Web Services - The Technology and its Security Concerns.

White paper – May 2007

This white paper examines the technology behind Web Services, how the system is made available to the user, and the way connections are made to back-end (and therefore sensitive) data. These different elements come together to make Web Services a portal for users to access data, but also provide different entry points which may be exploited for illegitimate purposes. These security flaws bring about the need for an added security-assessing component in the Acunetix WVS solution. Support for Web Services vulnerability scanning is now provided by a dedicated component which is specifically designed to detect exploitable entry-points in a Web Services system.

Table of Contents

1. The Web Services Building Blocks

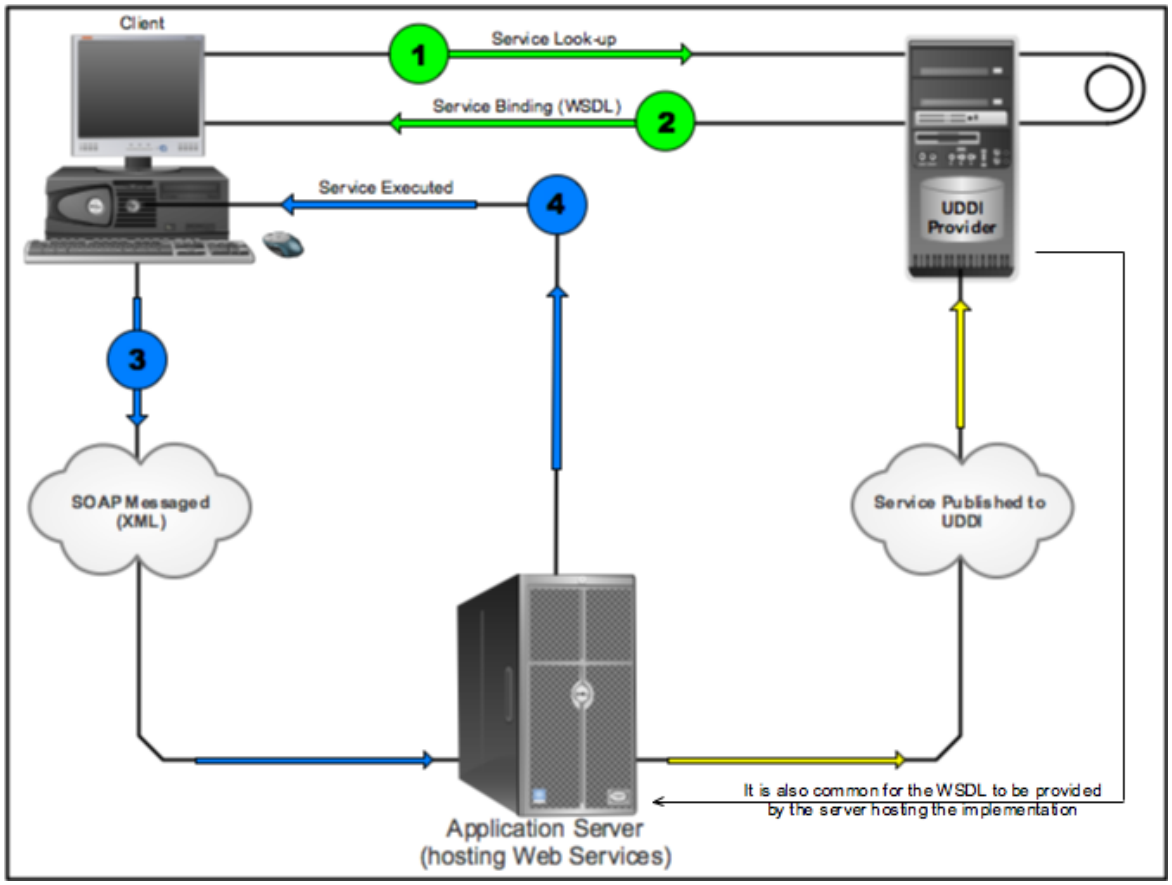
1.1 The Web Service architecture comprises different technologies which enable a client to obtain data from a server, using the SOAP protocol. SOAP originally stood for Simple Object Access Protocol; however it is now a free standing acronym since the W3C body deemed it as misleading. A Web Service provides a web API (application programming interface) which enables two applications to communicate using XML over the web, or a network connection. This system was created to act as a middle agent when *application-to-application integration* was an issue which required a solution. A Web Service may be developed in any language and deployed over any platform, but most importantly it may be accessed by any other application regardless of the language used to develop it. SOAP serves as the entity which uses XML to collect the specific message, the service, the interface or port type, and the service binding (the binding contains information about the service such as its hosting redirector and access point).

1.2 Technologically defined, the word *Service* describes a resource which is utilized by an application and not by a person. Following that definition, a Web Service is a server-oriented system which therefore operates on the server-side, and performs a task when it is called upon by an application. Like any service, a Web Service requires an API to provide an interface which allows it to be called by another application. As can be seen in an operating system of a common personal computer, a service is registered in the system registry which allows applications to locate the specific service to process a specific task. In the same way, a Web Service is registered in a Web Service registry, which an application uses to call the specific service it requires. As mentioned earlier, a Web Service is not language and platform dependent, it uses XML to communicate with other services or applications, and just like any internet web-based system it does not require a specific platform on which to operate.

1.3 XML (Extensible Markup Language) is a versatile language which was designed to enable various different systems to share information and instructions in a universal manner. Web Services use a format of XML developed to describe network services as a set of components which exchange messages containing procedure or document descriptive data. This language is known as WSDL (Web Serviced Description Language), and is a format of XML because of its flexibility as a markup language. A WSDL file contains information about the different components and their respective messages, the message format being used, and the network protocol over which the messages are being communicated. Simply put, the WSDL file is the key communicative agent between the various entities exchanging service messages, and instructions between them.

1.4 An essential element of the Web Services architecture is the central directory which contains all the service descriptions. A service-oriented system must have a registry which takes care of associating the right service to the request being processed, and also functions as a discovery system for the correct service to be identified by the requestor. The mechanism which performs this task is the UDDI Provider. UDDI stands for Universal Description Discovery and Integration. The UDDI Provider hosts a standardized record which creates the profiles of registered services, and through this standardized profile it is possible to match a particular request with its corresponding service. International and publicly available business service descriptions are hosted in a directory known as a Public Business Registry..

2. Web Services in action



After becoming familiar with the key elements responsible for making the Web Services work, one needs to see how these elements interact with the whole system, from the client requesting a service to perform a task, the service being executed, and data delivery.

A simple Web Service which may be used as an example is one which allows a client to convert one currency to another. The web application used as the front-end contains a simple form which allows the user to select the starting currency, and the currency to which he wants the conversion to be done. The user submits this data, and the application contacts the UDDI provider to look up the service required to perform this conversion. The UDDI provider then creates the binding, which associates the message to the service requested, and its location. The UDDI provider then returns a WSDL file to the client, which the application completes as a SOAP message. The SOAP message then gets sent to the application server which hosts the Web Service needed to execute the currency conversion. This is done using the binding details in the WSDL file from the UDDI. Using the SOAP instructions, the Web Service can correctly execute the task according to the parameters it was given, and deliver the processed currency conversion back to the requestor.

3. Web Services - Security Concerns

Fundamentally, Web Services operate on the same structure used by normal web applications. The beginning of the chain is a request forwarded by an application viewed in a web browser, which for Web Services is a SOAP request over HTTP. Since SOAP data is received by the server, but not sent to the client, one can understand that the threat is primarily aimed at the server itself. The following are methods of attack, and how Web Services can be exploited to fulfill these attacks:

3.1 Buffer Overflows:

Common Effects: DOS (Denial of Service), data corruption, malicious code execution.

An attacker can craft XML data causing the XML to call upon itself repetitively therefore constantly increasing in size. This causes a memory overflow, or trigger error messages which reveal information about the application.

A DOS attack can be caused by forcing a server to parse an abnormally long XML file, which in essence uses up much more resources then actually generating one, and can crash the application. Another type of attack consists of sending a block of data to an application, which is stored in a buffer of insufficient size. This block of data can then overwrite genuine data and cause a function return which gives control to the malicious code in the hacker's data block.

3.2 XML Injections:

Common Effects: Command execution, data theft and deletion, schema poisoning.

SQL Injection is a high-risk exploit which may be performed using SOAP messages. If a server does not validate data correctly, a SOAP message can easily be used to create XML data which inserts a parameter into an SQL query and have the server execute it with the rights of the Web Service. SQL Injection is only one of the threats a server is exposed to if data is not validated.

Another such example is Schema Poisoning. A schema file is what an XML parser uses to understand the XML's grammar and structure, and contains essential preprocessor instructions. An attacker may damage the XML schema or replace it with a modified one which would then allow the parser to process malicious SOAP messages and specially crafted XML files to inject OS commands on the server or database.

3.3 Session Hijacking:

Common Effects: Obtaining of user privileges within application or network

Session hijacking involves gaining illegal control of a legal user's session state. It occurs when an attacker steals a valid session ID (valid session cookie), and uses it to gain that particular user's privileges in the application. By intercepting or sniffing SOAP messages, an attacker can hijack a user's session in the same ways as with normal web application attacks, however once a hacker is authenticated as a valid user he may perform more dangerous activities.

4. Summary and Conclusions

The idea of the internet as we know it is quickly surpassing the simple need to obtain information with ease through web applications, and is now evolving into a multitude of systems which perform tasks, calculations, accurate searches, and many other complex operations. Web Services are the perfect example of a solution to the need for a simplistic system which allows many different technologies to collaborate and communicate with each other. Being available to the end user over the internet, Web Services will keep increasing in popularity due to their functionality, and this popularity will also expose the threat to the servers hosting them.

Over the past year, there has been an increased concern among developers and security analysts searching for a tool to reveal the vulnerabilities associated to Web Services. The increase in concern has not yet raised enough awareness about the risks which threaten the security of the servers hosting Web Services and the data which risks being compromised.

Acunetix Web Vulnerability Scanner is a feature-packed solution for detecting vulnerabilities and securing web applications. The upcoming release of Version 5 will see a new addition to the suite of tools in its arsenal. The Web Services scanning tool will allow you to run an automated vulnerability assessment against a Web Service with a more accurate and improved version of the same scanning engine which till now assessed web applications. Another new addition is the Web Services Editor which extends the functionality of the Web Services scanner by allowing deeper analysis of XML responses, WSDL structure, WSDL XML analysis, syntax highlighting for all coding languages, and regular expression searching. These new features make Acunetix WVS a complete solution for securing web applications and now also Web Services.

Jacques Guillaumier, May 2007

About Acunetix

Acunetix was founded to combat the alarming rise in web attacks. Its flagship product, Acunetix Web Vulnerability Scanner, is the result of several years of development by a team of highly experienced security developers. Acunetix is a privately held company with headquarters based in Europe (Malta), a US office in Seattle, Washington and an office in London, UK. For more information about Acunetix, visit: http://www.acunetix.com; http://www.acunetix.de.

© 2007 Acunetix Ltd. All rights reserved. The information contained in this document represents the current view of Acunetix on the issues discussed as of the date of publication. Because Acunetix must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Acunetix, and Acunetix cannot guarantee the accuracy of any information presented after the date of publication. This White Paper is for informational purposes only. Acunetix MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT. Acunetix, Acunetix Web Vulnerability Scanner and their product logos are either registered trademarks or trademarks of Acunetix Software Ltd. in the United States and/or other countries. All product or company names mentioned herein may be the trademarks of their respective owners.