INTERNATIONAL JOURNAL OF ENGINEERING AND MANAGEMENT SCIENCES

www.scienceandnature.org

SECURING WEB SERVICES

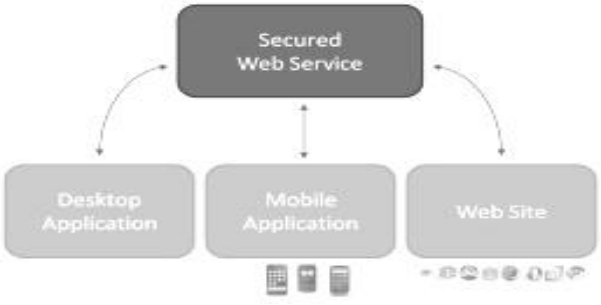
Ahmad Tasnim Siddiqui & Arun Kumar Singh
Research Scholar, Singhania University, Pacheri bari, Jhunjhunu Rajasthan, India

ABSTRACT
HTTP, Web Server and Web Services share very complicated set of functionalities and exchanges of information. Each and every component plays very important role in the thousands of functions which any user can access and utilize over Internet. Hyper Text Transfer Protocol allows users to interact with Web Servers and hence they can access the information via the Internet. If any user requests data and files, Web servers serve them. Web Services allow cross-system, cross-language communication among various types of machines and enable inter-business transaction and communications. Although each technology works on its own and performs many useful functions, it is the combination of these technologies that has created the dynamic functionalities of the Web that are available today. This research paper will explore the inter-relationships between HTTP, Web Servers and Web Services technologies that have facilitated the functionalities and convenience of the Web.
Web Services are very powerful tool that has greatly enhanced the efficiency and communication among businesses. According to the World Wide Web Consortium (W3C), "a Web Service is a software system designed to support interoperable machine-to-machine interaction over a network." According to Zeldman, Web Services are a "reusable software components based on XML and related protocols that enable near zero-cost interaction throughout the business ecosystem." In other words, Web Services are the software system that allows servers and client computers to communicate with each other regardless of each individual machine's environment (operating systems and programming languages). The Extensible Markup Language (also popularly known as XML) page provides a very nice formula that clearly defines the major components of Web Services. According to Simon, Web services = XML + SOAP + WSDL + UDDI.
There are only two reasons for the need to secure a Web Service. First, the data it serves is sensitive in some way – it all needs to be locked away from any un-authorized access, or only certain users can view certain sets of data (such as financial data) and secondly it allows users to upload data or otherwise modify something, and there's a need to know who did what, or to restrict who can do what.
Web services are used by an increasing number of companies as they expose products and services to customers and business partners through the Internet and corporate extranets. The security requirements for these service providers are of paramount importance. There are various threats to the web services but few major threats can be given as Unauthorized Access, Parameter Manipulations, Network Eavesdropping, Disclosure of Configuration Data, Message Replay etc. We should also keep in mind about the threats from SQL Injection.

KEYWORDS: Web service, web service security, securing a web service, threats to web services, web service security requirements.

INTRODUCTION

Web Services are a promising solution to an age-old need: fast and flexible information sharing among people and businesses. Web Services enable access to data that has previously been locked within corporate networks and accessible only by using specialized software. Along with the benefits of Web Services comes a serious risk: sensitive and private data can be exposed to people who are not supposed to see it. Web Services will never attain their tremendous potential unless we learn how to manage the associated risks. Web Services represent the next phase of distributed computing, building on the shoulders of the previous distributed models. Widespread distributed computing started with the Transmission Control Protocol/Internet Protocol (TCP/IP). Using TCP/IP to build distributed products was hard work for application programmers, who just wanted to build business applications. To ease the burden of distributed programming the computer industry developed the Distributed Computing Environment (DCE) based on the client/server computing paradigm, followed by the Common Object Request Broker Architecture (CORBA). About the same time, Microsoft introduced the Component Object Model (COM), followed by Distributed COM (DCOM) using DCE technology as a base, and COM+. Sun, building on its Java language introduced the Java 2 Platform, Enterprise Edition (J2EE), with its popular Enterprise Java Beans (EJBs), using many concepts and research ideas from the previous technologies. Each step made distributed computing easier but each technology still lived, for the most part, in its own world, making interoperability between the different middleware technologies difficult.



Now Web Services have burst on the scene. There are two major Web Services goals—to make distributed computing easier for the business programmer and to enhance interoperability. These goals are aided by:

- Loose coupling between the requesting program and the service provider
- The use of Extensible Markup Language (XML), which is platform and language neutral

An important requirement for Web Services is to support secure interoperation between the underlying object models, such as .NET and J2EE, as well as to support interoperation between the perimeter security and the mid-tier, and between the midtier and legacy or back-office systems. To this end, we give significant detail describing the problems of maintaining secure interoperability and how you can overcome these problems. The distributed security community, as represented by the Organization for the Advancement of Structured Information Standards (OASIS), the World Wide Web Consortium (W3C), and the Internet Engineering Task Force (IETF), has offered the solutions to some of these problems in specifications that have been developed by the cooperative efforts of their member companies. Other organizations, such as the Web Services Interoperability Organization (WS-I) and the Java Community Process (JCP) have worked on additional solutions.

In today's global marketplace, the Internet is no longer just about email and Web sites. The Net has become the critical conduit powering a growing list of revenue-generating e-business activities—from e-commerce and e-supply chain management to online marketplaces and collaboration. Web Services leverage the ubiquity of the Internet to link applications, systems, and resources within and among enterprises to enable exciting, new business processes and relationships with customers, partners, and suppliers around the world.

The benefits of Web Services are not limited to interactions between different companies. Business units within the same enterprise often use very different processing environments. Each policy domain (that is, scope of security policy enforcement) is likely to be managed differently and be under the control of different organizations. What makes Web Services so interesting is that they provide interoperability across security policy domains.

II.0 COMMON THREATS TO WEB SERVICES

There are many complexities specific to, and inherent in Web services that further complicate their security. Numerous threats can compromise the confidentiality, integrity, or availability of a Web service or the back-end systems that a Web service might expose. Some of these threats are shared with conventional Web application systems (Web sites), while others are specific to Web services. However, before delving into specific Web service security issues, it would be wise to first examine the general security threats that can occur in any Web application.

Typically, Web sites and Web services use common technologies in terms of the programming languages of the application. For example, both applications use data stores and application servers on the back end; and on the front end, both typically use a Web server and are exposed over HTTP. Such architectural and technological similarities result in Web services inheriting many common Web site security threats.

II.1 SQL Injections

When SQL statements are dynamically created as software executes, there is an opportunity for a security breach: if the hacker is able to break perimeter security and pass fixed inputs into the SQL statement, then these inputs can become part of the SQL statement. SQL injections can be generated by inserting spatial values or characters into SOAP requests, Web form submissions, or URL parameters. A hacker who knows his SQL can use this technique to gain access to privileged data, log-in to password-protected areas without a proper log-in, remove database tables, add new entries to the database, or even log-in to an application with admin privileges.

II.2 WSDL Scanning & Access

A WSDL document contains information pertaining to the Web service such as available operations, the content of the messages each of these operations accepts and returns, and the endpoints that are available to invoke the operations. Securing a Web service should rely on cryptographic measures to fully protect information rather than obscurity (security by obscurity should be avoided) because the information that WSDL provides can expose certain architectural aspects about the Web service that could make it easier for an unauthorized user to mount an attack.

II.3 XML Bombs

DTDs may have recursive entity declarations that, when parsed, can quickly explode exponentially to a large number of XML elements. This consumes the XML parser resources causing a denial-of-service. For example:

```
<?xml version="1.0" ?>
<!DOCTYPE foobar [
<!ENTITY x0 "Bang!">
<!ENTITY x1 "&x0;&x0;">
<!ENTITY x2 "&x1;&x1;">...
<!ENTITY x99 "&x98;&x98;">
<!ENTITY x100 "&x99;&x99;">
                                    ]>
```

If it is processed, the DTD above explodes to a series of 2100 "Bang!" elements and will cause a denial-of-service.

II.4 XPath Injections

XPath injections are similar to SQL injections in that they are both specific forms of code injection attacks. XPaths enable one to query XML documents for nodes that match certain criteria. For example, an XPath can be as simple as the following:

```
//*[localname(.)="
user"][attribute:
:username="somebody"]/@*
[local-name(.)="password"]
```

The above XPath returns the value of the password attribute for the username "somebody." If such a query is constructed dynamically in the application code (with string concatenation) using invalidated inputs, then an attacker could inject XPath queries to retrieve unauthorized data.

III.0 WEB SERVICES SECURITY REQUIREMENTS

Let's begin by defining some core security services that are fundamental to end-to-end application security across multitier applications. They are:

III.1 Authentication: Verifies that principals (human users, registered system entities, and components) are who they claim to be. The result of authentication is a set of credentials, which describes the attributes (for example, identity, role, group, and clearance) that may be associated with the authenticated principal.

III.2 Authorization: Grants permission for principals to access resources, providing the basis for access control, which enforces restrictions of access to prevent unauthorized use. Access controls ensure that only authorized principals may modify resources and that resource contents are disclosed only to authorize principals.

III.3 Cryptography: Provides cryptographic algorithms and protocols for protecting data and messages from disclosure or modification. Encryption provides confidentiality by encoding data into an unintelligible form with a reversible algorithm, which allows the holder of the decryption key(s) to decode the encrypted data. A digital signature provides integrity by applying cryptography to ensure that data is authentic and has not been modified during storage or transmission.

III.4 Accountability: Ensures that principals are accountable for their actions. Security auditing provides a record of security-relevant events and permits the monitoring of a principal's actions in a system. Non-repudiation provides irrefutable proof of data origin or receipt.

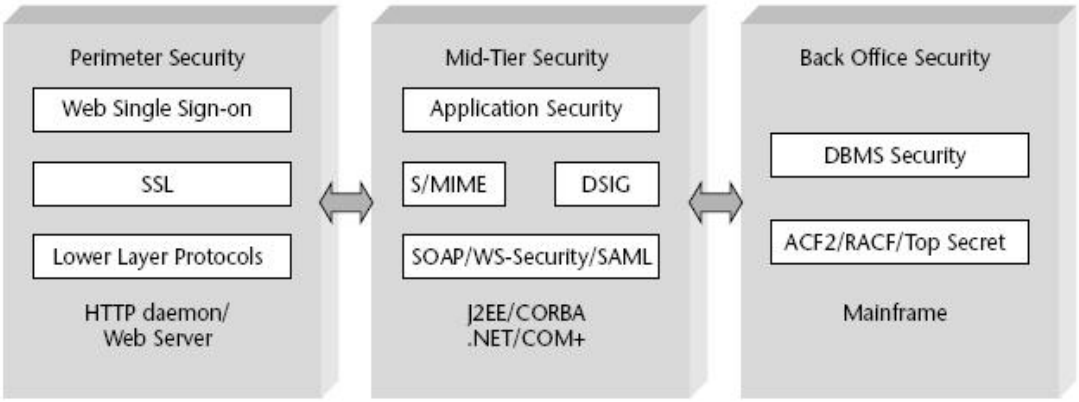
III.5 Security administration: Defines the security policy maintenance life cycle embodied in user profiles, authentication, authorization, and accountability mechanisms as well as other data relevant to the security framework.

All security services must be trustworthy and provided with adequate assurance. That is, there must be confidence that security services have been implemented correctly, reliably, and without relying on the secrecy of proprietary mechanisms.

IV.0 PROVIDING SECURITY FOR WEB SERVICES

Given the diverse nature of these distributed environments, it is not surprising that Web Services security efforts to date have taken a "patchwork" approach. This patchwork may include a range of existing, standalone Web security mechanisms, together with operating system security (domain logins), communications security (SSL), applications environment security (J2EE, COM+, .NET, or CORBA), and SSO (Netegrity SiteMinder, IBM/Tivoli Policy Director, or others) solutions. Even electronic mail systems can support Web Services. The problem is that each of these solutions has evolved to solve a specific problem within a single tier or domain. While there have been attempts to extend these solutions beyond their original scope, the results have not been very rewarding.

The following diagram illustrates new and existing security mechanisms for securing Web Services at different security tiers. For instance, where access to the Web Service is through a Web Server, Secure Sockets Layer (SSL) and Web SSO can be used. At the application level, Security Assertion Markup Language (SAML) can be used to support authentication and authorization across domains. Finally, access to a mainframe is needed to complete the request, and a mainframe authentication system is in place.



Example of Web Service security implementation (Source: Mastering web services, Bret Hartman, Donald J. Flinn etc.)

V.0 SECURING .NET WEB SERVICES

We can use the security features when your web services are implemented using the Microsoft .NET framework. Not only are there multiple options for building a Web Service with .NET, there are also multiple alternatives for protecting Web Services applications. An important building block of Microsoft's Web Services solutions, IIS plays a critical role in protecting the hosted .NET Web Services. The security mechanisms IIS provides can be classified in the following basic groups: authentication, message protection, access control, logging, and fault isolation.

V.1 Access Control

All these efforts by IIS—to use a system account for anonymous requests, map clients authenticated with X.509 certificates into user accounts, and support impersonation by HTTP handlers—are mainly for the sake of leveraging native OS access control mechanisms for protecting IIS resources. When an HTTP handler accesses an HTML or any other file in order to process the corresponding HTTP request, Windows file system permissions in the form of a

discretionary access control list (DACL) are used by the OS to enforce access control policies. Keep in mind that this mechanism is only available on NTFS file systems. If a file to be accessed resides on a FAT partition, no access checks are done.

Two other access control mechanisms—Web permissions and IP-based restrictions—could be used in addition to DACL controls.

V.2 Logging

Security auditing is not an option for IIS, although its logging facilities can serve as a partial substitution. If configured to do so, IIS logs information in text format about HTTP requests into %winnt%\system32\LogFiles\W3SVC<n>, where <n> is the number of the Web site instance. You can select, per Web site instance, what information is recorded about processed requests. There are around 20 request-specific details and a number of process accounting properties that could be recorded on each request. Whether to log a request to a particular IIS resource is determined by the option "Log visits".

V.3 Fault Isolation

Fault isolation can be considered as a part of the group of security mechanisms known as service continuity. Although IIS does not offer full-blown service continuity solutions, at least it provides a way to isolate HTTP handlers from the main process in which IIS is executing, InetInfo.exe, and from each other. This is done through configuring applications in virtual directories to run with one of the following options (supported by IIS v5):

V.4 ***IIS process:*** All requests to the files in the virtual directory are handled in the space of InetInfo.exe. Having the best performance, this option does not offer any fault isolation. That is, if the handler crashes because of an error in the application code, IIS will crash as well.

V.5 ***Pooled:*** Requests to the resources of all virtual directories configured with this option run in the same process external to InetInfo.exe. The process runs under the identity of an account controlled by IIS, IWAM_<machinename>. This offers the best performance versus robustness trade-off, because if a Web application crashes, it takes down only other applications, but not InetInfo.exe, which will be able to relaunch the pool process on the next request that needs one of the pooled handlers. This is the default option.

V.6 ***Isolated:*** Executes each Web application in its own process that runs under the IWAM_<machinename> account. This option has the highest level of fault isolation—no faulty application could bring down any other application—but it is not as fast as the previous one.

VI.0 ASP.NET AUTHENTICATION SERVICES

Authentication facilities can either ride on top of IIS authentication or get by without it. First of all, if IIS authentication is used, then ASP.NET can be configured to accept the authenticated identity so that Web Services will have access to it. This is done through the following element in the ASP.NET configuration file, *web.config*:

```
<configuration>
<system.web>
<authentication mode="Windows"/>
</system.web>
</configuration>
```

Use of the value "Windows" instructs ASP.NET to take advantage of the authentication performed by the IIS.

VI.1 HTTP Modules

If you are not satisfied with any of the above methods of authentication, you still have one more option available in ASP.NET.

```
<configuration>
<system.web>
<httpModules>
<add name="FooAuthenticationModule"
type="eBusiness.Authentication.FooAuthModule,
eBusiness.Authentication" />
</httpModules>
<authentication mode="None" />
<authorization>
<deny users="?" />
</authorization>
</system.web>
</configuration>
```

The element http Modules contains information necessary for ASP.NET run time to locate the module and place it at the interception point for all incoming HTTP requests. The use of value "None" in the authentication mode section instructs the run time to turn off preinstalled ASP.NET authentication modules.

VII.0 CONCLUSION

Securing one's Web services is a vital aspect of ensuring a successful deployment. When deployed externally for consumption by partners or customers, only secure Web services can provide a justifiable integration solution, because the benefits they expose should far outweigh the risks. For true security, one needs to understand the potential security risks and proactively minimize those risks. Using the right tool for the job is important, both in terms of products and technologies. Make sure that every security decision is followed by attention to detail in the implementation and by extensive testing; then one is on one's way to developing Web services that are less vulnerable to attack. WS-Security also addresses message privacy and integrity issues. You can encrypt whole or partial messages to provide privacy, and digitally sign them to provide integrity.

REFERENCES

An Oracle White Paper June 2009, Securing Web Services and Service-Oriented Architectures with Oracle Web Services Manager 11g

Anoop Singhal, Theodore Winograd, Karen Scarfone, Guide to Secure Web Services, NIST Special Publication 800-95

B. Evjen, S. Hanselman, D. Rader, Professional ASP.NET 3.5 In C# and VB, Wrox, ISBN: 978-0-470-18757-9

B. Hartman, Donald J. Flinn, Konstantin Beznosov, Shirley Kawamoto Mastering Web Services Security, Wiley Publishing inc. (2003).

Brown, Keith, Programming Windows Security. Upper Saddle River, Addison-Wesley, 2000.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vsent7/html/vxconIISAuthentication.asp

http://www.w3.org/XML/1999/xml-in-10-points

http://xml.com/lpt/a/2001/04/04/soap.html

IBM and Microsoft. “Security in a Web Services World: A Proposed Architecture and Roadmap”,http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwssecur/html/securitywhitepaper.asp, 2002

Lim Vu, Securing Web Communications“Microsoft .NET Passport.” http://www.microsoft.com/myservices/passport, 2001b

Petr PALAS, Securing Web Services Using Microsoft Web Services Enhancements 1.0, www.PortSight.com

Rami Jaamour, Securing Web Services, www.infosectoday.com.

Securing ASP.Net Web Services with Forms Authentication http://dotnetslackers.com/articles/aspnet/Securing-ASP-Net-Web-Services-with-Forms-Authentication.aspx

Zoran Zaev, Securing Web Services with VB.Net, xmldevleoper Jul, 2002, www.xmldevelopernewsletter.com