

# Watch your SOA Testing Blind Spots

Mamoon Yunus and Rizwan Mallal, Crosscheck Networks

Identify and eliminate common SOA Testing Blind Spots through lessons learned from real-life SOA deployments.

## I. INTRODUCTION

Web services – the foundation of modern Service Oriented Architecture (SOA) – are self-contained, modular applications that one can describe, publish, locate, and invoke over a network. Web services are agnostic to operating system, hardware platform, communication protocol or programming language and have blurred the boundaries between network devices, security products, applications and other IT assets within an enterprise. Almost every IT asset now advertises its interface as a Web Services Definition Language (WSDL) interface ready for SOAP/XML messaging. Using SOAP for system-to-system messaging and WSDL for interface description, IT professional now have unprecedented flexibility in integrating IT assets across corporate domains. It is this flexibility of distributed computing provided by web services that makes developing and deploying a robust, resilient and reliable Service Oriented Architecture challenging. SOA Architects, Developers and Testers are now responsible for adapting their testing techniques, selecting appropriate testing tools and developing web services domain expertise to make their SOA deployments deliver business value reliably and securely.

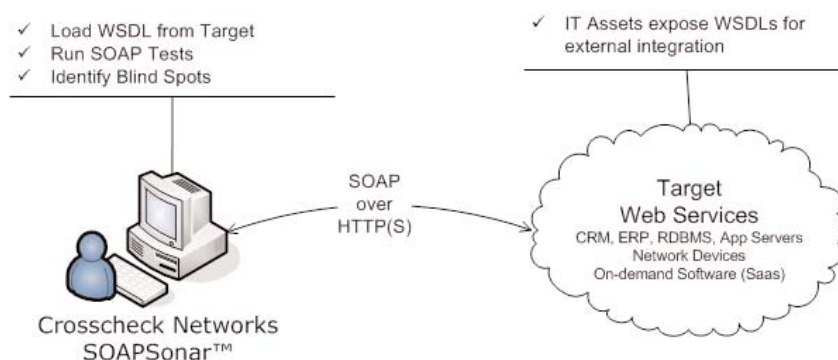
In this article, we describe common SOA Blind Spots that SOA testers have experienced recently in real-life deployments. The Blind Spots and remedies focus on the following areas:

- Performance
- Security
- SOAP Attachments
- WSDL
- Interoperability

For these 5 areas of SOA Testing, in the following sections, you will learn how to identify common SOA blind spots and techniques for avoiding them.

## II. SAMPLE SETUP

To identify and fix SOA Blind Spots, the target web services WSDL is loaded into a test tool as shown in Figure 1. The web services client can consume the WSDL-based API published by a variety of IT Assets including ERP, CRM, RDBMS, Application Servers, ESBs and On-Demand hosted services such as Salesforce.com, StrikeIron and Xignite. The WSDL file provides all the necessary constructs for the SOAP client to send SOAP requests to the target web services over HTTP protocol.



**Figure 1: Target Web Services Invocation using SOAPSonar.**

Using the test setup in Figure 1 as a simple web services consumer-producer test scenario, in the subsequent sections, we identify SOA Testing Blind Spots and describe remediation techniques and challenges associated with the identified blind spots.

### III. PERFORMANCE BLIND SPOTS

**1. Web Service are Independent:** Modern SOA is based on building *reusable* web services that can be accessed by a variety of consumers. Web service re-usability encourages a web service to become dependent on other web services. As a SOA tester evaluates performance characteristics of a web service 'A' directly in "line-of-sight," and may not be aware of the "hidden" web services that web service 'A' may depend on. Without knowledge of internal or external web services that web service 'A' may depend on, the tester assumes that the performance bottle neck is in web service 'A.' The real bottleneck, however, may exist in the web services that 'A' depends on.

**Remedy:** Modern web services applications do not operate in silos but are connected with other web services. To trace the root cause of the bottleneck, following options can be pursued:

Isolate the problem by identifying and directly testing each web service in the chain with specific test cases. This means consuming the WSDL of each web service along the chain.

Disable downstream web services and substitute the disabled web services with similar test web services with known performance characteristics. Using such simulated downstream web services, the behavior of the "line-of-sight" web services can be determined accurately.

Monitor CPU and memory statistics of each web service along the chain as the test transaction flows through the chain. This will give the tester a broad sense as to which service is the cause of the delay in the chain of web services.

**Challenges:** To isolate a web service's bottleneck, access to downstream WSDLs are necessary for simulating these web services. Corporate technical, legal and governance obstacles may prevent access to such WSDLs. Service Level Agreements (SLAs) with downstream 3<sup>rd</sup> party service providers may not include provisions for testing without incurring transaction costs.

**2. Web Services & Web Site Performance Measurement is the same:** Web site testers focus on a number of performance characteristics including response times, transactions fail/pass ratios, and throughput. For scalability testing, web site testers typically ramp up the number of concurrent clients coming into a web site and look at its performance characteristics. Testers keep an eye on the response times and determine whether the latency is acceptable. Independent of the latency, they look at the transaction fail/pass ratios and have acceptable percentage thresholds of "503 - Service Unavailable". In the web services-based SOA world where interdependent services are the norm, fail/pass ratios should be measured beyond the HTTP codes. In web services, failures are returned within the message as SOAP faults. These faults can originate from downstream web services and can propagate all the way to the calling client. Understanding when a web service has hit its "performance knee" requires deeper content awareness than just examining HTTP return codes for success and failure.

**Remedy:** Error states should be considered more comprehensively beyond just HTTP return codes. To identify fail/pass ratios, the following options should be considered:

Use HTTP success and failure codes and SOAP faults appropriately. In certain deployments consumer may require only 200 OK HTTP responses even in the presence of a SOAP level failure. This is a non-standard approach. Adhere to specifications.

Use performance tools that are web services aware so that scalability testing is not restricted to HTTP codes only. SOAP faults should trigger failure counts during performance tests.

Beyond HTTP codes and SOAP faults, business (semantic) data within a SOAP message may also indicate failure conditions. By using XPath, any data element in a SOAP body can be reviewed for business level errors. For example, during scalability testing, if a stock quote response returns a negative value, then an error has occurred. In this scenario, the HTTP codes and even the SOAP Faults indicate lack of errors.

**Challenges:** SOA implementers have to make sense of SOAP Faults generated by a web service and its downstream components. Without becoming an integral part of a web services design and development process, SOA implementers can create unwanted blind spots that prevent them from seeing and identifying a failure state. This inability to see web services failure conditions result in meaningless web services performance characteristics reduced to simple web site performance profiles. Propagating excessive information in a SOAP fault, such as stack traces, are beneficial while diagnosing a systems prior to deployment, however detailed SOAP faults are a risk and can expose corporate infrastructure internals unnecessarily. Such information leak should be avoided and only recoverable SOAP faults should be presented to consumer in run-time deployments. Finally, the business context needs to be understood of a SOAP response to successfully identify an error in the SOAP message that lacks HTTP errors or SOAP Faults.

## IV. SECURITY BLIND SPOTS

**1. Static SOAP Messages are sufficient:** A key part of SOA Web Services security testing involves WS-Security validation. A tester has to ensure that target web services that require message-level security such as WS-Signatures, WS-Encryption or SAML can consume such messages without any errors. A common blind spot in such test occurs when a tester uses SOAP messages with static timestamps, nonces, and security headers. The specifications around timestamps and security elements require unique wire signatures for all messages. The static messages should be detected by the consuming servers as an expired message, or a replay attack.

**Remedy:** Regression suites should be developed that:

Generate dynamic time stamps, nonces, and security tokens as required by the WS-Security 1.1 specifications.

Test whether the target web services honor expired timestamps by sending stale messages and replay attack detection by sending duplicate nonces.

Perform load testing where each SOAP request is unique. Taking a SOAP message with static timestamps, nonces, or security tokens should result in SOAP faults during a load test.

**Challenges:** Typical practice is to morph existing web application testing tools into web services testing tools. Static WS-Security message are generated and using a "cut-&-paste" scheme, the message is loaded into a load testing tool. Understanding the nuances of WS-Security can be overwhelming, however developing such skill and utilizing the right tools is essential for building secure and scalable SOA performance testing suite.

**2. Positive Tests Cases are sufficient:** Most of the focus is on Positive Test cases. For the more determined who want to test boundary conditions and perform negative tests, a usual approach is to build some hand picked negative tests manually and make such test part of the regression suite. Hand building negative test cases that test the boundary conditions of web services is inefficient and usually ineffective and typically based on hit-&-miss heuristics. Such manual generated negative tests may help identify a few quality issues but lack comprehensive coverage and usually result in leaving blind spots within a SOA deployment.

**Remedy:** Automate negative test case generation via automation tools. Such tools should provide sophisticated intelligence for test cases based on:

Specifics of the target web service interface. Negative Test Cases that are derived from the WSDL.

Ability to define success and failure criteria based on the SOAP responses. For example, if a web services invoked with a negative test case does not return an HTTP error or a SOAP fault, the test case should indicate a failure condition. Such behavior would indicate that the exception management of the target web services needs to be enhanced.

**Challenges:** Automating negative and positive test case generation requires sophisticated tools that are web services aware. With a focus on curtailing costs, IT shops are likely to attempt developing ineffective and inefficient heuristics-based negative and positive tests cases by using existing web site testing tools. Many SOA deployments, however, realize the ROI associated with investing in SOA tools that automate web services test generation.

## V. SOAP ATTACHMENT BLIND SPOTS

**1. Attachments are incorruptible:** SOAP is commonly used to transfer complex attachments such as MRI images, mutual fund prospectuses, and tax returns. Any binary content can be transferred as SOAP attachments based on MIME, DIME and MTOM standards. When transferring attachments from a client to a SOAP server, the tester can get an HTTP 200 OK response status, yet content may be corrupted during the transfer. Relying on just HTTP codes for evaluating SOAP Attachment transmissions can result in a blind spot that gives a SOA Tester a false sense of success.

**Remedy:** To address SOAP attachment testing, the following options can be pursued:

Develop regression suites that pre-calculate the check-sum (MD5 or SHA-1) of a message before transmittal. After the SOAP attachment is transmitted, the tester should retrieve the attachment and re-calculate the check-sum and make sure that the upload and download values match.

Ask the web services developer to return the check-sum value of the document after the upload is complete. The tester can then compare this value with the pre-calculated value. This test technique eliminates downloading the document; however it requires developers to add functionality to their code for calculating and returning check-sum values.

Consider WS-Security. For the sophisticated SOA deployments that are comfortable with WS-Signatures, the test

client can sign the SOAP attachments and the web services can verify the signature on the attachment. This signature-verification process ensures the integrity of the transmission.

**Challenges:** SOAP Attachment process requires a degree of sophistication within a SOA deployment. Although it provides significant ROI through electronic transmission of large documents compared to snail mail, regulatory mandates require attachments to be encrypted and signed. Such requirements subsequently have to be thoroughly tested by QA professionals. Investing in sophisticated tools that can handle various attachment standards, security nuances, and interoperability issues provides significant efficiency and accuracy in testing SOA deployments.

**2. Size Does Not Matter:** Well it does! SOAP attachments come in all sizes from a few Kilobytes to Gigabytes. Web services endpoints attachment handling limits depend on vendor-specific memory handling schemes. In real life, attachments such as tax return by large corporation can be a few Gigabytes in size, well beyond the 2 Gigabyte addressable memory limit of 32-bit architectures. Web services typically fail to handle SOAP attachments in the Giga byte range.

**Remedy:** To address SOAP attachment size testing, the following options should be pursued:

Understand business requirements for SOAP attachment size and build regression suites that identify the break point for message size. Ensure that web service behaves elegantly for messages that are larger than the established limit.

Communicate the tested and established attachment size thresholds to trading partners.

**Challenges:** Generating large SOAP attachments can be challenging. Developing, maintaining and running regression test for large SOAP attachments requires significant commitment in time, effort and cost from the QA Team. Identifying tools that can generate large attachment loads and provide WS-Security for such large SOAP attachments is necessary for deploying reliable SOA.

## VI. WSDL BLIND SPOTS

**1. Testing within the LAN is sufficient:** Web services operations, input & output messages, and message data types are defined in detail within a WSDL file. Developers attempt to abstract common data structures such as Business Address, Purchase Order Items, etc. as reusable data "types" expressed as schema definitions (XSDs). These schema definitions are then hosted externally so that they can be re-used by WSDLs across the enterprise. Web Service developers simply import or include these schema definitions within their WSDLs through a URI. One of the most common blind spots experienced by SOA testers is that they consume these WSDL within their internal LAN test harness and successfully run their test suites. However, once the WSDL is handed to an external trading partner, more often than not, the schema URI are inaccessible through the firewall. This blind spot becomes significant with increasing re-use within and enterprise - reuse that is the strongest virtue of a web services-based SOA.

**Remedy:** To address SOAP attachment testing, the following options should be pursued:

Test WSDL both internally and have their tests repeated from the WAN, outside the corporate firewall. This closely simulates external trading partner experience and ensures that the web services do not break when handed to external consumers.

Flatten the WSDL when handing it over to external trading partners. Inline the schema *includes & imports* such that all schema information directly appears within the WSDL and external URI are no longer required for accessing schema definitions. This WSDL flattening may be the only option where firewall restrictions prevent access to schema definitions hosted within the corporate network.

**Challenges:** WSDL complexity continues to increase and with an eye towards reuses, companies are abstracting schema definitions and making them accessible via URI imports.

**2. WSDLs are for Developers:** WSDL are the single most important contract between consumer and producers within a SOA. WSDL as typically generated or authored by developers with the intent of enabling consumers to integrate. The quality of a WSDL may vary. With generated WSDLs, a WSDL is as good as the WSDL generator and the parameters that the developer provides for generating the WSDL. For top down approaches where the WSDL is built by the developer before coding the web services, hand merging multiple WSDL operations, schemas and bindings is typically error prone. In both hand authored and auto-generated WSDLs, the developer may perform simple unit tests and fail to identify issue with the WSDL. For example, a number of IDE provide default namespaces (namespace=http://tempuri.org) that the developer may not change. This causes namespace collisions at intermediaries attempting to aggregate and merge WSDLs from different internal sources. QA testers have as much, if not more responsibility, in understanding the quality of a WSDL.

**Remedy:** To address WSDL quality issues, the following options should be pursued:

Dive into the WSDLs and understand how to identify WSDL quality.

Enable a WSDL quality assessment process that provides feedback to development teams on the quality of their WSDLs.

**Challenges:** QA Professionals should become an integral part of web services development process early. A SOA project's WSDLs should be well understood by the QA team and since the WSDL captures business specific information, the QA team should understand the business objectives in the initial phases of a web service project.

## VII. INTEROPERABILITY BLIND SPOTS

**1. All Web services talk to each other:** Interoperability assessment is the single most important part of a web services deployment. A web service is developed for consumers independent of the language, operating systems or hardware that the consumer runs on. Typical web service consumers are written in Java, .NET, and PHP. QA testers are responsible for ensuring that the published WSDLs are consumable by the languages used by the consuming applications.

**Remedy:** To address SOAP interoperability testing, the following options should be pursued:

Adopt SOA testing products that run interoperability test such as WSI-Basic Profile and WSI-Basic Security Profile interoperability tests. Such tests enforce requirements that enable cross-language interoperability. As an example, WSI-BP mandates that the SOAP message be of type *document/literal* and not *rpc/encoded*.

Establish a test harness that includes web service consumers in popular languages such as Java, .NET, and PHP and across popular parsers such as AXIS and XFire.

Test across different versions of a language framework. For example, .NET WSE 2.0 uses AES-128 for bulk encryption. WSE 3.0, however, defaults to AES-256.

**Challenges:** Building a comprehensive testing platform for SOA is challenging especially when wide proliferation is desired. As more complex operations are added to the mix, such as WS-Signatures, WS-Encryption, Attachment handling, the interoperability problem exacerbates. More configuration options are now available that need to be tested. Investing the upfront time and effort in building automation suites based on commercial SOA testing products is essential in facing the challenge of deploying a highly interoperable SOA.

## VIII. CONCLUSIONS

The promise of web service-based SOA lies in re-usability across distributed environments. The ease of developing web services puts significant burden on SOA Testers to ensure that web services are robust, reliable, secure and scalable. Through collaboration with development teams, a growing understanding of web services technologies and comprehensive SOA Testing Tools, a SOA tester can ensure that SOA Testing blind spots are reduced if not eliminated.

## IX. ABOUT CROSSCHECK NETWORKS: [DOWNLOAD Complimentary SOAPSonar Personal Edition](#)

Crosscheck Networks is focused on providing products for **SOA Diagnostics**. Crosscheck's flagship product, **SOAPSonar**, reduces the pain point of testing Web Services. It provides comprehensive code-free web services testing with an intuitive interface. Simple navigation, Drag/Drop WSDL loading, Drag/Drop Test Suite Management, Complex WSDL Support (WSDL and XSD imports) and sophisticated Performance techniques make it easy to validate your web services. SOAPSonar provides comprehensive testing across the Four Pillars of SOA Testing: Functional Regression, Performance Testing, Interoperability Conformance and Vulnerability Assessment.

About the Authors:

**Rizwan Mallal** is the Managing Director at Crosscheck Networks. Also, as the founding member and Chief Security Architect of Forum Systems, Rizwan is responsible for all security related aspects of Forum's technology. Previously, Rizwan was the Director of Engineering at Sonicwall (SNWL). He joined Sonicwall through the Phobos acquisition, where he was the Chief Architect of the SSL product line. Before joining Phobos, he was the technical architect at Raptor Systems where he was one of the pioneers of VPN/Firewall space in the mid 1990s. Raptor after its successful IPO in 1996 was later acquired by Axent/Symantec (SYMC).

**Mamoon Yunus** is an industry-honored CTO and visionary in Web Services-based SOA technologies. As the founder of Forum Systems, Mamoon pioneered Web Services Security Gateways & Firewalls. He has spearheaded Forum's direction and strategy for six generations of award-winning Web Services Security

products. Prior to Forum Systems, Mr. Yunus was a Global Systems Engineer for webMethods (NASDAQ: WEBM) where he developed XML-based business integration and architecture plans for Global 2000 companies. He has held various high-level executive positions at Informix (acquired by IBM) and Cambridge Technology Group. InfoWorld recognized Mamoon as one of 4 "Up and coming CTOs to watch in 2004." He is a sought after speaker at industry conferences such as RSA, Gartner, Web Services Edge, CSI, Network Interop, and Microsoft TechEd. Mamoon has the distinction of showcasing Forum Systems' entrepreneurial leadership as a case study at the MIT Sloan School of Management. He has also been featured on CNBC as Terry Bradshaw's "Pick of the Week." Mamoon holds two Graduate Degrees in Engineering from MIT.