Executive Summary

Web Services is quickly earning its keep amidst its hype of providing extra functionality to computing environments. As with any new technology, this success comes with a level of increased risk. These risks are incurred explicitly when an organization deploys its own Web Services, and implicitly for any organization connected to the Internet and employing applications that have XML-awareness built in (every major enterprise software solution already has this capability).

This white paper describes the threat profile of a Web Services environment. It discusses the various techniques that may be used in an attack against the individual components using XML and soap documents.

The paper introduces the Top Ten Web Services Threats, a set of conceptual attacks that provide the most likely approach to compromising the Web Services environment. It then discusses ways to defend against these threat classes to protect a Web Services deployment.

Finally, this paper discusses how Forum Systems’ XWall firewall can be used to effectively protect against these attacks.

About Spire Security

Spire Security, LLC conducts market research and analysis of information security issues and requirements. Spire provides clarity and practical security advice based on its “Four Disciplines of Security Management,” an operational security model that encompasses identity management, trust management, threat management, and vulnerability management. Spire’s objective is to help define and refine enterprise security strategies by determining the best way to deploy policies, people, process, and platforms in support of an enterprise security management solution.

This white paper is sponsored by Forum Systems. Spire Security maintains its independence regarding the content and assertions that is the product of years of security audit, design, and consulting work.
# Attacking and Defending Web Services

## Table of Contents

**Introduction** ................................................................. 1

**Web Services Threat Profile** ........................................... 1
  - Web Services Threats .................................................. 1
  - Vulnerability Classes .................................................. 2

**Top Ten Web Services Threats** ......................................... 4
  1. Coercive Parsing ....................................................... 5
  2. Parameter Tampering .................................................. 5
  3. Recursive Payloads .................................................... 5
  4. Oversize Payloads ...................................................... 5
  5. Schema Poisoning ...................................................... 5
  6. WSDL Scanning .......................................................... 6
  7. Routing Detours ....................................................... 6
  8. External Entity Attack ................................................ 6
  9. SQL Injection ............................................................ 6
  10. Replay Attack ........................................................... 6

**Forum Systems XWall** .................................................... 7

**Spire ViewPoint** ............................................................ 7
Introduction

The key to securing any application architecture is understanding its threat profile and how that affects risk. Risk is evaluated based on an assessed level of platform weaknesses, or vulnerabilities, along with the likelihood of attack, or threat to the environment. Any time new technology is introduced in a computing environment this risk profile changes. Web Services is no different.

The characteristics of Web Services that comprise the value proposition also create a new set of exposures in the enterprise. New integration points, componentized architectures, and increased dynamic functionality contribute to new types of threats that directly impact the risk.

As is common with new technology, much work has been done with Web Services to design trust mechanisms through standards like SAML, WS-Security, XML-Encrypt, and XML-Sign, yet there has been relatively little work in defining the nature and types of threats in this environment. This paper will address the real and predictable threats that exist in the web services world.

Web Services Threat Profile

Threats have evolved with distributed architectures from monolithic mainframes to two- and three-tier client server and on to n-tier Web environments. Web services introduce the concept of an n-peer architecture where components participate in a collective manner. Three basic characteristics of Web Services create both its functional power and also risk:

► Standards provide common methods and processes but also create an opportunity for an attacker to broaden his number of targets. As standards move ‘up the stack’ this reach increases drastically and the impact is felt more.

► Loosely-coupled components create a flexible, ‘plug-and-play’ architecture with replaceable pieces that foster scalability. The communications among these components provide new risks.

► Federation of sources for data can eliminate redundancy and add to the flexibility and scalability value proposition. But this federation also assumes much about the quality of the data and the inherent trust built into the environment.

Web Services Threats

A threat profile involves evaluating the components of an architecture and identifying likely avenues of attack. As mentioned earlier, the component architecture of Web Services increases the number of touch points that can be attacked. Figure 1 shows a diagram of many of these touch points.

Every threat needs an actor, input, and a target, with a focus on the latter two (actors are assumed). With Web Services, those three points are the attacker (consumer/source), XML document (inputs) and the target (vulnerable component).
Vulnerability Classes

A specific review of the Web Services architecture provides some obvious attack points using traditional techniques. These vulnerabilities can affect both inputs and targets. What follows are descriptions of vulnerability classes based on weaknesses in inputs (in the case of XML/SOAP manipulation, protocol abuse, and untrusted configuration data) and targets (for legacy bolt-ons and untrusted entities).

XML/SOAP Manipulation

XML is the grammar and SOAP is the standard interface language of Web Services. New implementations, especially when pervasive across applications and entities, are prime targets for attackers.

XML documents are intelligent pieces of information. They may contain various types of data for input into a system. Some of the functional uses are described below:

► SOAP Headers provides a pre-defined structure with an XML message for context-sensitive information including security tokens (e.g. SAML) as well as other volatile information intended for intermediary or end-point processing

► Protocol requests/responses provide the underlying communication mechanisms that programs understand.

► Program instructions and variables can be passed as the content of XML elements.

► Uniform Resource Indicators (URIs) are pointers to the source of other types of data or information.

► Data input provides transactional data to a program.

► Embedded code can insert data in other formats to support legacy systems or specialized formats.

► It is clear that XML messages themselves can be the target of an attack or contain specific data elements that require targeted filtering for out of the norm signatures.

Protocol Abuse

Protocol abuse involves a subset of the overall XML/SOAP infrastructure. Web Services has more higher-level protocols than any previous technology. Each of these protocols provides a set of rules that can be bent, stretched, and outright broken in pursuit of weaknesses.

Untrusted Configuration Data

In a manner similar to entities, configuration data such as XML Schemas and Web Services Description Language (WSDL) files ‘live’ outside the application yet provide key information to the entities involved.

Operating as a dynamic component, the configuration information that supplies details to a web services consumer has a unique standing in the architecture. These are the sources that determine the specific operations of a service and, as such, are highly sensitive to any form of manipulation or access. Typical web services configuration information data includes:

► XML Schemas provide specific details about the grammar of a document and create the
template from which a parser interprets the documents themselves.

- **WSDL** files provide detailed information about the services ports and bindings available to consumers.

- **XSLT** files provide a mapping from one schema to another, in order to support desired transformations such as the conversion of documents from one grammar to another.

- **WS-Policy** provides handling rules and guidance about preferences for entities in a web services system.

This configuration information described can be maintained on the application server itself, housed separately in a UDDI directory or part of shipped with the transaction itself. The accuracy and integrity of configuration information highlights the importance of addressing any possibility of compromise.

**XML Processors**

XML processors may be standalone utilities or integrated into any of the components described above. Basically, they provide the intelligence to interpret XML documents as inputs to an application. More specifically, these processors perform the following functions:

- Parse the XML document into its component parts. SAX and DOM are the most popular parsing approaches. DOM is a tree-based parsing technique that builds up an entire parse tree in memory. Rather than building a tree representation of an entire document, a SAX parser fires off a series of events as it reads through the document. Streaming API for XML introduces a streaming model to parsing that resembles the SAX approach. Finally, deferred DOM parsing does not create the full tree structure of objects in memory.

- Aggregate and instantiate an XML document for processing using configuration information that is fetched typically by resolving URI’s or external pointers to repositories.

- Transform the document by using XSLT to map content from one schema to another or any other mapping required by XML manipulations such as XML Digital Signatures.

- Canonicalize data to ensure that it is not only well-formed (which is a function of the parser) but also specifically formatted so that the document will be identical wherever it happens to be built, most notably on the producer and consumer sides.

- Compress the data to meet the performance needs of a particular enterprise function.

XML processors are being integrated into every facet of the enterprise computing environment. For example,

- Data repositories contain processors to recognize parse and “shred” XML documents to be stored in file systems, XML-aware relational databases, and new XML databases.

- Web service development environments, such as applications that support J2EE and .Net, require XML processors in order to understand the inputs into the environment.

- Intelligent networks are becoming XML-aware relying on XML tags to perform common services such as content based routing and quality of service as well as value added
services such as Voice Over IP and multi-media services.

**Untrusted Entities**

The federation of services and the malleability of XML documents create an environment where a number of entities may participate in providing the application’s functionality. Like a house of cards, an untrusted entity may be attacked and bring about the fall of the entire application infrastructure.

**Legacy Bolt-ons**

Technologists always want to protect investments by prolonging their use. When new technologies come along, they work to provide interfaces to the existing application and data infrastructure. Embedded parsers can wreak havoc with an existing application if not implemented correctly.

**Putting Together the Threat Profile**

Figure 1 provides a graphical display of the interactions among all of the Web Services components that comprise its threat profile.

---

**Top Ten Web Services Threats**

In order to illustrate the nature of the threat, it is worthwhile discussing some of the types of attacks that are likely as Web Services architectures get deployed. Here are ten of the most likely techniques, employing multiple classes of input or target vulnerabilities, which will be used to attack the technology:
1. Coercive Parsing

XML is already recognized as a standard file format for many applications. As the obvious successor to legacy ASCII and presentation-oriented html, its position is unchallenged. This is easily seen by the number of grammars that claim XML as their parent.

The basic premise of a coercive parsing attack is to exploit the legacy bolt-on - XML-enabled components in the existing infrastructure that are operational. Even without a specific Web Services application these systems are still susceptible to XML based attacks that whose main objective is either to overwhelm the processing capabilities of the system or install malicious mobile code.

2. Parameter Tampering

Parameters are used to convey client-specific information to the Web service in order to execute a specific remote operation. Since instructions on how to use parameters are explicitly described within a WSDL document, malicious users can play around with different parameter options in order to retrieve unauthorized information. For example by submitting special characters or unexpected content to the Web service can cause a denial of service condition or illegal access to database records.

An attacker can embed, for example, command line code into a document that is parsed by an application that can create a command shell to execute the command. One instance of this problem is described by Georgi Guninski’s attack against Excel that formats an XML document to pass a command line to (in his example, but not limited to) enumerate the file system.

3. Recursive Payloads

One of the strengths of XML is its ability to nest elements within a document to address the need for complex relationships among elements. The value is easy to see with forms that have a form name or purpose that contains many different value elements, such as a purchase order that incorporates shipping and billing addresses as well as various items and quantities ordered. We can intuitively acknowledge the value of nesting elements three or four levels, perhaps more. An attacker can easily create a document that attempts to stress and break an XML parser by creating a document that is 10,000 or 100,000 elements deep.

4. Oversize Payloads

XML is verbose by design in its markup of existing data and information, so file size must always be considered. While an enterprise’s programmers and analysts will work to limit the size of a document, there are a number of reasons to have XML documents that are hundreds of megabytes or gigabytes in size. Sometimes this is a function of converting a batch file transfer process into real-time. It may also be anticipated in the multimedia (e.g. digital video) world where gigabyte files are the norm. Or, it could be an attacker again exercising the parser to execute a denial-of-service attack. Parsers based on the DOM model are especially susceptible to this attack given its need to model the entire document in memory prior to parsing.

5. Schema Poisoning

XML Schemas provide formatting instructions for parsers when interpreting XML documents. Schemas are used for all of the major XML standard grammars coming out of OASIS. Because these schemas describe necessary pre-processing instructions, they are
susceptible to poisoning. An attacker may attempt to compromise the schema in its stored location and replace it with a similar but modified one.

Denial-of-service attacks against the grammar are straightforward if the schema is compromised. In addition, the door is open to manipulate data if data types are compromised, like changing dates to numbers when the application is performing arithmetic operations, or modifying the encoding to allow for data obfuscation that eventually gets through to a parser and re-formed into an attack, in the same way a Unicode attack can traverse directories through web servers.

6. WSDL Scanning

Web Services Description Language (WSDL) is an advertising mechanism for web services to dynamically describe the parameters used when connecting with specific methods. These files are often built automatically using utilities. These utilities, however, are designed to expose and describe all of the information available in a method.

In addition, the information provided in a WSDL file may allow an attacker to guess at other methods. For example, a service that offers stock quoting and trading services may advertise query methods like requestStockQuote, however also includes an unpublished transactional method such as tradeStockQuote. It is simple for a persistent hacker to cycle thru method string combinations (similar to cryptographic cipher unlocking) in order to discover unintentionally related or unpublished application programming interfaces.

7. Routing Detours

The WS-Routing specification provides a way to direct XML traffic through a complex environment. It operates by allowing an interim way station in an XML path to assign routing instructions to an XML document. If one of these web services way stations is compromised, it may participate in a man-in-the-middle attack by inserting bogus routing instructions to point a confidential document to a malicious location. From that location, then, it may be possible to forward on the document, after stripping out the malicious instructions, to its original destination.

8. External Entity Attack

Another benefit of XML is its ability to build documents dynamically at the time of insertion by pointing to a URI where the actual data exists. These external entities may not be trustworthy. An attacker can then replace the data being collected with malicious data.

9. SQL Injection

Database parsers are aimed at native database languages in the same fashion as SQL injection, SQL injection could allow an attacker to execute multiple commands in an input field by using native command separators like ‘;’ or pipes. This capability may allow an attacker to execute native stored procedures or invalidated SQL commands.

10. Replay Attack

Similar to the “network ping of death” a hacker can issue repetitive SOAP message requests in a bid to overload a 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 not legitimate and constitutes an XML-based intrusion. In this manner, a completely valid XML payloads can be used to issue a denial of service attack.
Forum Systems XWall

Forum Systems’ XWall is an intrusion detection and prevention system for web services. It provides content and context-aware inspection and filtering XML documents to limit the exposure of an enterprise. The XWall extends its inspection to cover the entire XML document. In particular, XWall provides the following capabilities:

- **Layer One: Format and Syntax Inspection and Validation.** XML documents need to conform to the protocols and specifications governing their use. It is common for attackers to attempt to manipulate documents against the rules to conduct a denial-of-service attack, for example. A basic first step, then, is to validate XML documents first based on the rules of the XML and SOAP specifications, then match up and validate a particular document to its governing schema or DTD.

- **Layer Two: Content Inspection and Policy Protection.** A perfectly formed XML document may still be inappropriate and unwanted in a web services application. Deeper inspection and an understanding of the application is necessary to apply policy and filters to XML documents that pass the first level of inspection. These are custom specifications to look for policy violations like oversized documents, inappropriate or unexpected values in fields, and data dependencies for the content.

- **Layer Three: Protected Entities.** The distributed architecture of web services still requires an understanding of what entities are allowed to participate in the application. WSDL files and schemas may be enumerated or spoofed to gain an advantage. External sources may also be compromised. At the third layer, entities are identified in advance and protected through validation, masking, and local caching of details.

The XWall solution should be evaluated not only on its intelligence and analytical capabilities, but also on its ability to actively enforce policies and filters. The XWall’s capabilities begin with basic logging and alerting of violations, then progress to termination of transactions. Most importantly, XWall retains flexibility in-between logging and blocking – the point where traffic may be throttled to meet performance needs or documents are quarantined for further inspection.

Spire ViewPoint

Web Services can’t be avoided – it is unquestionably the next-generation computing architecture and even the most conservative organizations will see the underlying capabilities (like XML awareness) built into existing infrastructure. Of course, the reason Web Services can’t be avoided is that so many organizations see the power it can bring to functional computing. For that reason, deployments are cropping up everywhere.

This ubiquity breeds another negative form of attention: the attacker. As attackers learn about the characteristics in the Web Services world, they will (and already are) attack the individual components. It is necessary to consider the entire threat profile of Web Services in order to ensure that the usage is protected.

Forum Systems’ XWall provides full content inspection of XML documents to identify abnormalities and manipulations that are inappropriate to the environment. This capability is necessary for any enterprise-class web services deployment.
Contact Spire Security

To comment about this white paper or contact Spire Security, LLC about other security topics, please visit our website at www.spiresecurity.com.

This white paper is sponsored by Forum Systems. Spire Security maintains its independence regarding the content and assertions that is the product of years of security audit, design, and consulting work.