The Journal of Systems and Software 109 (2015) 18–31

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Assessing the security of web service frameworks against Denial of Service attacks

Rui André Oliveira, Nuno Laranjeiro*, Marco Vieira

CISUC, Department of Informatics Engineering, University of Coimbra, Coimbra, Portugal

ARTICLE INFO

Article history:
Received 29 July 2014
Revised 28 June 2015
Accepted 2 July 2015
Available online 14 July 2015

Keywords:
Security
Web service frameworks
Experimental assessment

ABSTRACT

Web services frequently provide business–critical functionality over the Internet, being widely exposed and thus representing an attractive target for security attacks. In particular, Denial of Service (DoS) attacks may inflict severe damage to web service providers, including financial and reputation losses. This way, it is vital that the software supporting services deployment (i.e., the web service framework) is able to provide a secure environment, so that the services can be delivered even when facing attacks. In this paper, we present an experimental approach that allows understanding how well a given web service framework is prepared to handle DoS attacks. The approach is based on a set of phases that include the execution of a large number of well-known DoS attacks against a target framework and the classification of the observed behavior. Results show that four out of the six frameworks tested are vulnerable to at least one type of DoS attack, and indicate that even very popular platforms require urgent security improvements.

1. Introduction

Web services (WS) are being deployed in diverse scenarios, ranging from small local businesses to more global scale business-to-business environments. SOAP-based web services consist of self-describing components that can be used by other software across the web in a platform-independent manner, are supported by standard protocols such as SOAP and WSDL (Curbera et al., 2002), and nowadays play a key role in the integration of heterogeneous systems. In a web services environment, the provider offers a well-defined interface to consumers, which includes a set of operations and typed input/output parameters.

A set of steps must be performed for a provider to offer a SOAP-based web service. First, a developer must code the **web service application**. After the service application is coded, it is deployed in the context of an **application server** (e.g., Apache Tomcat, JBoss AS). This requires the use of a **web service framework** (or stack) that, together with the application server, acts as a container for the web service. In the context of this combined set of software components, the application server delivers the SOAP messages arriving from clients (typically via HTTP) to the web services framework, which then processes (parses messages, checks for errors, and builds programming language-level objects) and delivers them to the web service application (U and Rao, 2005).

The web service framework is a key software component and handles all SOAP client requests including valid requests (those that are compliant with the WS specification), invalid requests (not compliant with the WS specification), and malicious requests (that try to exploit vulnerabilities in the web services frameworks and/or applications). Deploying a web service is currently a fairly simple process, as the supporting technologies are in a matured state and existing development tools provide easy ways to create services based on local (i.e., non-remote) software applications.

A key problem is that developers tend to assume that the maturity of the underlying web service frameworks, resulting from years of field experience, refined development, tests, and practical use, guarantees that the service applications are deployed in a stable and secure environment, which is not always the case. In fact, previous research shows that, although web services technology has reached a mature state and is designed based on a large amount of networking experience, WS frameworks are not more secure than any other network-based system (Jensen et al., 2009; Suriadi et al., 2010).

One of the main threats to the availability of the services deployed on the web are the well-known Denial of Service (DoS) attacks, which consist on overloading the server resources in a way that leads to legitimate users being denied to use a service (Ranjan et al., 2009). The authors in (Imperva, 2012) show that the target of this type of attacks is moving from the lower layers of the communication system (e.g., TCP/IP protocols) to the upper layers (e.g., the web

* Corresponding author. Tel.: +351 239 790 000.
E-mail addresses: racoliv@dei.uc.pt (R.A. Oliveira), cnl@dei.uc.pt (N. Laranjeiro), mvieira@dei.uc.pt (M. Vieira).

http://dx.doi.org/10.1016/j.jss.2015.07.006
0164-1212/

service). In fact, DoS attacks are more effective at the application level and, since many anti-DoS solutions target lower communication layers (Cloudfare, 2014; "Defeating DDOS Attacks," 2014), they often pass undetected.

Studies that try to understand how web service frameworks behave when facing DoS attacks are quite limited. Although some work has been conducted to detect vulnerabilities in web service applications and to understand how they can be exploited (Antunes et al., 2009; Duchi et al., 2014; Vieira et al., 2009) studies focusing on assessing the behavior of frameworks in the presence of DoS attacks are reduced to exploratory and isolated examples (Jensen et al., 2009; Suriadi et al., 2010). Furthermore, the existing tools that allow emulating WS attacks (e.g., penetration testers and fuzzers) are also limited, mostly targeting the service implementation itself and disregarding the potential vulnerabilities of the underlying framework ("soapUI - Functional Testing," 2012; "WSBang," 2012; "WS-Fuzzer Project," 2012). In fact, the tools that can be used to attack frameworks frequently include limited sets of attacks, but most of all, even though they can be used to attack some platform, they are not prepared to support the evaluation of WS frameworks, as they lack the description of the set of steps and metrics required to assess a given framework. Thus, tools become rather useless for both practitioners (e.g., to assess the security of their platforms) and researchers (e.g., to study frameworks in terms of different security properties).

In this paper we propose an experimental approach to evaluate the security of web service frameworks. The approach builds on the work presented in Oliveira et al. (2012a) and is based on a set of DoS attacks that have been compiled from security research studies, existing security tools, and field experience, and that target core message exchange features (i.e., communication involving common data types). Attacks focusing web services extensions (which provide features such as message integrity or confidentiality) are out of the scope of this work. In practice, we use the compiled attacks in combination with regular requests in a set of runtime tests (during three distinct phases) to assess the behavior of frameworks in the presence of such malicious requests. Observed failures are classified using an adaptation of the CRASH scale (Koopman et al., 1997) and dubious behaviors (that indicate abnormal allocation of system resources) are also analyzed. Comparing to the work in Oliveira et al. (2012a), we perform the following extensions: 1) we augment the approach with a technique for analyzing the impact of the attacks on frameworks in a quantitative manner; 2) we include the comparison of pairs of test periods (e.g., the comparison of the behavior observed in the presence of attacks with the one previously observed when executing non-malicious requests), which allows us to understand the impact of the attacks in a more detailed manner; and 3) we evaluate the latest version of a larger number of web service frameworks and for some cases we compare the results with the ones from previous versions (in order to understand the security evolution of the frameworks over time).

The experimental evaluation includes well-known and widely used frameworks, namely: Apache Axis 1, Apache Axis 2, Apache CXF, Oracle Metro, Spring JAX-WS, Spring-WS and XINS ("Apache Axis2/Java," 2012; "Apache Axis," 2006; "Apache CXF," 2012; "Metro," 2012; "Spring support for JAX-WS RI—Project Kenai," n.d.; "Spring Web Services - Home," 2013; "XINS - Open Source Web Services Framework," 2013). Results show that most of frameworks are quite resistant to the large set of security attacks executed by our tool, but the issues detected (e.g., high CPU/memory usage and unexpected exceptions) also indicate the potential presence of major security vulnerabilities in the frameworks, requiring urgent attention and corrective measures from developers. The quantitative analysis performed, highlighted previously unnoticed impact of attacks and disclosed additional security issues, even for frameworks that were tested in previous work (Oliveira et al., 2012a).

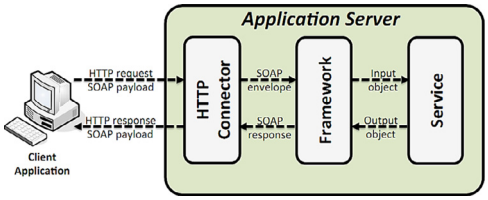

Fig. 1. Web services interactions and supporting infrastructure.

The paper is organized as follows. The following section presents background and related work. Section 3 describes the approach used to test the security of web service frameworks. Section 4 presents the experimental scenarios designed to demonstrate the proposed approach and Section 5 presents the results obtained for each of the tested frameworks. Section 6 discusses the quantitative impact of the attacks and compares the behavior of different versions of the same framework. Finally, Section 7 concludes the paper.

2. Background and related work

This section presents core concepts regarding the web services technology and supporting environment, and previous work on DoS attacks and framework-level vulnerabilities. The section concludes with an overview of the state of the practice, with focus on the limitations of current WS security testing tools.

2.1. Concepts on web services

In a web services environment a provider supplies a set of services to consumers (Curbera et al., 2002). The Simple Object Access Protocol (SOAP) is used for exchanging XML-based messages between the consumer and the provider over the network (usually using HTTP). In a typical interaction the consumer (the client) sends a request SOAP message to the provider (the server). After processing the request, the server sends back a response with the results. A service may include several operations and is described using WSDL (Web services Description Language) (Curbera et al., 2002). Restful Web services (Oracle, 2013), based on plain XML or JSON, are nowadays becoming quite popular also and represent another architecture being used in service-based environments. Despite sharing common characteristics (e.g., XML-based responses), this paper is specifically focused on SOAP-based services, and the adaptation of the proposed approach to other WS technologies is considered as future work.

As shown in Fig. 1, the core software components supporting SOAP web services and allowing end-to-end communication between clients and servers are: 1) a web services framework (e.g., Apache Axis or Metro); and 2) an application server (e.g., Apache Tomcat). In practice, a client sends a SOAP message via HTTP, the HTTP connector handles and processes the incoming HTTP request, retrieves the SOAP message and delivers it to the web service framework. The framework then processes and delivers the SOAP message to the actual service implementation (i.e., the web service application). In short, the framework validates each message and transforms it into an object that can be handled by the application. After this object is processed by the application, the reverse path is taken, with the return object being serialized into a SOAP response that is sent via HTTP to the client (U and Rao, 2005).

There are currently tens of web service frameworks available. Some can be used with their own proprietary application server (e.g., Axis), while others are included in enterprise application servers (e.g., Glassfish or JBoss AS) or can generally be added to more lightweight servers, such as Apache Tomcat or Jetty. In fact, these are very popular among developers and providers, as they deliver the typical core features that are required by web applications (e.g., HTTP request handling, servlet container, JSP engine, etc.) and are not loaded with