FORUM SYSTEMS™
THE LEADER IN WEB SERVICES SECURITY

Anatomy of a Web Services Attack

A GUIDE TO THREATS AND PREVENTATIVE COUNTERMEASURES

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

AUTHOR:

Walid Negm
Vice President, Product Marketing

Forum Systems Inc.

Release Date: 03/01/2004

Introduction

The ever-expanding World Wide Web is evolving to a state whereby software agents and computer systems autonomously interact with one another to serve the information exchange needs of their respective organizations. As such, today's customer-facing applications rely heavily on Web Services technology to conduct transactions such as back-office financial trades and on-line e-commerce purchases. Web Services also enable enterprises and organizations to synergistically work together by automating end-to-end business processes for seamless collaboration without human intervention.

However, along with these enhanced information exchange capabilities comes significant security considerations and challenges. It will be up to IT executives and managers to recognize new information security threats posed by Web Services-enabled activity, and put appropriate security countermeasures in place to maintain order and control.

IT organizations must revisit how they protect critical applications and systems in the face of new security challenges posed by Web Services implementations by examining the following issues:

- J2EE, .Net, Web-based and legacy applications assume that authentication has filtered out the "bad guys". However, with Web Services accessible through the public Internet, these architectures and applications will be ill-equipped to detect new breeds of malicious application activity.
- Web Services that have been layered atop legacy applications will immediately expose underlying application business logic vulnerabilities, thereby raising the number of exploits exponentially.
- Data-level validation steps within client/server architectures exist primarily within the user-interface logic. This type of security is no longer relevant in Web Services-oriented architectures, and therefore needs to be updated.
- Packet-filters at the network level were never designed to recognize and diagnose the behavior of Web Services consumers. SOAP/XML-borne attack patterns require separate attention.

This paper covers the basics of Web Services "hacking", and brings to the forefront the need to guard Web Services-oriented systems from malicious activity by implementing information assurance measures for both Threat Protection and Trust Management:

- *Threat Protection* is primarily focused on protecting an organization's information content from vulnerability to attacks.
- *Trust Management* deals with the question, "Can someone be trusted to perform a particular action on a specific object?"

While the trust side of the security equation has received a great deal of attention in the world of security, the growing reliance on Web Services will raise security issues that go beyond traditional authentication processes. It remains important to know whether to be able to trust information, however; it will become increasingly imperative to verify that there is no threat-related activity associated with this information.

A New Type of e-Business Threat

The blueprint for Web Services communications is detailed in Web Services Description Language (WSDL) documents. An organization publishes WSDL documents in the equivalent of an on-line phone directory. Business partners can browse these documents and search for a particular Web Service.

The good news: WSDL documents are the handbook to a company's Web Services, and as such, they contain the recipe for interaction that yields improved communication between organizations.

The bad news: WSDL documents are also the handbook for attacking and hacking these same Web Services. Since a WSDL document contains explicit instructions on how to communicate with previously private applications, they can cause a serious security breach if the Web Services are compromised.

A network administrator must be given fine-grained control to:

- Lock down exposed Web Services
- Stop forbidden data from passing into the network
- Protect against XML-related vulnerabilities

This paper describes the fundamentals of Web Services communications, and the risks involved in not being vigilant against new security threats posed by this powerful set of technologies.

Understanding Web Services Interactions

The Simple Object Access Protocol (SOAP) is the Web Services specification used for invoking methods on remote software components using an XML vocabulary. It is used to describe the remote method being called, required parameters, return values, and any exceptions. A SOAP Envelope, Header and Body represent the basic structure of a SOAP message. The Body contains the message to exchange, and is usually a remote method call. The SOAP Header XML tag is optional, and if present, contains meta-information such as security, routing and processing instructions for intermediaries and end-points.

The technical architecture to facilitate a SOAP-based interaction requires the following components:

- **Consumer**
  Fundamentally, anything that can formulate a SOAP message can be considered a "consumer" of Web Services. SOAP clients, as they are technically termed, communicate with the Web Service (also known as "producer") via a common transport protocol that sends and receives SOAP messages. In order for SOAP to work, the SOAP client must have code running that is responsible for generating the SOAP request.

The Web Services Interoperability Organization (WS-I) has defined a set of standards to provide guidelines that ensure that the consumer and producer avoid application miscommunication. For example, SOAP was designed as a message format that would be independent of the transport layer. However, to enable optimal communication, WS-I recommends that the HTTP(S) protocol be used to move SOAP messages between consumer and producer.

The consumer generates a SOAP message according to a well-defined format that includes the action they would like to take according to what is offered by a given producer. These actions are named "operations" and represent remote method calls. The SOAP client transmits all the necessary parameters, and data, to complete the request. The producer sends back requested data, confirms a transaction or transmits an error message.

▸ Producer

The term "producer" is used to refer to the Web Services software that responds to the consumer's request, independent of the physical machine where the software is 'running'. A producer can offer one, or many, 'Web Services' – each of which contains any number of discrete functions that are offered to the consumer. A producer receives a SOAP message request over a transport, invokes the specified method, builds the response message and then sends the response (or a fault message) back to the client.

Any legacy application or architecture, such as J2EE or .Net, can be extended with a Web Services responder "stub". This stub interfaces with the application on one end by sending it the communication parameters, and then waits for the formatted result from the legacy application. It then translates the result into a standard SOAP message response that can be interpreted by the consumer. This is a common way to expose legacy applications/ systems, packaged applications and other mission critical systems that use Web Services. Many organizations rely on this approach to preserve their investment in existing application infrastructures.

▸ Transport

Although SOAP is transport independent, the WS-I recommends HTTP(S) as the standard application transport protocol for exchanging SOAP messages. This assures a scalable model that embraces new users without having to worry about low-level communication translation.

▸ WSDL Document

A Web Services Description Language (WSDL) document is XML-based, and details the Web Services exposed by an application. It contains all of the requirements to access a Web Service by describing the exact "recipe" for interaction. In doing so, the WSDL document articulates what the consumer can expect from the producer. As a result, the consumer and producer can properly interact to perform the desired task.

- **SOAP Message Response**

  A SOAP message response contains the results of the function call issued by the consumer. It is the response from the producer, and can include content such as attachments and status information (e.g., transaction confirmation). SOAP message responses can be as sensitive as the SOAP message request itself. For instance, the SOAP message response can accidentally (due to poor programming) include hints to the application architecture – for example, a basic parsing error can reply with the name of the parsing engine, or allude to the database and process used for authentication.

- **SOAP Message Fault**

  A SOAP message fault is a special SOAP message response that contains error information from the producer. It explicitly specifies a mandatory "fault" element. The server implementation detects any exceptions in the service and returns them to the client as a defined fault message code.

Hacking Web Services 101

The basic principles of security are authentication, authorization and auditing – simply termed "AAA". They exist across all computing models – client/server, Web Services and grid/utility computing. However, the application of these security principles on Web Services is different. For example, since SOAP messages can move across multiple legs of a communication link, a transport-centric authentication technique (such as SSL) is insufficient to assure data integrity and confidentially across the multiple "hops". The security context is lost as SOAP messages are routed between end-points.

Trust-related challenges associated with Web Services are readily addressed by protecting the SOAP messages themselves. This can be accomplished by applying encryption to elements within SOAP messages or by signing and then attaching a digital signature to the SOAP message request. This form of persistent, embedded, security is enabled by well-known standards such as Security Assertion Markup Language (SAML), WS-Security, Digital Signatures and XML Encryption.

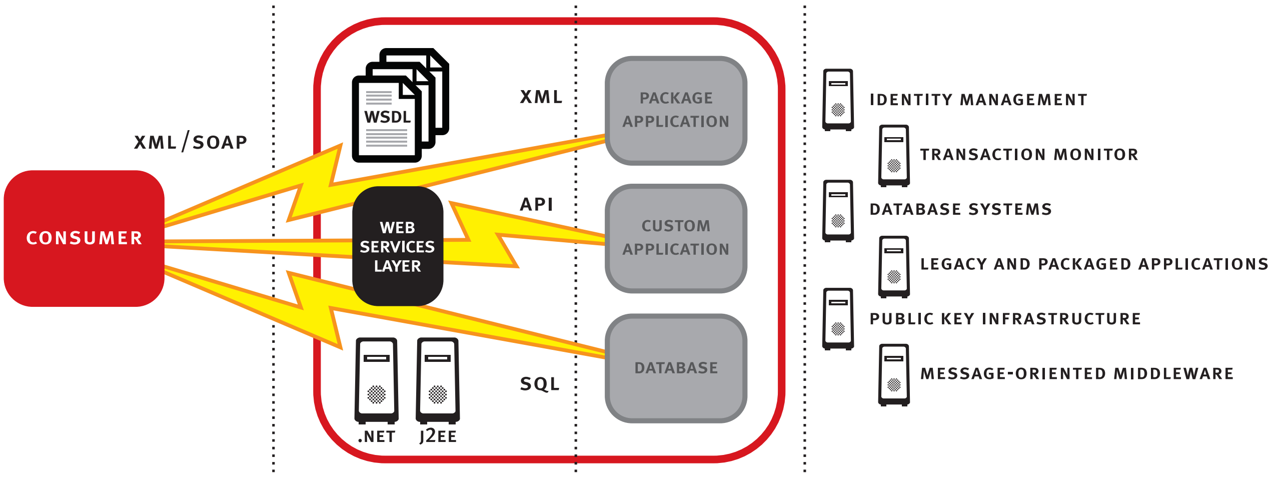
Authentication by itself does not provide adequate security for the threat protection side of the security equation. Denial of Service (DoS) attacks, information leakage and malicious activity can all arise regardless of authentication strength. Most attacks on traditional Web-based applications exploit security holes by sending requests that bypass authentication and results in website's being brought down. Buffer overflow attacks and submission of arbitrary commands to gain access to a remote machine are commonplace. Other more invasive attacks target databases, file-systems, programs, scripts, and HTML content. Well known Web Application security attacks include: SQL injection attacks, URL parameter tampering, URL directory traversals that attempt to bypass publicized URL's by accessing resources directly, and submission of URL strings that try to overwhelm the application by sending corrupt parameters (or queries) for unexpectedly large data sets.

All of the above scenarios target traditional Web applications by exploiting weaknesses in HTML-enabled custom, or packaged applications. However, hackers and other malicious users are getting smarter about Web Services. They are uncovering new techniques at the SOAP/XML data-level that bypass HTML and target weaknesses in Web Services programming, technology and architecture. Among other things, a hacker can undermine security and penetrate systems by relying on the expressiveness of SOAP to develop multiple attack patterns to achieve his/her goal. Web server security must now complement Web Services security in order to provide an adequate defense against these new types of attacks.

WSDL documents are the first point of weakness because they may be used as a “guide book” for vulnerabilities. Using this information, a security breach can penetrate deep into the business logic, and ultimately into back-office systems. The result can be exposure of mission-critical systems and applications to security attacks. The security challenge is magnified, as Web Services directly expose critical business systems across public connections. The goal of Web Services is to enable new ways of doing business in an automated fashion. For example, an order entry application can be tied directly to the invoicing application of a trading partner. This goes beyond simple application integration. It includes real-time movement of information and transactions. With Web Services, information knows no boundaries.

Since Web Services enable the full automation of business systems, the human verification element is not of consequence. The periodic virus scans, checks and balances and workflow approvals that are part of today’s human centric Web are not relevant in the world of automated Web Services-enabled applications.

The following diagram illustrates how Web Services enable a ‘straight-through-processing’ path into mission-critical systems.



VULNERABILITY PIPELINE: HACKER DISCOVERS WEAKNESSES + ATTACK PENETRATES WEB SERVICES LAYER + MISSION CRITICAL APPLICATION EXPOSED + ENTERPRISE INFRASTRUCTURE EXPOSED

The Vulnerability Pipeline

The following sections discuss in more detail how a malicious hacker can plan and launch a number of basic attacks against Web Services.

Vulnerability Discovery

WSDL documents are essentially advertisements of the functions that can be invoked by consumers, along with the expected results. Since most WSDL files are publicly available to consumers, they are not subject to privacy restrictions. Even WSDL documents that are seemingly secured will inevitably get into the hands of a malicious attacker.

The first step that a hacker would undertake is to embark upon a 'discovery' of system weaknesses and vulnerabilities. Given its open nature, the WSDL document is a great place to start. It provides a clear view into the application and details the application purpose, functional break down, entry points and message types.

Since the WSDL document contains an XML Schema, the next step would be to create a set of valid requests that can be submitted to the Web Service. By selecting a set of operations, and formulating the request messages according to the rules of the XML Schema, it is very easy to establish an interactive conversation with the Web Service. At this point, the hacker's goal is to begin manipulating a valid request to include illegal content. By doing this, it is possible to ascertain if Schema validation is turned on, and if so, what error messages are generally returned by the Web Service. Because Web Service application developers, by design, want to let the consumer know as much as possible about failures and exceptions, the hacker can simply study the responses to gain a deeper understanding of potential security weaknesses. For example:

- By knowing that strict validation is performed on one operation and not on another, a hacker can target the less secured operation to insert invalid data. This will enable more in-depth discovery that can lead to a security exploitation plan, such as a DoS attack.
- By noting how the system reacts to invalid data, a great deal can be understood about the system's underlying technology, such as the XML parser and validation engines. This information can then be used to exploit known vulnerabilities in vendor-specific XML technology stacks.
- The reaction to unauthorized access can include useful information regarding the existing authentication mechanisms. This information can then be used to bypass the system's security.

At this stage, the hacker is learning about the system and discovering weaknesses in order to formulate the most effective attack and exploit plan.

Probing Attacks

After a discovery phase the hacker may then proceed to probing the system for weaknesses and launch simplistic attacks. These attacks are characterized by 'brute force' cracking using combinations of letters, numbers and special characters in an attempt to determine which service requests result in a security breach. Examples include:

- *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 an oversight that will occur more often within global enterprise deployments of Web Services with weak central access point enforcement.

- *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.

Coercive Parsing

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 a hacker to create a DoS attack or degrade application performance. Other examples of coercive parsing include:

- *Recursive 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.

- *Oversize 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.

- *Replay Attack.* 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.

External Reference Attacks

Poor configuration, or improper coordination of internal resources can be readily exploited by hackers to create DoS scenarios or information theft.

- *External Entity Attacks*. External entity references give SOAP documents the ability to build themselves dynamically by accessing URL's that point to third-party content. A third-party reference to XML input from an untrusted, 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.

- *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 the XML grammar are straight forward 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).

- *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.

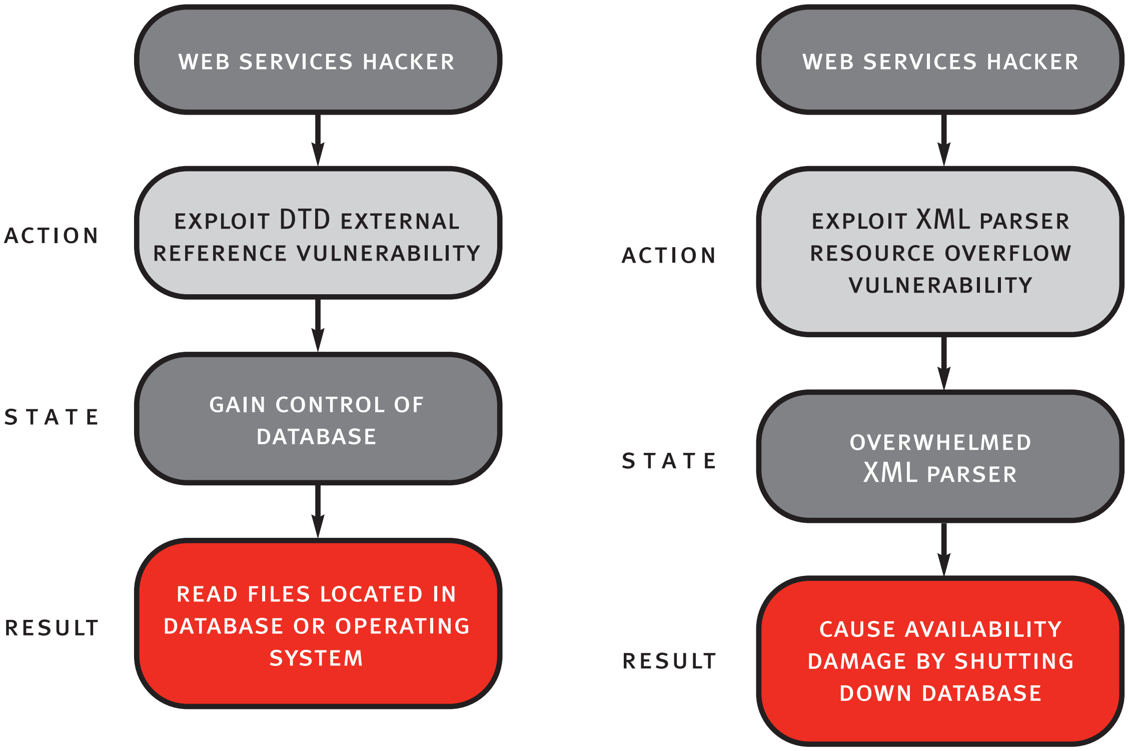
Malicious Content

Viruses, or Trojan horse programs, being transmitted within otherwise valid XML messages can easily cause havoc on Web Services applications. Binary attachments such as images, executables, and application-specific documents can all be modified to cause exceptions within the Web Services application. Furthermore, widely used tools such as spreadsheets can be easily corrupted to include macro calls, illegal content, etc. that crash applications, degrade performance, disrupt workflows or cause DoS scenarios.

SQL Injections

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. These statements should not be in SOAP messages and must be caught and sanitized prior to arrival to the Application Server.

This diagram illustrates two XML-related attack scenarios:


WEB SERVICES HACKER
ACTION
EXPLOIT DTD EXTERNAL REFERENCE VULNERABILITY
STATE
GAIN CONTROL OF DATABASE
RESULT
READ FILES LOCATED IN DATABASE OR OPERATING SYSTEM
WEB SERVICES HACKER
ACTION
EXPLOIT XML PARSER RESOURCE OVERFLOW VULNERABILITY
STATE
OVERWHELMED XML PARSER
RESULT
CAUSE AVAILABILITY DAMAGE BY SHUTTING DOWN DATABASE

Intrusion Prevention and Threat Containment

If Web Services are being deployed across an organization or enterprise, then appropriate trust-side and threat-side security measures must be put in place to protect against the onslaught of potential SOAP/XML security challenges, such as:

- Interception of and tampering with sensitive messages
- Forged client requests resulting in identify theft
- Forged server responses
- Spying
- Viruses and worms
- DoS attacks
- Sabotage and vandalism of critical systems

A comprehensive Web Services security strategy includes the basic, trust-centric “AAA” requirements mentioned earlier, as well as other techniques for threat containment that have been designed for Web Services, including support for:

- Data-level Authorization
- Exploit Immunization
- Data Validation

The following sections examine data-level authorization, data validation and exploit immunization. For IT professionals, these are critical tools for use in averting potential XML/SOAP security threats. For each threat, the challenges are discussed and practical remedies are presented.

Data-level Authorization

Web Services that are exposed through the corporate firewall are liable to be attacked if they are not properly secured with authentication and access control measures. Administrators need to create and enforce policies that guard against unauthorized use of Web Services by allowing only trusted/ authorized consumers permission to read, write or alter data according to predefined privileges. A Web Service operation, along with its request and response messages, will need security-related restrictions on invocation that can't be enforced at the transport or session level.

Today's Challenges:

- *Transport Independent Security.* Although SOAP messages are often sent over HTTP (as encouraged by WS-I Basic Profile) other transports can be used since SOAP is not dependent on the underlying communications layers. SOAP messages can be sent via Simple Mail Transfer Protocol (SMTP), FTP or Message Queues. Since HTTP is stateless and not entirely reliable, many organizations have no intention of using it to transport SOAP messages.

- *Granular Message-Level Access Control.* SSL certificates and HTTP username/password are used per application session to grant access to a specific set of screens or Web pages – usually based on roles, responsibilities and privileged groups. This type of application-level authorization is session-oriented, and has no notion of granular access control to "application functionality". As a result, individual SOAP/XML messages and WSDL files are not checked for inappropriate content.

- *Federated Security (i.e., Single-Sign-On).* A single SOAP message can traverse a number of different transports (e.g., HTTP and SMTP) in the context of a single transaction. SSL provides adequate confidentiality and authentication for a single point-to-point HTTP session. However, SSL does not guarantee that the proper security credentials travel across multiple hops of a transaction.

- *Document-Centric Authentication.* Web Services are, by their nature, message-oriented and lack the guarantee of a direct connection between service provider and consumer. Therefore, traditional connection-oriented security approaches are insufficient because a connection-oriented protocol cannot guarantee confidentiality between a message's initial producer and its final consumer.

- *Auditing.* Traditional security technologies do not provide an audit trail that documents when the session occurred. For example, SSL-based authentication will not capture the necessary information for tracking, thereby making it possible for a security breach to remain anonymous and possibly undetected.

Practical Solutions

Applying security to the SOAP message itself, rather than relying on the transport on which the SOAP message is sent, is essential for any meaningful access control security when using Web Services. If a message arrives over SSL or HTTP Basic Authentication, it should be parsed to recognize what operation is being requested and then the proper access privileges should be determined based on existing security policies.

- *WSDL Access Control Permissions*. The body of a SOAP message contains the requisite information to invoke a remote Web Service. This message payload contains the following items: name of the service called, location of the service, parameter name and data passed to the service. WSDL access control rules grant or deny access to any type of WSDL-based SOAP messages including the request, response as well as faults. Permissions are enforced based on privileges (i.e., roles, responsibilities and groups) that are established by the Web Services administrator. Other operational policies include publishing multiple restricted WSDL documents for a particular Web Service and restricted access to WSDL documents only by authenticated users.

- *Use of SAML for Authentication*. SOAP messages can cross multiple organizational bound aries, each requiring its own authentication process. It is better to rely on document-centric authentication credentials that use SAML. SAML allows for a single sign-on approach by embedding authentication status, as well as authorization permissions, within a particular message. As a result, authentication can occur seamlessly across multiple applications, and organizations. This single sign-on information provides access-control information to various Web Services by providing the authentication status of the message, the message's origin, what actions can be performed on the message and other important attributes.

- *Multiple Credential Types*. Supporting multiple security credentials (e.g. X.509 Cert, SAML assertions and WS-Security) is crucial to developing a scaleable Web Services deployment. While SSL remains a popular method to authentication it is no longer enough.

Exploit Immunization

Hackers are also becoming more sophisticated by relying on a variety of data-level techniques to carry their malicious code, application instructions and corrupt data. This problem is compounded by the fact that information technology vulnerabilities are also increasing as a new crop of exposure points and attack targets are revealed through the use of XML parsers, SOAP processors and WSDL-enabled applications. Exploit immunization involves enhancing business behavior analysis to mitigate these security risks. Administrators need to prevent Web Services-related security threats by looking beyond packet flows. They must also monitor data flows to ensure that all attacks are caught before they can cause damage.

Traditional network firewalls are designed to allow HTTP/SSL traffic to pass through without inspecting the data that is being carried across network ports. It is standard practice to tunnel application data, such as SOAP, through port 80 for HTTP and port 443 for SSL disguised as HTML traffic. When a network-centric firewall examines a SOAP request received over HTTP, it will only check whether or not the transport (HTTP) is valid according to the syntax of the HTTP (RFC) protocol. If it is valid, it will let the request pass through the firewall. However, network firewalls do not check whether a SOAP message is malicious, corrupted, or otherwise potentially damaging to an application.

Today's Challenges:

- *Firewalls are blind to XML.* While syntax checking of the transport protocol is important, the only way to detect malicious activity that is transported within the payload of messages is to scan the data and determine if the context of the information being presented represents a threat. The process of accurately inspecting SOAP messages requires precise knowledge of the format and the content of the XML data. This is referred to as "parsing" the SOAP or XML message. Traditional network firewalls do not have this functionality and are therefore inadequate for securing environments that use Web Services.

- *Data Flow vs. Packet Flow Monitoring.* The only way to distinguish legitimate Web Servicse traffic from illegitimate traffic is to go beyond the packet-level and monitor content-specific, Web Services usage. Scanning for patterns and anomalies in the data being exchanged is the only way to catch hackers or other malicious users who are trying to send inappropriate data. Monitoring traffic exclusively at the packet-level will not help detect Web Services-related security incursions.

- *Data-level Attack Prevention.* A typical Web Services message payload highlights a multitude of attack points, potential targets and known weaknesses. These include parsers, transforma tion engines, repositories and other components that interact with SOAP messages. Since message payloads can also contain malicious content, there are multiple opportunities for security breaches unless proper security is provided at the data-level.

- *Known and Unknown XML-related Vulnerabilities.* Web Services includes new components that have yet-to-be discovered exposure points, and vulnerabilities that are a function of the relative immaturity of the technology. Examples of these newer components include XML parsers, repositories, documents, client/server conversations and SOAP stack protocols (e.g. UDDI, WSDL).

Practical Solutions

Filtering network traffic by inspecting XML and SOAP messages, allows for intelligent decisions to be made regarding which data should be allowed to pass through a certain point on the network. By capturing and scanning XML and SOAP messages at the edge of the network, relevant and context-sensitive "data sanity checks" can identify and stop inappropriate data before damage is done to a system.

- *Anomaly Detection.* Web Services anomaly detection distinguishes valid from invalid SOAP messages by intelligently monitoring Web Services conversations for malicious/unusual behavior. Malicious SOAP/XML messages sent during normal hours are unrestricted. If the same messages defy normal behavior patterns by either passing through the system at odd hours, or if the messages arrive at an unusually fast rate, they are blocked.

- *Signature Detection.* An attack against an XML/SOAP parser or delivery of a malicious SQL command will have a specific "fingerprint" that is picked up by a Web Services-capable firewall. A signature detection database will detect known attacks before they can inflict damage to a system.

- *SOAP and XML Filtering*. Improper syntax, be it accidental or malicious, can result in significant damage that can span multiple systems. Targeted content filtering mechanisms for suspicious text, commands or parameter values can prevent such damage from occurring. The following are examples of improper syntax:

  - A data string that is too long.
  - A data string that contains illegal characters.
  - A data string that contains the word "password".

- *Service Provisioning*. By "cloaking" the WSDL document and restricting the content that is released, hackers are reduced to guessing about a system's security weaknesses. Cloaking and other vurtualization techniques involve using rules that allow, deny, expose or obfuscate specific WSDL-based operations and information. Examples include:

  - WSDL document must be re-published with limited Web methods exposed.
  - Specific operations must be denied access at specific times of day, or by IP addresses.

Data Validation

Data being carried via SOAP messages can be headed to multiple applications. Some may not even perform the most basic error checking. However, even when proper data validation is performed within the application-layer, the large volume of SOAP message flows still needs to be consistently "sanity checked" for malicious or inappropriate content. This, very rapidly, becomes a burden to the back-office system.

Today's Challenges:

- *Data Corruption*. While "data-type" validation may occur within the application, it is usually inadequate. Unless early checks against offending parameter values are performed at the net work's perimeter, applications are left unprotected and can be brought down due to repetitive data validation processing. Conducting data sanity checks early in the process (i.e., at the edge of the network) mitigates the high cost of fixing data corruption issues at the application level.

- *Validation Coverage*. Unless all transactions/messages are scanned for unusual content, sporadic erroneous information can cause system inefficiency or business process dysfunction (e.g., an erroneous "country" or "state" can cause delays in shipping). Since there isn't a simple/inexpensive way to initiate proper XML Schema validation within Web Services applications, all XML Schema checking (and other filtering) needs to be performed at the edge of the network before it touches an application's business logic. Traditional, network-centric firewalls are incapable of performing this level of content filtering.

- *External Entity Attack*. XML Schemas can contain references to external URLs that may contain malicious instructions and/or content. This manner of attack circumvents traditional data validation that is performed at the application layer, thereby leaving the system exposed.

- *Performance Degradation*. The verbosity and density of XML and SOAP creates a parsing performance issue that sometimes degrades application performance to unacceptable levels. XML Schemas may be used within application logic, however, this is not a scalable, nor reliable model to utilize XML Schemas. If not used at the network edge where attacks can be stopped before they get to the application, the use of XML Schemas can result in programmatic challenges (i.e., the need to maintain multiple XML Schemas that will provide adequate protection) and high developer costs (i.e., creating and maintaining multiple XML Schemas that will be required for each application).

Practical Solutions

The first line of defense is to validate SOAP messages against constraints, syntax and semantics. This checking should be performed at the edge of the network. This type of data validation can be accomplished by using XML Schemas to check for everything from SOAP syntax to more stringent content filtering prior to messages arriving at the application server.

- *Centralized Schema Validation*. Centralizing the validation process of requests and responses at the network perimeter removes this burden from application servers that already have a considerable workload. By doing this, peak application performance and high availability can be attained.

- *In-line Schema Validation*. Responses and requests must be checked for conformance and accuracy at multiple levels (i.e., SOAP standards, message syntax and semantic filters). Schema validation rules need to be enhanced beyond standard business process data type validation, to check for possible malicious activity that would not be detected by application-specific XML Schemas.

- *Schema Caching*. A secure repository of XML Schemas that is regularly refreshed and synchronized with the most current updates is the best way to avoid compromised, unapproved and questionable XML Schemas from being used.

- *External Reference Control*. Blocking external references and relying on sanitized pre-fetched (or approved) content when de-referencing URLs can avert most external entity attacks.

A Safe Passage for Web Services

With an ever-increasing amount of sensitive information being exchanged via the Internet, Web Services security is no longer “nice to have” functionality. It is now a primary business and IT consideration. In order to deliver comprehensive Web Services security, both sides of the security equation (trust management and threat protection) must be addressed.

Trust Management is Not Enough

Providing adequate threat protection is crucial to enabling comprehensive Web Services security. Furthermore, due to the nature of Web Services, threat-based attacks need to be addressed at the edge of the network instead of the traditional method of dealing with security at the application layer. The following are some reasons why it is important to ensure that proper security measures are taken:

- *Vulnerabilities Are Proliferating:* Web Services technologies, by their very nature, expose new areas where threat-based attacks can occur – e.g., XML parsers, Web Services abstraction layers, application logic and development frameworks (e.g., .NET, Java, Outlook, etc.)

- *Insider Attacks Pose an Enormous Threat:* A successful exploitation of a mission critical Web Service can result in, among other things, identities being hijacked, data being stolen and transaction processing systems being brought off-line.

- *Attackers Are Getting Smarter:* Legacy systems were not designed for the new breed of Web Services-related attacks. Web Services enable hackers to directly target systems unless adequate measures are taken. Hackers are rapidly learning how to effectively compromise Web Services technologies to carry out their attacks.

Web Services Firewalls

A Web Services-capable firewall is essential in managing the risk brought about through the ever-increasing use of Web Services technologies. This new generation of firewalls addresses the following issues:

- Enabling Widespread Use of Web Services
  - Without the protection afforded by the use of Web Services-capable firewalls, the significant power of Web Services can be used to attack an organization's systems. If this is allowed to happen, the benefits of using Web Services will be overwhelmed by the high costs of malevolent attacks.

- Legacy Systems Were Not Designed to Detect Hackers
  - Many legacy systems contain no business logic for detecting hacking, and are therefore considerably exposed to potential attack.
  - Content-based attacks will become the "avenue of choice" for the Web Services hacker. A Web Services-capable firewall is the only way to prevent these types of attacks from damaging systems that are seeking to take advantage of the many business benefits offered through the use of Web Services.

- The Risks to Business Can Be Dire
  - A successful exploit of a mission-critical Web Service can result in significant damage to an organization's systems, their business and their relationships with customers and partners.
  - Once Web Services-related attacks are successfully carried out an organization's systems, it can be too late to 'roll back' a malicious transaction that may affect multiple internal systems, or even partner systems.

- Early Prevention of Malicious Activity
  - Web Services-capable firewalls scrutinize SOAP/XML messages, and WSDL files, early in the physical network pipeline, thereby halting threats before they even get to critical systems.

Forum XWall™

Forum XWall™ Web Services Firewall is a turn key solution that IT departments can use to centralize and enforce critical aspects of Web Services-specific security at the edge of the network. Forum XWall™ is designed to complement network firewalls, which enables organizations to provide comprehensive security – Web Services and otherwise – to their information systems. Forum XWall™ can easily be added to existing firewall configurations to scan and block malicious Web Services traffic, thereby ensuring that critical applications are appropriately accessible and continuously available.

For more information on Forum XWall™ and other Forum Systems' products, please visit:

WWW.FORUMSYSTEMS.COM.