Web Services network can potentially touch all the users and applications in an enterprise—as well as users and applications at other companies—the enterprise must be able to manage the identities of those users, as well as the security policies that apply to those users, separate from the applications that interact with those identities. An enterprise identity management solution is therefore a critical prerequisite to building a Web Services network.

An enterprise identity management solution is a critical prerequisite to building a Web Services network.

Identity management is a set of processes for the creation, maintenance, and use of identities and their attributes, as well as credentials and entitlements, plus a supporting infrastructure. Identity management involves both technology and process. It must enable enterprises to create a manageable user lifecycle, where it meets the business needs of rapid registration, use, and termination of users. An identity management solution must also be able to scale from internally facing systems to externally facing applications and processes.

The problem with many existing identity management approaches is that they are built in an *ad hoc* fashion, as an IT organization builds one application or system at a time. Furthermore, most of today's applications and operating systems lack a scalable means for managing identity, credentials, and policy across boundaries, leading to a fragmented identity infrastructure at many enterprises. Such fragmented infrastructures contain overlapping identity repositories, inconsistent policy frameworks, and process discontinuities. Such infrastructures are also error prone, expensive to manage, and can have weaknesses that can lead to security breaches.

Companies that have fragmented identity infrastructures often have what is infamously called the "sticky note problem": employees often have so many usernames and passwords that they must write them on sticky notes and stick them on their computer displays, defeating the entire purpose of having passwords in the first place. Such companies also find that the majority of

their help desk activities involve resetting lost or forgotten passwords—clearly an enormous, avoidable expense. Even before a consideration of Web Services crosses the radar of many enterprises, the sticky note problem often leads to enterprise identity management requirements known as "single sign-on." *Single sign-on* (SSO) is the ability of an identity management solution to enable users to log into their corporate application infrastructure with a single username and password, where the identity management solution then automatically logs them into every other application and system they are entitled to access.

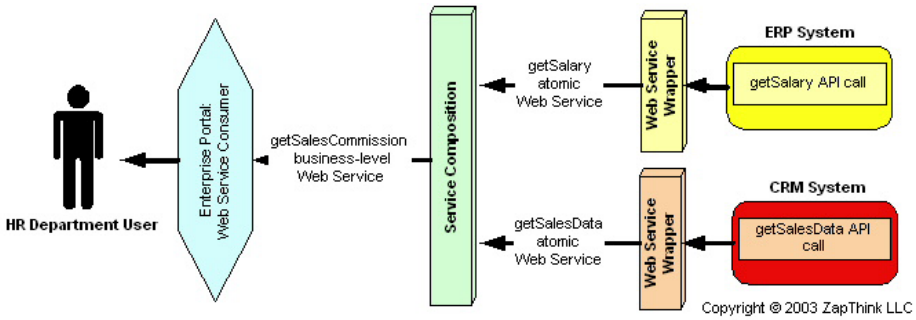
Identity management solutions typically offer directory services, and provide an authoritative identity repository that contains people, organizational units, groups, and roles. The solution can then provide authentication based on the identity information stored in the directory, based on user attributes like roles and groups. Many identity management solutions then use meta-directories to synchronize identity repositories and authoritative sources of security policies. In addition, identity management solutions offer user management, including the creation, propagation, and maintenance of user accounts and rights. These solutions also have provisioning systems that support workflows that automate processes, reduce administration costs, and enhance security. They offer centralized administration of roles, groups, and policies, as well as centralized password management and rapid termination of accounts.

In addition to the above capabilities, identity management solutions frequently offer delegated administration tools that distribute the workload and liability for user management, enabling enterprises to assign a subset of the overall user administration authority to a designated user or group. These solutions also increase the ability for users to take advantage of self-service password resetting, registration, and subscription services that can kick off workflow and provisioning processes. To provide access management, identity management systems determine rights and privileges using policy-based systems.

Regaining context in a Web Services network

To understand how an identity management solution can maintain the user context within a Web Services network, it's important to illustrate how that context can be lost. Figure 1 below shows an example of an enterprise portal that accesses a getSalesCommission Web Service. This business Service touches upon two different back-end systems—the ERP system that provides a getSalary API, and a CRM system that exposes a getSalesData API. The ERP and CRM systems each have their own security policies, with separately defined users, indicated by yellow and red. The HR user logs into the portal, thereby being authenticated at the interface. The security policy for the getSalesCommission Service (indicated in orange), however, is related to, but different from the policies governing the underlying systems.

Figure 1: Securing an HR Service



The challenge in an example like the one above is that the enterprise needs a single identity management and security policy infrastructure that governs the access to the four interfaces in the example (the portal, the business Service, and the two atomic Services) in a way that allows for flexibility in the event the systems, Services, or policies change.

To address this challenge, the identity management system contains a centralized repository of identity and policy information that every component in Figure 1 can access. When the user logs in, the portal authenticates that user via the identity management system, and creates a *token* that identifies that user and the privileges that user has. That token then remains with the user request as it traverses the Service composition layer, the atomic Web Services, and finally to the back-end systems. When the ERP and CRM systems receive the request, the token provides the context for that request, and those systems can use the token to validate the request within the identity management system.

Traditional distributed computing security was modeled by islands of security, which describe systems and users on isolated networks or subnetworks.

Such an approach to enterprise security is quite different from the traditional approach to application security in a distributed environment. Traditional distributed computing security was modeled by islands of security, which describe systems and users on isolated networks or subnetworks. In that scenario, the network acted as an island, with its own perimeter security, and only users within the network were considered to be trusted. This "trusted vs. untrusted" dichotomy breaks down in a Web Services network, because users can access Services located on systems across one or more network segments or enterprises. The concept of trusted groups no longer has the same 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 with a centralized root administrator. 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.

III. SAML and WS-Security: Standards for identity management

The focus of identity management within a Web Services network is on *authorization*, *confidentiality*, and *data integrity*. 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

ACLs are generally insufficient to handle the real-world security policies required at many enterprises.

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 a Web Services specification known as *WS-Security* is intended to address. WS-Security specifies an abstraction layer on top of any company's particular application security technology (PKI, Kerberos, etc.) that allows such dissimilar infrastructures to participate in a common trust relationship.

WS-Security specifies an abstraction layer on top of any company's particular application security technology that allows such dissimilar infrastructures to participate in a common trust relationship.

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 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.

Unlike confidentiality, *data integrity* comprises two requirements: first, the data received must be the same as the data sent. In other words, data integrity systems must be able to guarantee that a 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.

Preserving loose coupling while ensuring confidentiality and data integrity in a Web Services network is particularly challenging, because of the constraints such requirements put on both the Web Service producers and consumers. Specifications like WS-Security and the *Security Assertion Markup Language* (SAML) are intended to address the issue of preserving loose coupling by providing standard ways for both ends of a secure Web Services message to participate in the various forms of application security. An understanding of WS-Security and SAML is therefore an important aspect of understanding how identity management works in a Web Services network.

Specifications like WS-Security and SAML are intended to address the issue of preserving loose coupling by providing standard ways for both ends of a secure Web Services message to participate in the various forms of application security.

What is an assertion?

Assertions are XML representations that can concern authorizations, authentications, and attributes of specific resources within a Web Services network, including individuals, applications, or computer systems. These

resources, or *entities,* must be identifiable within a specific security context, such as a person who is a member of a workgroup or a computer that is part of a network domain. An assertion can be a claim, statement, or declaration, which means that whether or not the recipient of an assertion can accept it as being a true assertion based on to the integrity and authenticity of the entity making the assertion.

If you can trust the entity making the assertion, the assertion can be accepted as true with the same level of certainty as you have in the entity making the assertion.

The OASIS SAML specification allows trust assertions to be specified using XML. In the SAML model, authorities for attributes and for authentication play a key role, because of this transferal of trust. Essentially, if you can trust the entity making the assertion, the assertion can be accepted as true with the same level of certainty as you have in the entity making the assertion.

SAML places assertions in tokens. A *token* is an XML representation of security information, including assertions. A token may either be signed or unsigned. An example of an unsigned token would be a password or symmetric encryption keys used as shared secrets. Examples of signed security tokens are X.509 digital certificates (which are signed by a certificate authority) or a Kerberos ticket. If the system uses an unsigned token, then it must assure confidentiality must to insure that a third party doesn't intercept the token.

How SAML provides for portable trust authentication

This ability to transfer trust from one entity to another is one of the core benefits of SAML. SAML enables *portable trust* by supporting the assertion of authentication of single principals (users and systems) between multiple, different domains. Trust portability is an important goal for B2B Web Services. When multiple companies are involved in a cooperative B2B eCommerce system where there is an aggregation of services offered through a single intermediary, all participants (customers, partners, staff, and systems) must be authenticated in the same way across the entire eCommerce system. Naturally, each of these companies is responsible for having its own identity management system that acts as an authoritative system that is capable of authentication for all subjects within its own domain.

Normally, when a company offers a Web Service within their own domain, it is only available to principals that its own identity management system has authenticated within its domain. There is a problem, however, when a principal in one company's domain wants to access another company's Service whose authentication is handled by the second company. Clearly, it's usually not practical for one company to take on the responsibility for managing the identities of principals at another company. For example, a bank is not going to hand over its authentication data to an external party just so that party can invoke a Web Service from the bank, and vice versa.

SAML resolves this issue of authenticating principals in another company's domain with its ability to provide portable trust. The SAML token contains the assertions about a principal's identity, privileges, and roles, and the company creating the token vouches for its authenticity. A principal from a remote domain looking to access a local Web Service need only present its SAML token, and if the local identity management system trusts the company who generated the token, then it can accept the assertions within that token as true, and allow that principal to take the actions the assertions state it is authorized to take.

SAML Tokens and the role of WS-Security

WS-Security helps to complete the portable trust mechanism by defining how security tokens should be contained in SOAP messages and how XML Security specifications are used to encrypt and sign these tokens.

SAML, however, does not provide a complete portable trust mechanism, because it only specifies the format of the tokens and the assertions contained within them. WS-Security helps to complete the portable trust mechanism by defining how security tokens should be contained in SOAP messages and how XML Security specifications are used to encrypt and sign these tokens. WS-Security also specifies the XML elements and attributes that are used to enclose tokens into SOAP messages, as well as the means to enclose XML Signature and XML Encryption into SOAP.

WS-Security is part of the *Security in a Web Services World* roadmap of specifications from IBM, Microsoft, and VeriSign that includes later specifications such as WS-Trust, WS-Policy, and WS-SecureConversation. WS-Security is the first of these specifications to be submitted to OASIS for ratification as a standard. However, even though WS-Security is only the first in a series of security-related specifications, it can still be used by itself (and in conjunction with SAML).

WS-Security is particularly applicable when different companies are using different platforms, programming languages, or security technologies. One company may use Kerberos (the security technology used in Microsoft Windows), while another may consume X.509 certificates (a key component of PKI). Just as Web Services themselves provide a layer of abstraction, the IBM/Microsoft roadmap provides a layer of abstraction for companies using different security technologies to communicate securely using SOAP. This level of abstraction means not only that existing security infrastructure can be used for Web Services security, but that companies can incorporate new security technologies as they develop.

SAML explains how to express security assertions in XML format, where WS-Security describes how to place security information in SOAP messages.

WS-Security and SAML solve different problems: SAML explains how to express security assertions in XML format, where WS-Security describes how to place security information in SOAP messages. The SAML assertion is contained within a security block, which in turn appears inside the SOAP header. The processing of a SAML assertion, contained in a WS-Security formatted SOAP message, should be processed the same as any other type of security token expressed using WS-Security.

IV. Maintaining context and identity with Netegrity TransactionMinder

TransactionMinder leverages SiteMinder, Netegrity's flagship access management product, to provide a scalable solution that relies on the Netegrity Policy Server, which provides a shared, centralized point of control for Web Services identity management.

Netegrity TransactionMinder is a Web Services identity management solution that that centralizes the authentication, authorization, and audit activities for all Web Services transactions by leveraging a set of policy-based shared Services. TransactionMinder leverages SiteMinder, Netegrity's flagship access management product, to provide a scalable solution that relies on the Netegrity Policy Server, which provides a shared, centralized point of control for Web Services identity management, and empowers existing SiteMinder deployments to extend their existing Netegrity Policy Server to emerging Web Services projects.

Netegrity designed TransactionMinder to provide secure access to Web Services. Leveraging the Netegrity Policy Server, TransactionMinder delivers security policy as a shared Service, offering centralized authentication, authorization, audit, and federation services for a company's Web Services. TransactionMinder bases its authentication approach on message content

(for example, whether a message contains a purchase order) as well as the WS-Security and SAML standards.

TransactionMinder externalizes security logic outside of applications, relieving developers of the responsibility for coding security directly into their applications. Externalizing the security logic from the application speeds application development and prevents the creation of individual islands of security for each individual application. TransactionMinder also provides a single point of access control and administration for Web Services security by binding XML data flows to user identities. TransactionMinder supports fine-grained, document-based credential checking and generation, and it checks Inbound messages for authentication and authorization and adds credentials and attributes to outbound Web Services requests.

TransactionMinder also supports the federation of identities in the B2B environment by describing user identities to partners in WS-Security headers, and supporting the consumption of credentials issued by partners, including SAML assertions. The product also supports synchronized sessioning in multi-step chained Web Services transactions, which provides two key benefits. First, synchronized sessioning improves the Web Services' user's experience by offering single sign-on capabilities. Secondly, it improves Web Services performance by avoiding the need for XML message re-authentication within the same transaction.

Policy-based security Services

TransactionMinder bases its Web Services security on a central set of configurable, flexible policies that consistently enforce message level authentication, authorization, session management, federation and audit across multiple Web Services within an enterprise. TransactionMinder avoids the creation of isolated islands of security, and reduces security administration and software development costs.

TransactionMinder is compliant with the major Web Services standards, including XML, SAML, SOAP, and WS-Security. The product's WS-Security support provides the ability to issue and consume three different tokens, including password digest, X.509 certificates, and SAML assertions, allowing the enterprise to deploy a variety of authentication mechanisms to match various security requirements. TransactionMinder can also generate SAML session assertions, either in the SOAP envelope, the HTTP header, or an HTTP cookie. The TransactionMinder solution is designed to handle high transaction volume Web Services environments with support for load balancing, two-level caching, as well as user store replication and automatic fail-over.

Identity-based Web Services security

When using TransactionMinder, administrators can configure and define policies based on user identity and XML message content, creating a security context that they can enforce across multiple Web Services. The product binds XML requests to user identities, leveraging enterprise identity stores to integrate Web Services and existing enterprise-wide security into a single infrastructure.

TransactionMinder also provides a proper context and business relationship among Web Services networks at different companies and the associated authorized users regardless of architecture complexity. As a result, it

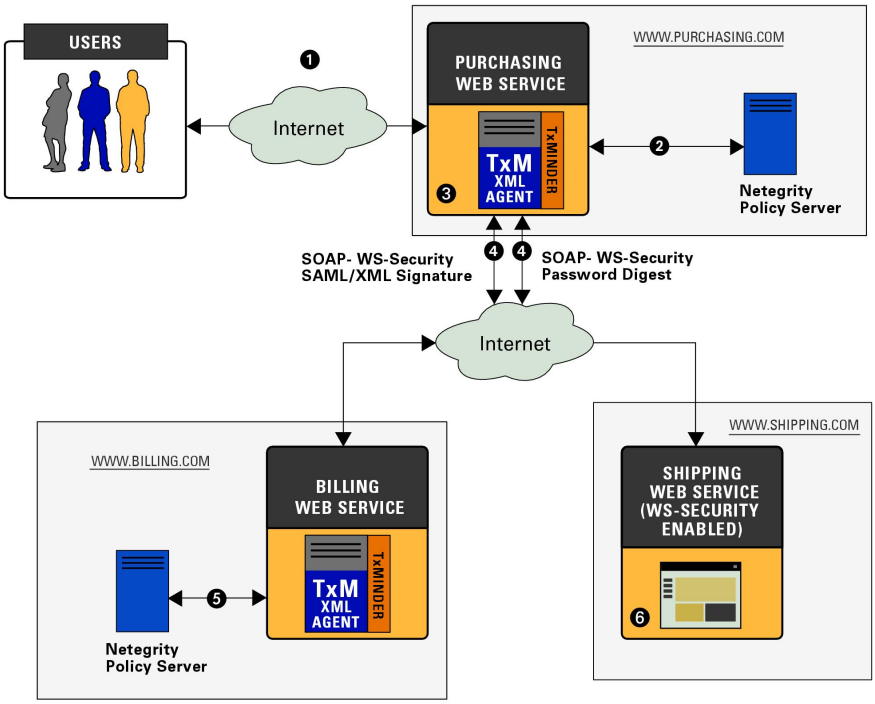
mitigates the deployment risk inherent in Web Services by leveraging existing systems, reducing complexity, and enforcing consistent security policies across the enterprise.

An example of TransactionMinder in action

The operation of TransactionMinder is straightforward, as shown in Figure 2 below.

1. The end-user submits a purchase order via an HTML form to the Purchasing portal (www.purchasing.com, for example).
2. TransactionMinder authenticates the user against the user stores configured in the Netegrity Policy Server.
3. TransactionMinder generates one SOAP message for the Billing Service, and one for the Shipping Service. The SOAP message for the Billing Service includes a signed SAML assertion in the WS-Security header, and billing information in the SOAP envelope body. The SOAP message for the Shipping Service includes the username and password digest in the WS-Security header, and shipping information in the SOAP envelope body.
4. The Purchasing portal posts both SOAP messages over SSL.
5. TransactionMinder at www.billing.com authenticates the sender by processing the SAML assertion.
6. The Shipping Service authenticates the sender using the username/password information.

Figure 2: TransactionMinder Example



TransactionMinder can also implement SSO across chained Web Services, authenticating users at the first Web Service, and then providing them a session token valid for all Web Services protected by TransactionMinder. TransactionMinder can also offer federation using the SAML token profile of WS-Security, allowing the export of user identities across B2B partners. Users identified via SAML assertions by an enterprise can then access resources provided by remote partner sites.

The ZapThink take

Enterprise identity management is a fundamental prerequisite for Web Services networks, forming a requirement that all companies looking to offer mission-critical Web Services must address. Furthermore, companies cannot effectively address identity management for Web Services separate from their broad-based identity and access management capabilities across the enterprise. Web Services herald a new set of approaches to IT security, where the closed applications and networks of the past give way to open applications and location-independent Services. The economic and technical forces that are leading companies to invest in enterprise identity management systems are actually much the same as the forces that are leading to the greater adoption of Web Services networks and the Service-oriented architectures based upon them.

Netegrity TransactionMinder is the right solution at the right time for many enterprises who are struggling with the various changes that the move toward open, standards-based computing requires.

Netegrity TransactionMinder is the right solution at the right time for many enterprises who are struggling with the various changes that the move toward open, standards-based computing requires. Such companies must resolve their identity management issues as well as plan and implement their Web Services networks, and Netegrity is one of the few security vendors who offers a comprehensive identity management solution that can help such enterprises solve their overall identity and access management problems while they are also implementing Web Services across the enterprise and among business partners.

Copyright, Trademark Notice, and Statement of Opinion

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