



THE DAWN OF DATA-LEVEL NETWORKING

THE NEXT INTERNET BOOM:
INTERNETWORKING XML AND WEB SERVICES

Forum Systems, Inc.

BOSTON, MA
95 Sawyer Road, Suite 110
Waltham, MA 02453

SALT LAKE CITY, UT
45 West 10000 South, Suite 415
Sandy, UT 84070

TOLL FREE
1-866-333-0210

www.forumsystems.com



TABLE OF CONTENTS

THE CHANGING FACE OF THE WORLD WIDE WEB	3
WEB SERVICES ADOPTION	4
▶ AMAZON.COM WEB SERVICES: NETWORK EFFECT IN ACTION	5
INTERNETWORKING: SAME GAME, DIFFERENT RULES	6
▶ DIGGING DEEPER INTO THE EXTENSIBLE MARK-UP LANGUAGE (XML)	9
▶ TECHNOLOGY SHIFTS: NECESSITY IS THE MOTHER OF INVENTION	15
THE FUTURE OF THE FIREWALL	16
▶ WEB SERVICES FIREWALLS	17
CONCLUSION	21

AUTHOR:

Walid Negm
Vice President, Product Marketing

Forum Systems Inc.

Release Date: 06/01/2004

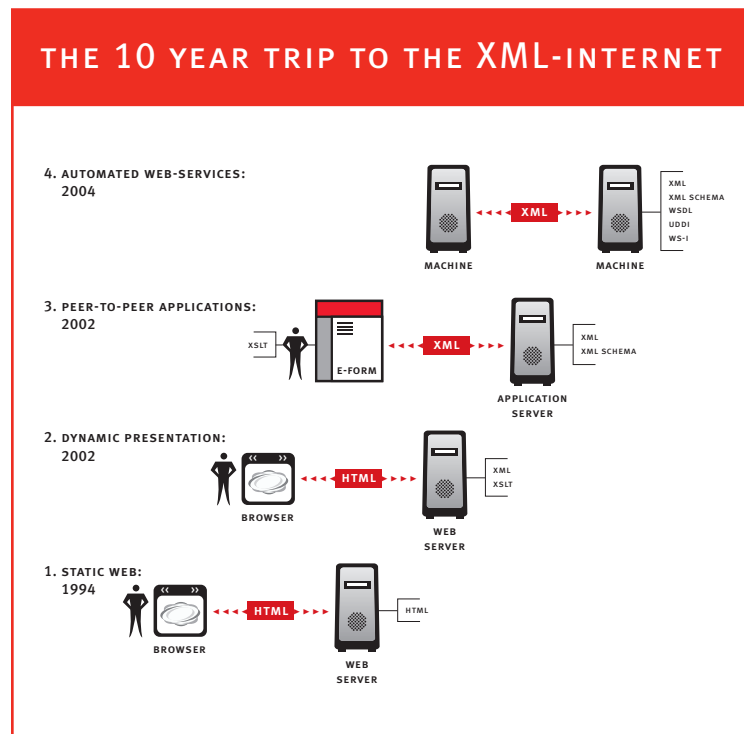


THE CHANGING FACE OF THE WORLD WIDE WEB

In the future, every Web site will be a Web service that automates machine-to-machine interactions. This computing paradigm will enable information to be exchanged freely without the need for time consuming, manual human intervention. While seamless information sharing is a primary deliverable of Web services, small and medium sizes companies can also level the business playing field as they are able to participate in trading relationships that previously would have been prohibitive to join due to cost and complexity.

The open and standards-based technologies upon which Web services are built permit companies to describe, and then share information without having to create rigid dealings with business partners. Web services make this possible by eliminating the need for expensive and proprietary approaches to information sharing like Electronic Data Interchange (EDI), Virtual Private Networks (VPNs) and Value Added Networks (VANs).

While HTML (Hyper Text Markup Language) is the most widely used format to describe the human-centric Internet, XML (eXtensible Markup Language) will form the foundation of an analogous machine-centric, XML-Internet. In fact, a migration from HTML Web sites to XML Web services has been occurring for the past decade (see figure below) and XML will soon usurp HTML as the lowest common denominator to interconnect networks, devices and software.



The XML-Internet is attracting the attention of executives and business managers across industry sectors and global governments as they seek to make effective and competitive decisions - faster. This whitepaper examines how the widespread adoption of XML is changing the roles and responsibilities of the enterprise network that has become the primary medium for conducting business and communications. This paper defines "data-level networking" as an expected evolution of today's IP (Internet Protocol)-centric network and one that explicitly recognizes XML data and Web services. This paper also discusses the continued development of network security and internal controls that govern and harness the flow of shared information.



WEB SERVICES ADOPTION

Web services are Internet technology-based, "machine-to-machine services" built on dynamic component connectivity and interoperability using open standards, including Internet Protocol (IP), Simple Object Access Protocol (SOAP), and Web Services Description Language (WSDL) - IDC

In business terms: Web services promote interconnection of value chain participants by: (a) streamlining and simplifying business processes, (b) reducing the need for unnecessary human intervention, and (c) cutting the costs of conducting digital transactions.

BENEFITS OF WEB SERVICES: EXPAND MARKETS, INCREASE EFFICIENCY AND REDUCE COSTS

- ▶ Streamline operations and connect systems easily with those of multiple business partners
- ▶ Automate and consolidate business processes
- ▶ Increase presence on the Internet and reduce reliance on call centers or paper-based information
- ▶ Improve customer satisfaction by capturing or providing timely information
- ▶ Redirect staff to more productive tasks, thereby driving operation costs down
- ▶ Leverage third-party information

Source: IDC

As the economy continues on its course of recovery, the need to increase productivity, decrease costs and increase revenue has become a high priority for companies. The most prominent way to reduce costs is to discard inefficiencies, and in particular, leverage existing information assets. The reuse of existing investments typically means integrating disparate databases, exploiting data warehouses and extending on-line transaction processing applications - all to better synchronize and coordinate business processes that are working as ineffective silos.

While the need for internal application and external business-to-business integration is not a new trend, the methods of integration have historically proved complex and costly. For larger companies, mergers and acquisitions have yielded a hodge-podge of custom applications and heterogeneous e-business suites (Customer Relationship Management, Supply Chain Management, Business Intelligence, etc.); each with proprietary interfaces and data formats. The effective sharing and cross pollination of information held captive to vendor-specific and unwieldy Application Programming Interfaces (APIs).

For smaller companies, business integration typically means high costs associated with setting up point-to-point relationships and technology hook-ups. Indeed, for small and medium sized companies to participate in electronic data exchanges traditionally requires the implementation of EDI (or a proprietary equivalent mechanism) and the handling of incompatible data formats. The resulting



“integration” is mainly webified applications riddled with manual workflows that are the equivalent of swivel chair integration – personnel having to enter and re-enter data into multiple systems.

Web services take business integration to a new level using open standards and vendor-neutral technologies including UDDI (Universal Description, Discovery and Integration), SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), and the foundation building block XML. Using these standard technologies, communicating parties (machines, software and systems) are able to speak a common language that preserves the meaning of information such that it can be interpreted without confusion. Web services technologies remove the need for point-to-point connectivity by promoting scalable business interoperability through well-known protocols of HTTP (Hyper Text Transport Protocol) and SOAP.

Today, there are numerous XML document “Schemas” and functional taxonomies that are a semantic database of common meaning between industry verticals and groups. For example, by standardizing what an insurance policy looks like, a variety of insurance providers can compete for a client’s business and even work together to offer a personalized policy. Semantic information regarding XML-formatted documents are being defined by organizations such as OASIS, Accord, SWIFT, RosettaNet and eBXML to name a few.

AMAZON.COM WEB SERVICES: NETWORK EFFECT IN ACTION

As smaller companies begin to adopt WSDL and UDDI, they will drive value upon value and ignite a network effect similar to the early days of the World Wide Web.

Today, large companies are generating return on investments through internal and external systems integration and in particular, looking at out-sourcing (application and business processes) to cut costs and increase efficiency. Using WSDL and UDDI, large enterprises are able to establish unique “business platforms” where software becomes a utility.

Companies like Amazon, Microsoft and eBay are creating business platforms in the areas of electronic commerce, portal communities and content distribution. When enough customers and service providers are participating in these business platforms, the formation of new markets is sparked. As this vision plays out in a mainstream manner, small and medium businesses will have dynamically and “on-demand” integration into business platforms from the mega vendors. As they start to receive competitive pricing and product selection, new value will be generated.

Amazon.com is a good example of the success of Web services for business. The Amazon.com Web Services (AWS) program was established in 2002 and as of 2004 the number of participating developers reached 50,000. AWS exposes the Web retailers’ core e-commerce platform as an adjunct marketplace to the human-centric shopping experience.

AWS merchants use Web services technologies including XML, SOAP and WSDL to gain access to selling functionality such as listing themselves and their products, searching for product pricing and retrieving inventory information. Amazon sellers increase revenues as they better price their products. Today, these merchants would have to pull data from individual web pages, parse the underlying HTML and essentially “screen scrape” a product catalogue for targeted information.



Amazon benefits not only from an additional revenue stream but also by increasing the selection of products within its marketplace.

"Web services are a now thing, they're not even a next-year thing. It's time to get going with them if you haven't already." - Amazon.com Program Manager

INTERNETWORKING: SAME GAME, DIFFERENT RULES

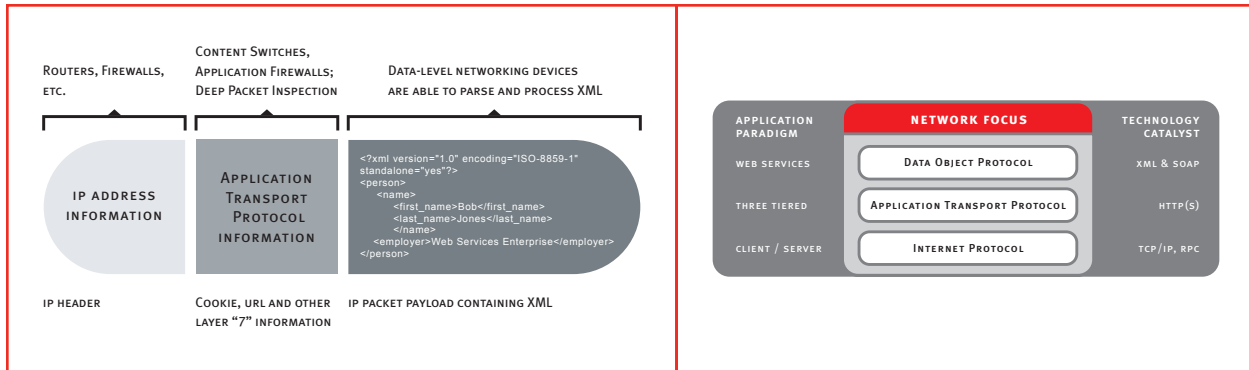
"By 2005, Web services will have reopened 70 percent of the attack paths against Internet-connected systems that were closed by network firewalls"— Gartner Group, Security Strategies for Enterprises Using Web Services, May 2003

In the non-XML Internet, Web browser-based applications rely on the popular HTTP (Hyper Text Transport Protocol) application transport protocol to transmit requests and their unstructured parameters, through a URI (Universal Resource Locator) and HTTP Headers with the results presented as HTML formatted web pages. This is a familiar process that we see everyday by requesting a URI such as www.amazon.com/listing in the Web browser address location bar.

Today's networking vendors make this request/response model possible by specializing in areas such as load balancing HTTP traffic, encrypting TCP/IP (Transport Control Protocol/Internet Protocol) communication using SSL (Secure Sockets Layer) or preventing malicious web site attacks such as web page defacement, denial of service attacks, 'phishing' and other forms of web site hacking. Application firewalls are able to inspect HTTP traffic for further clues about positive and negative Web browser user behaviors to permit or deny requests.

With the adoption of XML as the foundation for machine-to-machine communication, security decisions in particular cannot be based on Web browser technology because the critical factors required to take decisive action are dependent on the content of the XML message itself. This content can take the form of a SOAP-encoded purchase order or an XML-tagged engineering design document. Trojans and other malicious content may reside within these XML documents. Moreover, the privileges to read, write or execute actions on SOAP-based requests (vs. Web browser requests) can only be effectively verified when the "purpose of the transaction" is first recognized. This can only be accomplished by knowing the nature of the request, hence, distinguishing between a 'purchase order entry' and a 'purchase order search'.

Firewalls are blind to XML because they cannot perform the necessary context-sensitive processing of structured XML data. While traditional load balancers, switches and firewalls perform a relevant role in the network topology, they are unable to handle XML and Web services traffic. Even so-called "content switches" merely peer into the surface of the communication protocol and not the actual cargo. Today's networking products are focused on the Internet Protocol or Application Transport Protocol (see figure below). They are oblivious of whether a financial transaction, engineering document or executive decision is being exchanged. The idea of knowing that a business transaction of a specific value (e.g. \$100,000) is being moved across the network and that it should take precedence over a phone call made to a masseuse is ultimately what is sorely lacking.



THE DATA-LEVEL NETWORKING CONCEPT: center on the information that is moving across the network, and in particular, the increasing reliance on structured XML and Web services. Data-level networking moves the focus from the Internet Protocol and Application Transport Protocol toward the data itself. Data-level networking applies the principles of security, routing, switching, filtering and prioritization of structured XML data in order to get more value out of the business network and offload processing of XML from the application.

However, in order to “know” the purpose of a transaction or message, one must put the information being exchanged into context. The data-level network must understand a number of Web services technologies to protect, and then enhance the efficiency of XML-enabled and networked back-office business applications:

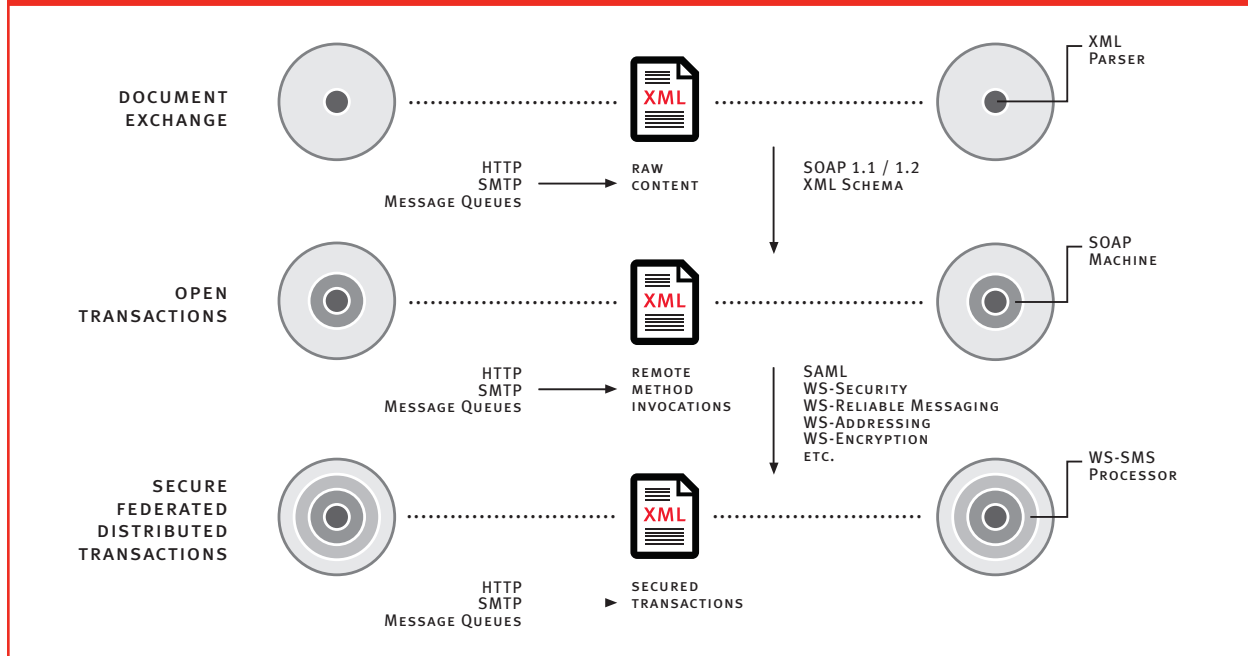
- ▶ **XML 1.0:** XML is the base standard upon which all other Web services standards are constructed. The hierarchical taxonomies that can be constructed with XML (written in XML Schema) provide a vendor-neutral mechanism that can power large-scale, e-business integration. (IDC)
- ▶ **SOAP:** SOAP is an XML messaging language that is used for exchanging information among peers in a distributed environment. It provides a standardized means by which XML data can be transported from one place to another. A SOAP document (called an envelope) contains a request for data or a response with the requested data. These envelopes are carried on HTTP, or other transport protocols, such as SMTP (Simple Mail Transport Protocol). A SOAP message is completely independent of the network protocol, the application, the platform, and the programming language that it is operating within. This independence is the key to the loosely coupled architecture of Web services. (IDC)
- ▶ **WSDL:** WSDL describes how specific services function. The service can be written in any programming language, such as Java or .Net, but the client accessing the service will only interact with the WSDL interface. (IDC)
- ▶ **UDDI:** As Web services move further outside the walls of an enterprise, there is an increased need for the equivalent of a yellow pages that enables requesters to find suppliers. UDDI provides this directory function by listing available Web services. Through UDDI, applications can be assembled on the fly. (IDC)



- ▶ **WS-Encryption:** Web Services Encryption is the XML syntax for representing encrypted XML data elements, including established procedures for decrypting this data. Unlike SSL, WS-Encryption allows the encryption of only the data that needs to be kept confidential. For example, only the credit card number within in a purchase order.
- ▶ **SAML:** Security Assertion Markup Language defines an XML/SOAP-based protocol that supports real-time authentication and authorization across federated Web services environments. The standard defines request and response messages that security domains use to exchange authentication, attribute and authorization information in the form of trust-assertion messages about named users and resources.
- ▶ **WS-Security:** Includes component specifications: WS-Digital Signatures, WS-Soap Message Security, WS-Trust, WS-Federation, WS-SecureConversation and WS-SecurityPolicy. WS-Security describes enhancements to SOAP messaging that provides quality of protection through message integrity, message confidentiality, and single message authentication. These mechanisms can be used to accommodate a wide variety of security models and encryption technologies.
- ▶ **WS-ReliableMessaging:** WS-Reliable Messaging describes a protocol that allows messages to be delivered reliably (regardless of transport) between distributed applications in the presence of software component, system, or network failures. WS-Reliability, promoted by the OASIS organization, also addresses the same problem. While message reliability is important, this does not preclude the requirement for transport and transactional reliability.
- ▶ **WS-Transactions and WS-Coordination:** A set of Web service interface definitions and protocols that support participant control and agreement on the outcome of distributed, multi-party interactions.
- ▶ **WS-Addressing:** WS-Addressing helps to define how Web services should be invoked, routed and replied to in a transport-neutral way. Rather than relying on HTTP to carry along the information necessary, these requirements and actions are explicitly defined in the SOAP Header.
- ▶ **Message Queues:** Allows the reliable transport of messages using a publish and subscribe application communication model. Vendor-specific solutions include: IBM WebSphere MQ, Tibco Rendezvous, Microsoft Message Queues, Java Messaging Service and others.



THE VALUE OF THE BUSINESS NETWORK IS THE CONTENT, NOT THE PACKETS



DIGGING DEEPER INTO THE EXTENSIBLE MARK-UP LANGUAGE (XML)

The previous sections briefly mentioned how network and application firewalls cannot understand XML, and therefore, cannot apply data-level networking policies of routing, switching, filtering, prioritization and security on structured XML data. While they may protect ports 443 and ports 80 over which HTTP traffic flows, they are simply unable to ascertain what is exactly flowing across these ports. Their goal has been to begin securing the communication and not the XML document payload.

It is the XML document that holds this payload, such as a hotel reservation, personal identification or instructions for a financial trade. XML documents are made up of a set of hierarchical nodes that clearly describe the content they enclose. The top parent node is referred to as a root node or element. It is usually labeled in a manner that describes the purpose of the document. The following is an example of an XML document that holds an employee record:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<person>
  <name>
    <first_name>Bob</first_name>
    <last_name>Jones</last_name>
  </name>
  <employer>Web Services Enterprise</employer>
</person>
```



The key strength in XML is that it was designed to better represent arbitrary, hierarchical data by lending structure to an otherwise potentially undistinguished piece of information. Consider this same employee record described using a word processor. While structure can be achieved using a word processor, text editor or spreadsheet, it remains hidden in proprietary and typically binary encoding that can not be readily shared. XML documents themselves, contain text as well as binary attachments that are packaged in the MIME specification. The default text encoding for XML documents is UTF-8, which doesn't differ from ASCII when only ASCII characters are used.

XML DOCUMENTS CAN BE THOUGHT OF AS HAVING A NUMBER OF LAYERS OF ABSTRACTION:

- 1 ▶ A binary data sequence representation
- 2 ▶ A contiguous character sequence
- 3 ▶ Segmented mark-up and character data
- 4 ▶ Hierarchy of logical artifacts including elements, attributes, character data, etc.

XML itself has been defined by the W3C (World Wide Web Consortium) as a structured representation with a set of attributes. To determine whether an XML document is "well-formed" (i.e. fully compliant with the W3C XML 1.0 standard), an XML document must pass, at a minimum, the following conformance tests:

- ▶ a valid XML document must have one (and only one) root element.
- ▶ elements which contain entries must possess both an opening and a closing tag. (In the case of an empty tag, which looks like this: `<example/>`, the tag is taken both to open and close itself in a self-contained manner. Its purpose is to save having to code `<example></example>` to preserve well-formedness)
- ▶ all attribute values must be enquoted
- ▶ tags may be nested, but may not overlap
- ▶ a form of the XML declaration `<?xml version="1.0" standalone="no" encoding="UTF-8"?>`

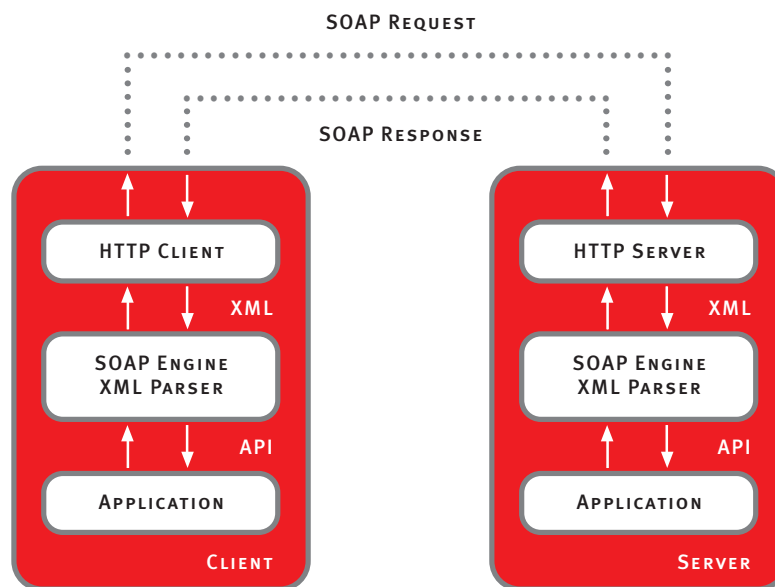
In order to accomplish these basic conformance tests, the content of the XML document must be scanned and separated into its textual parts with any associated binary data identified. This process is referred to as "parsing", and includes all of the steps to prepare an XML document for further XML processing (security or otherwise).

The XML parsing process breaks apart the stream of characters within an XML document into words, phrases, and other useful components. It then assigns attributes to tokens such as: word, delimiter, operator, verb, noun, noun-phrase, punctuation, noise-word, etc. XML parsers must recognize the difference between the mark-up data and the character data that make up XML-based information. This mark-up data model is represented as a tree structure that gives structural, logical and semantic details. Note that reading XML content without the XML validation step circumvents the primary objective of putting XML documents into context (e.g. when digitally signing a document).



By reading an XML stream and looking for the <xml? tag, one can assume that the input is an XML document. However, the document may be malicious if it is confirmed not to be an XML 1.0 document. This sanity check is very important for screening against hackers sending corrupt data that can disrupt business transactions which are programmed to rely on well-formed XML messages. The degree to which a message is well-formed also extends to validating messages against specific XML Schemas.

While XML parsing is the first step in putting XML document payloads into the necessary context for applying a routing or security policy, the story of efficient XML data exchange does not end with XML 1.0. The SOAP (Simple Object Access Protocol) was created as an abstraction layer on top of XML to standardize application message communications. SOAP specifies how remote procedure calls will be accomplished among applications using an XML encoding. SOAP compliant requests/responses are analogous to underlying TCP/IP connections.



While knowledge of XML 1.0 is important, it is not sufficient to interpret Web service requests and responses. The instructions required to perform content-based routing, SOAP security and other XML intermediary processing on XML documents are resident in the SOAP document Headers. The W3C defined SOAP 1.2 as an XML document with SOAP Headers and a SOAP Body that must be parsed and interpreted for processing.

Here is the same XML document previously displayed within a SOAP-formatted message along with security information written in the OASIS approved Security Assertion Markup Language (SAML). The SAML markup contains security relevant information that can be read by a data-level network element in order to make access control decisions.





```
<?xml version="1.0" encoding="ISO-8859-1"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <wsse:Security
      s:mustUnderstand="1" xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/07/secext">
      <saml:Assertion AssertionID="localhost.1085658264722.24655070" IssueInstant="2004-05-
        27T11:44:24.722Z" Issuer="http://forumsys.com/sentry/saml/issuer" MajorVersion="1"
        MinorVersion="0" xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
        <saml:Conditions NotBefore="2004-05-27T11:44:24.722Z" NotOnOrAfter="2004-05-
          27T11:45:24.722Z"/>
        <saml:AuthenticationStatement AuthenticationInstant="2004-05-27T11:44:24.722Z"
          AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:unspecified">
          <saml:Subject>
            <saml:NameIdentifierFormat="urn:oasis:names:tc:SAML:1.0:assertion#emailAddress">bobjones@
              forumsys.com</saml:NameIdentifier><saml:SubjectConfirmation><saml:ConfirmationMethod>urn:
                oasis:names:tc:SAML:1.0:cm:sendervouches</saml:ConfirmationMethod></saml:SubjectConfirmati
              on></saml:Subject></saml:AuthenticationStatement></saml:Assertion></wsse:Security>
          </s:Header>
        <s:Body>
          <person>
            <name>
              <first_name>Bob</first_name>
              <last_name>Jones</last_name>
            </name>
            <employer>ACME</employer>
          </person>
        </s:Body>
      </s:Envelope>
```

XML PARSING: BREAKING UP AN XML DOCUMENT INTO ITS LOGICAL CONTENT:

- ▶ XML Declaration or Text Declaration
- ▶ XML Schema and Namespace Declaration
- ▶ Processing Instructions
- ▶ Comments
- ▶ Text (Character Data)
- ▶ Elements and their Attributes
- ▶ Message header meta data
- ▶ Binary attachments
- ▶ Embedded security context



Another specification that must be understood by a data-level network element is the Web Services Description Language (WSDL). WSDL documents are used to describe the requests and the corresponding responses of communicating parties. They are essentially the rules of engagement between a Web services consumer and a Web services producer describing everything from data types, to communication patterns (request/response, publish/subscribe, etc.) and operations published.

The following is an example of a WSDL document that describes Google's Web APIs for searching the Google Engine automatically using SOAP Web services:

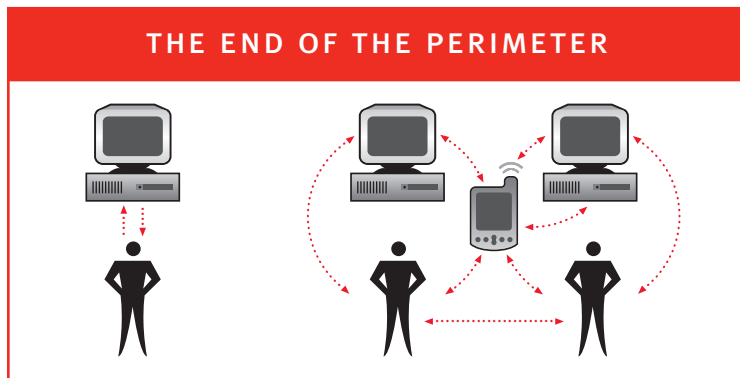
```
<?xml version="1.0" ?...>
<definitions name="GoogleSearch" targetNamespace="urn:GoogleSearch"
  xmlns:typens="urn:GoogleSearch" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
<types>
<message name="doGetCachedPage">
<message name="doGetCachedPageResponse">
<message name="doSpellingSuggestion">
<message name="doSpellingSuggestionResponse">
<message name="doGoogleSearch">
  <part name="key" type="xsd:string" />
  <part name="q" type="xsd:string" />
  <part name="start" type="xsd:int" />
  <part name="maxResults" type="xsd:int" />
  <part name="filter" type="xsd:boolean" />
  <part name="restrict" type="xsd:string" />
  <part name="safeSearch" type="xsd:boolean" />
  <part name="lr" type="xsd:string" />
  <part name="ie" type="xsd:string" />
  <part name="oe" type="xsd:string" />
  </message>
<message name="doGoogleSearchResponse">
<portType name="GoogleSearchPort">
<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">
<service name="GoogleSearchService">
  </definitions>
```

Only by having prior knowledge of WSDL documents can a network element be able to correlate what is correct and incorrect XML traffic patterns, and for example apply access control policies on to request and corresponding responses. WSDL documents are a key part of the intelligence of a data level networking element.



EVOLUTION OF THE ENTERPRISE PERIMETER

The enterprise perimeter is the barrier between a company's internal network and the external public Internet. But in the extended enterprise, the boundaries of partners are blurred with closer coordination of design, development and marketing activities giving way to stable collaborative relationships. The traditional view of network perimeter security falls short of providing the granularity to control sensitive, shared information and application resources.



Creating a trusted and un-trusted boundary using today's network security solutions is not sustainable as the enterprise continues to be deluged with edge access devices and wireless access points. Un-trusted information is naturally being tunnelled into corporate networks that use encrypted communications channels, such as VPNs.

There is a fundamental shift in the focus of the enterprise perimeter from an internal-to-external boundary to one of an extended enterprise with the following characteristics:

- ▶ A burgeoning number of requests for enterprise access is chipping away at the perimeter. The old "hard-crunchy outside and soft-chewy inside" security paradigm is not providing adequate protection, and so attention is shifting to the application itself as the focus for security efforts.
- ▶ Traditional perimeter security protects against threats coming from outside the network, but the threat from internal users - current or recently terminated employees - is also substantial. Web technology can act as a conduit into back-end corporate resources. Further complicating this picture is the rise of "extranets", which give business partners access to corporate computing and data resources.
- ▶ Enterprises have deployed firewalls on corporate networks to protect sensitive data platforms (e.g., legal, human resources, and accounting systems), but there are often so many holes poked in the firewall that its security is compromised and manageability is almost non-existent.
- ▶ The continued growth of outsourcing - transferring network, platform and physical security controls to a managed service - only increases the number of people who seek access into enterprise back-office systems.

(SOURCE: BUSINESS COMMUNICATIONS REVIEW)



TECHNOLOGY SHIFTS: NECESSITY IS THE MOTHER OF INVENTION

As the first-line of defense for most companies, the corporate firewall has been the cornerstone of network security; isolating a trusted internal network from an un-trusted public Internet. Firewall technology has taken three significant evolutionary steps:

- ▶ **PACKET FILTERING:** Operates at the network level (Layer 3) of the OSI (Open Systems Interconnection) model. Packet filtering examines each IP packet, and determines whether the packet is dropped or allowed to proceed. This technology does not inspect the associated content as the packet scanning only focuses on packet headers.
- ▶ **STATEFUL INSPECTION:** Operates at the network level (Layer 4-7) of the OSI model by inspecting IP header information without the need for separate application-specific protocol support.
- ▶ **APPLICATION PROXY:** Operates at the application level (Layer 7) of the OSI model using protocol-specific proxy functionality to examine the IP packet header as well as the application transport protocol data. Since there are multiple application transports (e.g., HTTP, SMTP, FTP etc.), there has to be proxy functionality for each of these application-specific protocols.

Unfortunately, these firewall technologies do not address the network security challenges of an enterprise with no real trusted boundary. NIPS (Network Intrusion Prevention Systems), application firewalls and network firewalls are specifically designed to filter and provide access control at the lower levels of the OSI Stack and cannot scrutinize the content beyond Layer 7 which is made up technologies including XML, SOAP, UDDI, SAML, etc.

Network and application firewalls must evolve to address the following requirements:

1. Perimeters no longer define trust relationships: The insider threat is as great, if not greater, than a known outsider, when you consider the risks that come with increased business outsourcing, subcontracting and temporary employment. Add the ever-present issue of disgruntled employees, and the significant threat posed by the insider is apparent. The information that is flowing within an organization becomes a composite of collaborative and shared data elements, each with its own security context arising from autonomous, yet related, trust relationships.
2. The business value is in structured XML data: With XML and Web services, the number of data sources, application responders, service listeners and exposed information nodes are intertwined. There are convoluted maps of information sources that must be shielded from attack and consumers must be assisted in navigation across interconnected XML-enabled network nodes. The familiar business-to-consumer networking model breaks down in the machine-to-machine environment, and must be replaced with a data-level networking model that supports routing, prioritization, filtering and security functions on structured XML data.
3. Data authentication supersedes user authentication: There is a native limitation with today's firewall technologies in that they cannot apply authentication and access control on to SOAP messages. This becomes critical in a multi-point business transaction where authentication and authorization information are embedded into the data itself. This requirement is related to how network boundaries no longer define trust relationships, requiring that information be self-authenticating carrying it's own set of privileges and authorization attributes.



THE FUTURE OF THE FIREWALL

“Today’s firewalls constitute a Maginot line against network level attacks. They have succeeded so well that cyber attacks mostly will occur at the application level, where new protections will be required. Bottom line: from a security perspective, Web services are about moving application integration into firewall-evading tunnels.” – John Pescatore, Gartner Group

While technology re-deployment has always been a logical and cost-effective IT strategy to solve problems, it is clear that this tendency breaks down with Web services and data-level networking. Moreover, specifically relying on user-, application- or network-centric approaches to Web services security fails to meet even the most basic security requirements, and quickly leads to breaches, exploits and increased risks to daily business.

Network and application firewalls must evolve to meet the security needs of the enterprise, where information has to be inspected on the content level. This detail must be obtained from the SOAP Headers and XML content, not from the source IP, destination IP or application-layer protocols (see table below).

NETWORK AS WELL AS APPLICATION FIREWALLS LACK THE CONTEXT NECESSARY TO PROCESS AND SECURE XML OR WEB SERVICES:

1. Objects manipulated: The parsing and handling of XML objects deals with technologies including URLs, operations and messages. XML message flow control is required, not merely packet or protocol flow control
2. Purpose: While XML is important in recognizing the objective of a data-level networking device, it’s purpose is to dig much deeper into SOAP, relying on WSDL, UDDI and XML Schema for more context to perform filtering, access control, routing and other functions
3. Access Control: By interpreting the service request within a SOAP message and the necessary business context, privileges for access control at a granular Web service operation or message can be recognized. This is opposed to access control rules defined at the IP address, port and protocol level
4. Data-level policies: While Web services security is a primary concern in the XML-Internet, other functions such as routing, filtering and reliability must be implemented on XML and Web services traffic



A WORD ON SSL:

- ▶ SSL clearly breaks down in a multi-point transaction where the end-point for the Web service is the one authenticating the consumer and not the intermediary. The break-down occurs in two ways: (1) the intermediary does not gain access to even partial payload information to determine if there are any threats, and (2) SSL client and server authentication cannot be performed mutually by the end point and the originating client; thus, the security context is lost unless it is a point-to-point communication
- ▶ When SSL client credentials are employed, they are not bound to the message (payload, document or transaction) in order to determine access privileges on a SOAP message operation request. The only method for achieving this level of granular authorization is for the authentication server to have a repository of WSDL documents and associated entitlements
- ▶ SSL lacks the audit trail for business transactions, both in terms of detailed logs as well as digital signatures and archival of messages for non repudiation purposes
- ▶ While SSL is primarily used for encryption of the communication channel, the administration infrastructure is not geared for authentication of business transactions.

WEB SERVICES FIREWALLS

“Web services Firewalls are unlike traditional network and application firewalls because they are “using an in-depth knowledge of the Web services, service requesters, and message content” – John Pescatore, Gartner Group

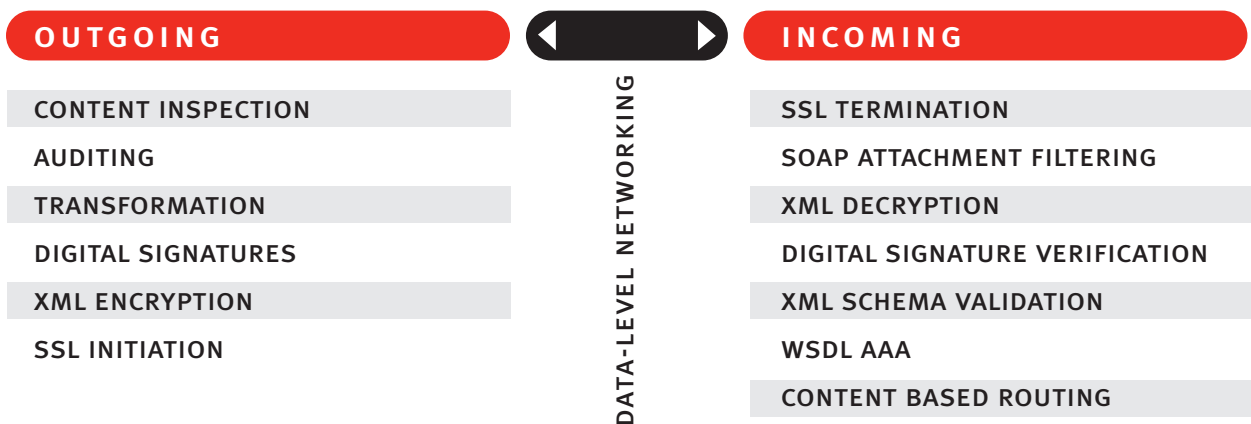
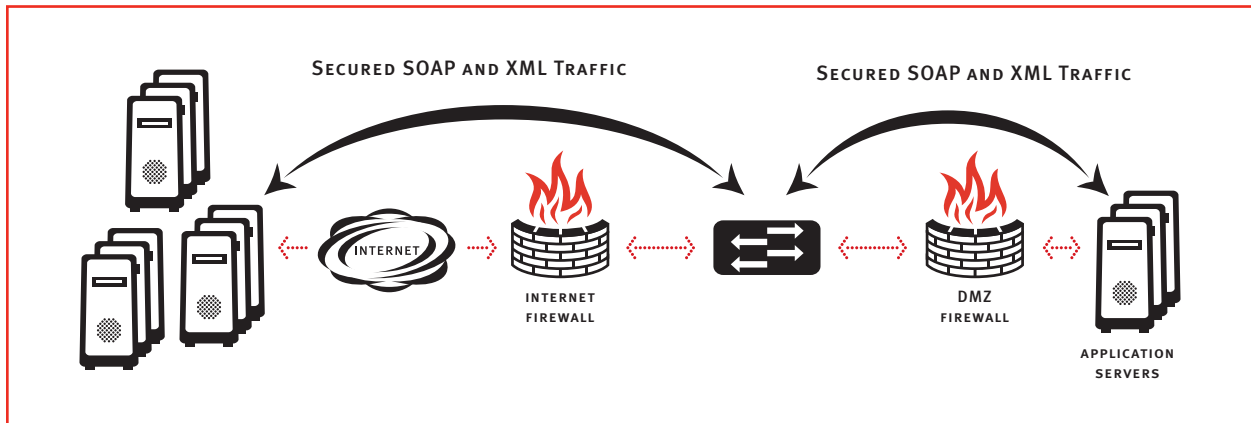
XML adoption changes the roles and responsibilities of the network because today’s firewalls cannot understand the meaning of XML; they let these messages pass freely through Web ports. We know that network and application firewalls can check for RFC protocol violations (i.e., making sure that the requests themselves are syntactically correct). However, they are unable to detect whether XML messages are contextually incorrect or erroneously formatted.

Moreover, the reliance on SSL (Secure Sockets Layer) technology to encrypt the communication channel keeps content obfuscated until it is delivered to the application logic. The network does nothing to stop hazardous or unauthorized content from entering databases, workflows and business applications. Only by comprehending the content, as well as the context, of the XML data that is flowing through the network can you effectively secure, control and govern information flows.



A Web Services Firewall is the next logical evolution for enterprise network security that incorporates:

- ▶ Data-Level Networking Technology
 - a. Can manipulate and control the actual XML ‘conversation’
 - b. Makes security decisions by inspecting the payload, and not just the network packets
 - c. Ensures interoperability of data exchanges
- ▶ Message-Centric Security
 - a. Treats all network traffic as un-trusted – internal and external
 - b. Applies authentication to each message flowing across the network
 - c. Granular data authentication enables the enterprise to permit data flows – such as Simple Object Access Protocol (SOAP) messages
- ▶ XML Web Services Policies
 - a. Native XML data processing tasks
 - b. Implements multi-faceted enforcement policies
 - c. Expands with the growing content layer: XML, SOAP, WSDL, UDDI, WS-Addressing, WS-Security, etc





Web services security breaches (malicious or accidental) rapidly penetrate deep into mission-critical business logic and back-office systems. They do not simply reach the Web or application server and stop. It is the role of the Web Services Firewall to not only protect a Web site, but also to secure all the Web services listeners and applications that make up the new extended enterprise.

PROTECTING THE WEB SITE :

ATTACKS	DESCRIPTION
URI PARAMETER TAMPERING	By tampering with the URI strings that are sent as part of a browser request, a malicious user can gain access to databases, for example. A typical URI can be injected with a SQL statement call which retrieves passwords and other account information.
BUFFER OVERFLOWS	This is a classic attack method that relies on a malicious user sending illegitimate data. This is typically executed in large amounts, or at a high frequency, in order to overload the target system. By doing this, the receiving system is essentially starved of memory. The most common server architecture weakness is to allow data that is stored in one section, called a “buffer”, to overflow into another reserved area, called a “stack”. The data that lands in the stack is then incorrectly executed by the CPU.
CROSS SITE SCRIPTING	This technique involves inserting code into interrelated Web pages. This code is then unknowingly executed by innocent users, and private and personal information is stolen to perform illegal activity.
COOKIE POISONING	‘Cookies’ store user-specific information, such as passwords, account numbers and other Web page state information. These values can be modified to gain unauthorized access.
DIRECTORY TRAVERSAL	This involves accessing a browser page by bypassing authentication, or the strictly pre-defined hyper links. Malicious users can access URLs that contain sensitive information, or view information that is not public and requires sign-on.





PROTECTING THE WEB SERVICE:

ATTACKS

DESCRIPTION

WSDL SCANNING

Since the WSDL document includes all of the operations that are available to the consumer, it is straightforward for a hacker to run through all of the operations with different message request patterns until a breach is identified. This “knocking on every door until one opens” approach is usually effective when poor programming practices are employed or simply the result of operations that were excluded from published WSDL documents yet are still up and running for some reason. This latter point is an oversight that will occur more often within global enterprise deployments of Web Services with weak central access point enforcement.

SOAP COMMAND AND SQL INJECTION

Structured Query Language statements are core to manipulating database records. Application Servers construct SQL instructions from information gathered by the requests from clients. These requests can contain SQL statements as well as specific fields that are formatted in a manner as to illegitimately retrieve, update or insert information in the database. Statements like this should not be in SOAP messages and must be caught and sanitized prior to arrival at the Application Server.

XML SCHEMA POISONING

XML Schemas provide formatting instructions for parsers when interpreting XML documents. Because XML Schemas describe necessary pre-processing instructions, they are susceptible to poisoning. An attacker may attempt to compromise the XML Schemas in its stored location and replace it with a maliciously compromised facsimile. DoS attacks against XML grammar are straightforward if the XML Schema is compromised. In addition, the ‘door is open’ to manipulate content if data types are compromised (e.g., changing dates to numbers when the application is performing arithmetic operations).

DENIAL OF WEB SERVICE ATTACKS

If XML is verbose in its self-description, then SOAP is extremely verbose. The first step taken by a Web Service producer after receiving a SOAP message request is to read through or parse the elements. The goal is to extract parameters, determine which method to invoke, insert content into a database or perform some other function. This basic operation is an easy target for the hacker when creating a DoS attack or degrading application performance. Similar to a network ‘ping of death’, a hacker can issue repetitive SOAP message requests to overload the Web Service. This type of network activity will not be detected as an intrusion because the source IP is valid, the network packet behavior is valid, and the HTTP request is well formed. However, the business behavior is invalid, and thus constitutes an XML intrusion.

SOAP ROUTING DETOURS

The WS-Addressing specification provides a way to direct SOAP traffic through a series of intermediaries by using XML tags that assign routing instructions. If an attacker overtakes one of these intermediaries, they may insert bogus routing instructions to point a confidential document to an unauthorized location where critical information can be stolen. This technique may also be used to execute a DoS attack by routing the document to a non-existent destination.

SOAP RPC PARAMETER TAMPERING

Since the parameters of an operation are described within a WSDL document, the hacker can again ‘play around’ with different parameter patterns in order to access unauthorized information. For example, by submitting special characters (or other unexpected content) the back-end implementation of the Web Service can be crashed, regardless of data validation.



XML EXTERNAL ENTITY ATTACKS

External entity references give SOAP documents the ability to build themselves dynamically by accessing URLs that point to third-party content. A third-party reference to XML input from an un-trusted or malicious source can coerce the Web Service to open arbitrary files or TCP connections, resulting in a DoS scenario. For example, a large number of entity expansions can overload the CPU, resulting in a DoS condition.

OVERSIZED XML AND SOAP PAYLOADS

XML parsing is directly affected by the size of the SOAP message. As a result, large amounts of CPU cycles are consumed when presented with large documents to process. A hacker can send a payload that is excessively large to deplete systems resources.

OVERSIZED XML AND SOAP PAYLOADS

This method of attack is accomplished by taking a perfectly legal SOAP document, and making a simple modification in order to deceive XML Schema validation. The hacker selects a node within the SOAP message and replicates it to create a deeply nested structure. For example, a line item of a purchase order can be copied over and over to force the Web Service producer to deplete CPU resources as it parses and validates each entry.

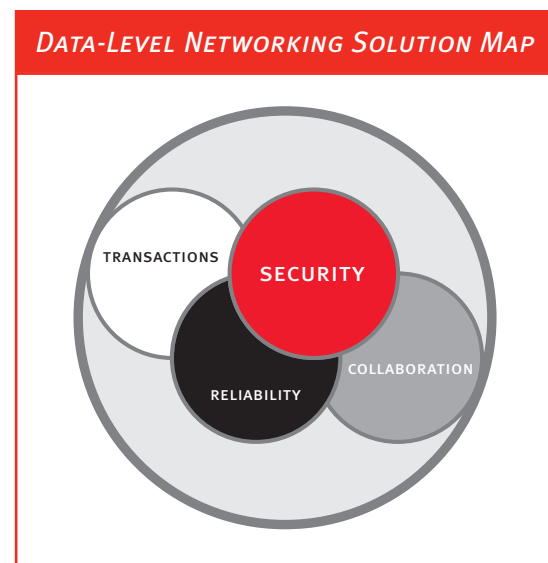
CONCLUSION

This paper has outlined the need for data-level networking above and beyond IP networking to control the increasing traffic of structured XML and Web services. Data-level networking has been defined to be vastly different from the traditional view of IP networking because it:

- ▶ Values the content being exchanged and not the packets.
- ▶ Moves the focus away from Transport Protocols towards the data itself.
- ▶ Applies routing, switching, filtering, prioritization and security on structured XML data.

As the automated Web takes shape to transform businesses and creating new value, it will be necessary for the network to continue on its path of increased intelligence. Data-level networking solutions will be able to recognize voice and rich multi-media described using structured XML; taking the business network to a new level of efficiently and reliability.

While the Web Services Firewall may be the first application of data-level networking, there will be other concerns relating to XML and Web services aside from security, including reliability, transactions and collaboration (*SEE FIGURE RIGHT*).





Enterprises must begin to overlay their existing IP-based network elements with data-level networking capabilities. Forum Systems is the Leader in Web Services Security and Data-Level Networking with a suite of products for both Threat Protection and Trust Management of XML and Web services.

Please visit <http://www.forumsys.com> for more information on the Forum XWall™ Web Services Firewall which is available as a stand alone product (Appliance, Software or PCI Card) as well as a module that can be layered on to traditional application firewalls to extend and enhance Web site security with robust Web services security.

© 2004 Forum Systems, Inc. All Rights Reserved

