Global Information Assurance Certification Paper

Hidden Layers – Challenges in Securing Modern Web Services Solutions

Mark A. O'Donnell

GSEC Assignment 1.4b
Option 1: Research on Topics in Information Security

May 31, 2004

Abstract

Since the turn of the century, industry has been abuzz about Web services. The vision of enterprises of all types linked across the Internet via language and platform agnostic applications is certainly compelling. By 2002, security was identified as perhaps the weakest link in the Web services standards and emerging architectures. The evolution of web services platforms has been supported by 'integrateable stacks'[1] – solutions integrated from layers of products. The flexibility offered this approach comes at a price tag of increased complexity – which is generally not good for security.

This paper focuses on securing Web Services-based solutions in the increasingly complex world of Internet and complex Web Services 'stacks'. It investigates how the fundamental security principle – 'know thy system' – is both increasingly difficult to implement and increasingly important to observe in modern, complex systems. For clarity of exposition, we work within the context of a 'canonical' Web Services solution for eCommerce, but the basic principles apply generally to Web services based solutions and 'integrateable service stacks'.

Introduction

Since the turn of the century, industry has been abuzz about Web services. The vision of enterprises of all types linked across the Internet via language and platform agnostic applications is certainly compelling. Every commercial middleware product has been re-invented by the marketing team as "Web Services based" or "an integrated Web Services Solution" and all serious software development environments -- from Microsoft's Visual Studio .Net to the open source Eclipse project -- support XML, SOAP, and Web Services development.

But by 2002, the industry had identified security as a key area that needs to be addressed for Web services to be widely accepted. By April of that year, Microsoft & IBM had joined forces to produce a "roadmap" for security in a web services world which stated[2]:

> The IT industry has been talking about Web services for almost two years. The benefits of having a loosely-coupled, language-neutral, platform-independent way of linking applications within organizations, across enterprises, and across the Internet are becoming more evident as Web services are used in pilot programs and in wide-scale production. Moving forward, our customers, industry analysts, and the press identify a key area that needs to be addressed as Web services become more mainstream:

[1] 'Integratable' product stacks have been used in Sun Microsystem's marketing in response to Microsoft's 'integrated product suite' approach. While the complexity of the resulting solutions is arguably the same, the 'integratable stack' approach increases the solutions provider's visibility into the solution. Sun CEO Scott McNealy has been quoted as saying, "You get best of breed as opposed to inbred," in his usual Microsoft bashing.

[2] IBM & Microsoft, "Security in a Web ServicesWorld"

> security. This document proposes a technical strategy and roadmap whereby the industry can produce and implement a standards-based architecture that is comprehensive yet flexible enough to meet the Web services security needs of real businesses.

Since that time, solutions providers and standards organizations have continued to struggle with technical strategies, roadmaps, and standards which would provide the comprehensive security necessary to do 'real business' while remaining flexible enough to preserve the original promise of Web Services. By early 2004 the standards bodies had produced a number of "Committee Drafts" for industry review. The wheels of bureaucracy appear not observe Moore's Law, so in the meantime, capitalists being capitalists, Web Services solutions have become ubiquitous; security standards be damned. What have they done?

This paper takes the road less traveled by not focusing on the WS Security Standards and SOAP Extensions and XML Schemas that support them which continue to move through committees. Instead the focus here is on the real-world challenges of Web Services Security; or security in a complex world of the Internet, Web Services, and 'integrateable stacks'. We demonstrate how the basic security principle – 'know thy system' – is more applicable and more difficult than ever. Using an illustrative Web Services solution for eCommerce to provide context, we will see that the 'integrateable stack' concepts extend down into vendor provided products in ways that are not readily visible to the system integrator. So 'know thy system' now means 'know thy vendor's stack' as well.[3] We will show how this complexity contributes to Web-based systems that are increasingly difficult to secure and defend.

An Illustrative Sample – ACME Inc's e-Commerce Portal

To provide context for the following investigations and discussions, we will look at Acme, Inc.'s e-Commerce portal as an illustrative example. Like many businesses Acme has chosen to conduct Web-based business by taking orders from select partners via the Web Services over the public Internet, rather than expend perhaps a significant portion of the 21st Century waiting for the standards bodies to converge on WS Security standard.

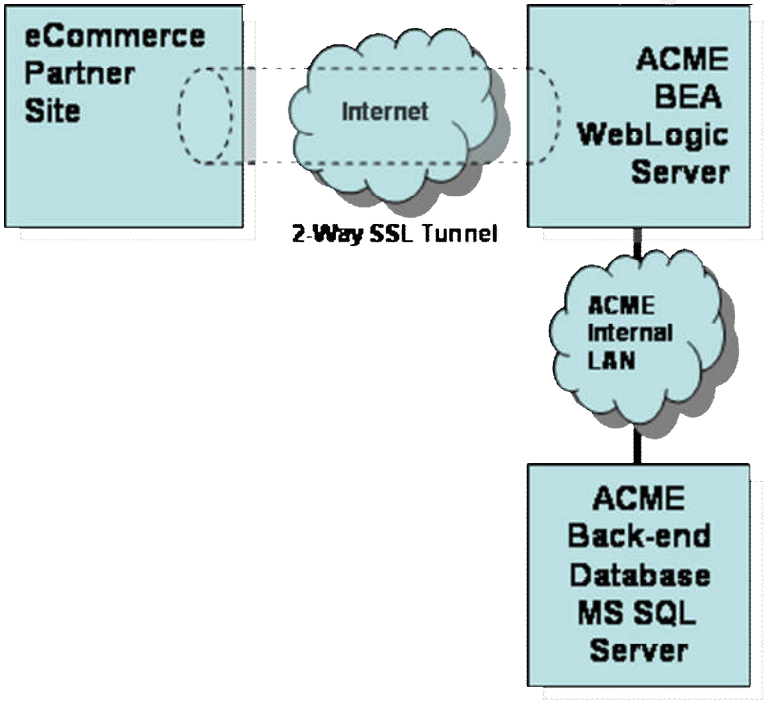
Acme has a wide base of business partners ranging from the U.S. government to foreign companies. The nature of their business imposes some significant security requirements for authentication of customers, non-repudiation of transactions, privacy of data in transit (over the Internet), and protection of the data in their back-end databases. Acme has chosen the following basic building blocks for their solution:

[3] One vendor (who shall remain nameless) responded publicly to a CERT advisory by stating that there was no problem with its product, the problem was with another product that was integrated into its stack and referring all questions to that vendor.

- 2-way Secure Socket Layer (SSL) to support authentication of customers and protect data communications between the ACME site and their partners over the Internet
- BEA's WebLogic Server (WLS) to support Acme's Web Services interfaces and their business logic through a high-end, scaleable industry leader application server that can support their planned growth, and
- A backend RDBMS

as shown in Figure 1.

Figure 1 Acme Inc.'s eCommerce System



At first glance, this system looks simple enough even granting that the BEA WLS is an industrial strength application server loaded with features. The Web Services interactions between Acme and its partners appear to be straightforward:

- Set up an SSL tunnel, which is pretty standard Web technology. (Well, we will see that there are a few little '2-way' details to work through.)
- Define a the necessary Web Services/SOAP messages and publish the WSDL for business partners
- Implement the business rules through servlets and the J2EE capabilities

- Interact with a backend database

These requirements appear to be tailor-made for any modern application server, including BEA WLS. Any salesman will tell you so, and he'd be right. But the salesman might fail to mention some security issues that are worthy of consideration. The challenges facing the security engineer are significant:

- Configuring the Operating System
- Securely configuring the BEA Weblogic Server
- Securely configuring communications with partners through 2-way SSL

Each of the items above is constructed from many other layers; often containing products from other vendors. For example, while the BEA WLS uses third party parsers to support its Web Services functionality, it is very difficult to find information on recent vulnerabilities against Web Services via XML and XML parsers; for example, Cross Site Scripting attacks[4], Cross Site Tracing[5] [6], and denial of service attacks against XML parsers[7].

In the remainder of this paper we illustrate the layering challenge in modern systems by focusing on the potential security vulnerabilities in only one component – 2-way SSL. An examination of the security issues surrounding the use of two-way SSL in the context of the Acme system, will clearly illustrate that 'know thy system' is a non-trivial exercise in the modern world of 'integrateable stacks' of Web Services product offerings. We will look at some issues which we have been unable to resolve with 100% satisfaction, and others of which would require design decisions and trade-offs, not only by the technical staff but also by the business team.

2-Way Secure Socket Layer in BEA WLS

We begin by looking at BEA's approach to 2-way Secure Socket Layer (SSL). The SSL protocol was originally developed by Netscape Communications for transmitting private documents via the Internet. The SSL protocol runs above TCP/IP and below higher-level protocols, such as HTTP. In fact, the state of the Web practice calls for the use of SSL under HTTP, or https, to secure communications and it has been approved by the Internet Engineering Task Force (IETF) as a standard[8]. SSL uses public key cryptography[9] to provide the following features:

- A 'handshake protocol' for the exchange of a shared secret – a symmetric key used to encrypt the data between a server and a client

[4] Champeon, Steve. "XSS, Trust, and Blarney".
[5] BEA has issued a Security Advisory for WLS 8.1 for this vulnerability (BEA04-48.01). The vulnerability is fixed in Service Pack 2.
[6] US Cert Vulnerability Note #VU867593.
[7] BEA Systems, Inc. Security Advisory #BEA02-23.01.
[8] Freier, Alan O., Karlton, Philip, & Kocher, Paul C. "The SSL Protocol".
[9] Burnett, Steve & Paine, Stephen.

- An encrypted tunnel in which data can be transmitted privately and the integrity of the data is ensured,
- Authentication of the server to the client, and
- Optionally, authentication of the client to the server.

When the fourth feature is used, the protocol is typically called “2-way SSL” (for 2-way authentication). On the general Internet, 2-way SSL is not used. The server does not care who is on the other end of the line, only that he or she can present a valid credit card number. But Acme only wants to conduct business with Trusted Partners who must prove their identity through 2-way SSL.

BEA strongly recommends the use of SSL in all operational deployments. The following four steps to setting up SSL on the BEA WebLogic Server are described in the BEA System Administrator documentation[10]:

(1) Obtain an identity (private key and digital certificates) and trust (certificates of trusted certificate authorities)

(2) Store the private keys, digital certificates, and trusted CA certificates.

(3) Configure the Identity and Trust keystores for the WebLogic Server via the Server Administrative Console.

(4) Set SSL attributes for the WLS private key alias and password and the attributes that require the presentation for client certificates for 2-way SSL in the WebLogic Server Administration Console

These steps are representative of SSL configuration generally.

The next four sections briefly discuss each of these steps and recommendations from BEA in more detail. Then we discuss ‘the rest of the story’ – some details overlooked in BEA’s documentation and other known SSL vulnerabilities.

1. Obtain an identity.

Due to the business it does with the US Government, Acme originally decided to obtain a Hardware Protected SSL Certificate from VeriSign[11]. This certificate complies with Federal Information Processing Standard (FIPS) 140-2 which specifies the standards to be used by Federal agencies in protecting sensitive information with cryptography. While Acme was not strictly required to use this standard to do business with the US Government, they chose to FIPS complaint certificates for because the US Government, specifically the Department of Defense (DoD) is their biggest single account. By protecting the private SSL encryption keys in a FIPS validated hardware security module (HSM), Acme laid claim to ‘best security practices’ in its marketing literature.

[10] BEA Systems, Inc. “Managing WebLogic Security/Configuring SSL”.
[11] Verisign. “Hardware Protected SSL Certificate”.

In addition, the DoD Public Key Infrastructure (PKI)[12] requires use of the RSA algorithm with a1024-bit key length and SHA1 with RSA signatures. So Acme complied with these government standards as well even though they did not obtain DoD PKI certificates.

We will discuss some ramifications of these decisions in 'hidden layers' in the following paragraph and in "Other 2-way SSL Configuration Issues" below.

2. Store the private keys, digital certificates, and trusted CA certificates.

WebLogic Server 8.1 stores keys and certs in Java Key Stores. Per BEA recommendations, the XML Gateway stores its private key and the corresponding digital cert are in the Identity keystore. The trusted CA certs are stored in a separate Trust keystore. But wait, if the private key must be stored in a Java keystore, what's the point of the Hardware Protected SSL Certificate?

3. Configure the Identity and Trust keystores for the WebLogic Server via the Server Administrative Console.

So while working through that certificate problem with BEA, Acme purchased a basic SSL certificate from Verisign, and stored the private key and the digital cert in the WLS Identity store and DoD PKI root certificate, specifically to support DoD customers, as well as Verisign's and several other of the major certificate vendors' root certificates in the Trust store.

It's also worth looking into vulnerabilities in Java keystores. The BEA WebLogic Server version 7.0 was vulnerable to a flaw that stored keystore passwords in clear text. That flaw has been patched[13] and no other vulnerabilities in Java Keystores have been found in the Java version distributed with BEA WLS 8.1.

4. Set SSL attributes for the WLS private key alias and password

Set the attributes that require the presentation of client certificates for 2-way SSL in the WebLogic Server Administration Console. This is straightforward and requires no further discussion here[14].

The above steps, as proscribed by BEA, seem straight-forward enough. We have already seen that obtaining an identity creates a dependency between certificates and the BEA WLS that is somewhat hidden. 'Highly secure' certificates did not integrate with BEA's Java Keystores. So again we see that you must 'know thy system and all its components'.

[12] "Draft Federal Public Key Infrastructure (**PKI**) Technical Specification Part D – Interoperability Profiles".
[13] BEA Systems. Security Advisory (BEA03-25.00).
[14] BEA Systems, Inc. "Managing WebLogic Security/Configuring SSL".

Hidden Layers & Potential Vulnerabilities

In the remainder of this paper, we will investigate other layers of the 2-way SSL solution and the potential vulnerabilities they might create, including:

- Proper configuration of 2-way SSL in the WLS. We assume that 2-way SSL has been configured properly via the System Administration Console as above; however there are other, non-BEA specific, issues to consider in configuring SSL.
- Known vulnerabilities of SSL
- Known vulnerabilities with XML parsing, which is fundamental to Web Services

2-Way SSL Configuration Issues

As mentioned above, Acme is very interested in supporting its critical government customers; particularly those in the Department of Defense. The DoD PKI requires use of the RSA algorithm with a1024-bit key length and SHA1 with RSA signatures. The 1024 bit keys are used to authenticate the client and the server and to share a secret key, NOT to secure the data stream. The shared secret is used for the symmetric encryption algorithm that creates the tunnel protecting the data stream.

The server and client negotiate which cipher they will use to communicate via the SSL handshake protocol. The intent of the SSL handshake protocol is that they negotiate the strongest mutually acceptable cipher suite and use it for the SSL session. However some implementations of SSL have instead negotiated the weakest cipher suite. More importantly, a client can attempt to negotiate a weak (or even a NULL) cipher by design. Therefore Acme's server should be configured to use only acceptably strong cipher suites by disabling support for weaker ciphers. So what suites are 'acceptably strong' and how does one disable the others in the BEA WebLogic Server? The following sections show that the issues that this layer are non-trivial.

'Acceptably Strong' Cipher Suites

As mentioned above, DoD PKI requires the use of the RSA algorithm with a 1024-bit key length and SHA1 with RSA signatures. However, it is not the responsibility of the DoD PKI to specify acceptable symmetric ciphers. There is no DoD-specific guidance beyond the general federal guidance to use FIPS standard algorithms. So Acme was on the right track with Verisign and 'High Security' certificates. The current acceptable FIPS algorithms are triple DES (3DES), SKIPJACK[15], and the (relatively new) Advanced Encryption Standard (AES). The BEA WLS currently supports only 3DES. So Acme proceeded to:

[15] SKIPJACK began life as a classified escrowed key system to support the Clipper initiative. Therefore it has not been used commercial applications.

- Establish and document 3DES as the only acceptable cipher for use by its partners,
- Configure WLS to ONLY accept connections using the cipher suite with RSA, SHA1, and 3DES.

But Acme must also support foreign partners with different requirements. Under U.S. law, cryptographic algorithms are munitions and export is controlled under the Export Administration Regulations (EAR) administered by the Department of Commerce's (DoC's) Bureau of Export Administration (BXA)[16]. Until U.S. export regulations were significantly relaxed in 2000, exporting 'strong' symmetric encryption (<56 bit keys), was illegal. Now any cryptographic product is exportable under a license exception (that is, without a license) as long as the end user is not a foreign government or designated country such as Iraq, Syria, North Korea, and Libya.

While the law has been relaxed, cryptographic software still falls under the category of 'munitions' under export law and the U.S. Government has no sense humor about illegal export, 'accidental' or otherwise. The legal risks remain serious enough that, by default, the BEA WLS uses an "exportable" cipher suite -- RSA key exchange, RC4 encryption with an exportable key length of 40, and MD5 digests. The product license key determines the cipher suites that are installed.

So while ACME does not plan to export or provide the BEA WLS or any other software to its foreign customers, it should proceed cautiously in conducting its foreign business. It does not do business with embargoed areas so standards based cryptography products – both exported from the U.S. and produced by foreign companies – are readily available. Acme will not export any cryptographic products. Having established their standards for digital certificates and authentication -- RSA algorithm with a1024-bit key length and SHA1 with RSA signatures – Acme can require foreign partners to obtain appropriate X.509 certificates with corresponding private keys. They can do so via a number of paths. In fact, asymmetric cryptography when used for identity in digital certificates and signatures has never been an export issue. The focus of the export regulations has always been the symmetric algorithms used for data encryption or privacy. With the 2000 law changes, exportable or non-US cryptographic suites, including 3-DES packages, are now readily available to most of the world. So there is something to be said for establishing inter-operable standards across partners.

Configuring the proper cipher suite in BEA WLS (or disabling the others) also proved to be an interesting endeavor. It turns out that the WLS the Admin Console, which is generally a very good tool, does not provide a window to control the allowable symmetric algorithms for SSL. It seems the state-of-the-practice is to simply negotiate the algorithm with the client across a default set of

[16] US Department of Commerce, "Commerical Encryption Export Controls"

algorithms. As discussed above, AMCE has reasons to enforce stronger privacy requirements. It turns out that the only way to set the cipher suites in BEA WLS 8.0 is to manually edit the `config.xml` file[17]. Did we mention that this file is very sensitive and must be secured VERY CAREFULLY?

SSL Vulnerabilities

BEA WebLogic Server supports SSL v3 and TLS v1 tunneling protocols, and X.509 v3 Certificates. TLS is slightly stronger than SSL so, all other things being equal, ACME preferred TLS. However, SSL remains the more common protocol; so ACME chose it as its protocol standard and specified SSL v3 to its business partners.

Recently several vulnerabilities have been exploited in SSL, particularly against OpenSSL. The BEA WLS uses the Certicom encryption libraries. After digging into the Certicom product documentation and reviewing the cert advisories for SSL, it turns out Certicom uses parts of the OpenSSL package. As of this writing, we have been unable to determine exactly which parts of the OpenSSL package Certicom uses, and they have yet to release an update on the CERT site. This is a good example of a primary theme of this paper – 'integrateable stacks' demand that the solution provider 'know the entire stack'. It is often non-trivial to dig out all the relevant information that is spread across numerous vendors.

The following paragraphs summarize several recent SSL vulnerabilities. The intent is to provide a sample of the range of issues the security professional must consider; not to exhaustively review all SSL and OpenSSL vulnerabilities.

1. CERT Vulnerability Note VU#888801[18]

This attack exploits features in PKCS #1 version 1.5 padding used for RSA encryption, and in particular how this PKCS #1 version 1.5 padding was used in the SSL/TLS protocol.

OpenSSL releases up to 0.9.6i and 0.9.7a are vulnerable but this particular attack is really against RSA with PKCS #1 version 1.5, not SSL per se. Certicom is a proponent of Elliptic Curve Cyptography (ECC)[19] which uses key agreement rather than key transport to transmit the 'pre-master secret' and is not vulnerable to this attack against RSA; so they may not respond to this CERT. Neither BEA WLS nor Certicom appear in the note's list of affected systems one way or the other, so this issue should certainly be pursued to ensure that Certicom and BEA WLS are using the patched versions of OpenSSL.

[17] This was re-confirmed by the BEA help desk for BEA WLS 8.1 as of May 2004.
[18] http://www.kb.cert.org/vuls/id/888801
[19] Many good summaries of this technology are available on the Web, for example see http://world.std.com/~dpj/elliptic.html

2. CERT Vulnerability Note VU#288574[20]

A remote attacker could perform a carefully crafted SSL/TLS handshake against a server that used the OpenSSL library and cause OpenSSL to crash. Depending on the application this could lead to a denial of service.

Updated versions of OpenSSL are now available which correct this security vulnerability but Certicom appears in the note's list of affected systems with status "Unknown"[21], so this issue should certainly be pursued to ensure that Certicom and BEA WLS are using the patched versions of OpenSSL..

3. CERT Vulnerability Note VU#484726[22]

This is a flaw in the SSL/TLS handshaking code when using Kerberos ciphersuites which could allow a DoS attack. As discussed above, the Kerberos cipher suites are on the government/DoD approved list, so ACME configuration of SSL in the BEA WLS is configured not to accept them. So this is not an issue as configured. Note however that Certicom appears in the note's list of affected systems with status "Unknown"[23].

4. Vulnerability Note VU#686224[24]

A vulnerability in the way OpenSSL handles invalid public keys in client certificate messages could allow a remote attacker to cause a denial of service. This vulnerability requires as a precondition that an application is configured to ignore public key decoding errors, which is typically only the case during debugging.

This vulnerability exists when an application is configured to ignore public key decoding errors. This configuration item is intended to support debugging and should not be set in any operational system. Open SSL has also been patched for this vulnerability (0.9.7c or 0.9.6k).

Neither BEA WLS nor Certicom appear in the note's list of affected systems one way or the other, so this issue should be pursued to ensure that Certicom and BEA WLS are using the patched versions of OpenSSL.

5. Vulnerability Notes VU#748355, VU#255484, VU#732952, VU#380864, and VU#935264[25]

These notes describe various OpenSSL vulnerabilities involving Abstract Syntax Notation number One (ASN.1). ASN.1 is the international standard used to describe and transmit data packets between applications across networks. The ASN.1 library used by OpenSSL had several parsing errors which allowed

[20] http://www.kb.cert.org/vuls/id/288574
[21] As of March 26, 2004
[22] http://www.kb.cert.org/vuls/id/484726
[23] As of March 26, 2004
[24] http://www.kb.cert.org/vuls/id/686224
[25] http://www.kb.cert.org/vuls/id/748355, /255484, /732952, /380864, /935264

malformed certificate encodings to be parsed incorrectly. Attackers could exploit this vulnerability in a variety of ways to cause a denial of service, potentially execute arbitrary code, or disclose sensitive information.

OpenSSL 0.9.6g removes all known instances of this vulnerability. Neither BEA WLS nor Certicom appear in any of these notes' lists of affected systems one way or the other, so this issue should certainly be pursued to ensure that Certicom and BEA WLS are using the patched versions of OpenSSL.

There are many other OpenSSL vulnerabilities that would need to be considered in fielding an e-Commerce system; especially one that do business with the U.S. Government and international partners. Again, the intent here is to illustrate the challenge facing the security professional; not to be exhaustive.

Conclusion

This paper began by looking at the industry trends toward Web Services and 'integrateable stacks'. To provide context we used a sample electronic commerce system designed based on standards:

- Web Services - the emerging standard for web commerce,
- BEA Weblogic Server - an industry-leading application server platform (BEA WLS), and
- Secure Socket Layer, the state-of-the-practice communications standard.

As one might expect from an industry leader, BEA provided a well-documented and straight-forward, four step process to implement 2-way SSL and 'secure' communications over the Internet including client authentication, non-repudiation, and privacy. But as we peeled away the onion, each layer presented its own set of independent but inter-related challenges:

- Keys and certificates must be chosen with care to meet US Government requirements and conduct business with international partners
- The BEA WLS must be properly configured to store certificates and keys
- Cipher suites must be properly configured to meet US Government requirements and conduct business with foreign partners
- Vulnerabilities in vendor products must be carefully evaluated down thru all layers – the WLS to the Certicom cryptographic libraries to communications through OpenSSL.

The issues associated with these layers, and the layers themselves, are not readily apparent. Vendors are generally not eager to expose this complexity; in fact, the sales force generally does not possess the required technical depth. It is not always clear where to look for information that is distributed across many vendors' sites. So it is up to the developers and security architects of Web

Services based systems to peel back the onion and “know thy system and all of its layers”. Let the buyer beware.

References

BEA Systems Inc. “A vulnerability in the Xerces Parser can cause a potential denial of service”. Security Advisory #BEA02-23.01. URL: http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA02-23.jsp (8 May 2004).

BEA Systems, Inc. “BEA WebLogic Server and WebLogic Express keystore password storage vulnerability”. Security Advisory (BEA03-25.00). URL: http://developer.bea.com/resourcelibrary/advisoriesnotifications/BEA03-25.jsp?printable=true (25 April 2004).

BEA Systems, Inc. “Managing WebLogic Security/Configuring SSL”. BEA dev2dev Product Documentation. WLS Version 8.1.
URL: http://e-docs.bea.com/wls/docs81/secmanage/ssl.html (3 April 2004).

Burnett, Steve & Paine, Stephen. RSA Security's Official Guide to Cryptography. RSA Press-Osborne/McGraw-Hill, 2001.

Champeon, Steve. “XSS, Trust, and Blarney”. WebMonkey- The Web Developer’s Source, 27 Apr 2000. URL: http://hotwired.lycos.com/webmonkey/00/18/index3a.html (8 May 2004).

“Draft Federal Public Key Infrastructure (**PKI**) Technical Specification Part D – Interoperability Profiles”. Federal **PKI** Technical Working Group. 27 September 1995. URL: http://csrc.ncsl.nist.gov/pki/cross.ps (17 April 2004).

Freier, Alan O., Karlton, Philip, & Kocher, Paul C. “The SSL Protocol”. Internet Engineering Task Force (IETF), Transport Layer Security Working Group Internet Draft. Version 3.0. 18 November 1996.
URL: http://wp.netscape.com/eng/ssl3/draft302.txt (17 April 2004).

Heiss, Janice J. “The Future of Web Services Security – A Conversation with Eve Maler”. Sun Developer Web Site, 18 March 2003.
URL: http://java.sun.com/features/2003/03/webservices-qa.html (3 April 2004).

IBM & Microsoft. “Security in a Web Services World: A Proposed Architecture and Roadmap”. A Joint White Paper from IBM Corporation and Microsoft Corporation. Version 1.0. 7 April 2002. URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwssecur/html/securitywhitepaper.asp (3 April 2004).

“Introduction to SSL”. Netscape Communications Corporation. 1998. URL: http://developer.netscape.com/docs/manuals/security/sslin/contents.htm (18 April 2004).

"ITxpo McNealy says Sun still strong". InfoWorld. 8 October 2002. URL: http://ww1.infoworld.com/article/02/10/08/021008hnmcnealykey_1.html (4 April 2004).

King, Christopher M., Dalton, Curtis E., & Osmanoglu, T. Ertem. Security Architecture - Design, Deployment & Operations. RSA Press-Osborne/McGraw-Hill, 2001.

Microsoft, IBM, & Verisign. "Web Services Security (WS-Security)". Version 1.0. 5 April 2002. URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-security.asp (10 April 2004).

"Security Requirements for Cryptographic Modules, Information Technology Laboratory". FIPS Publication 140-2. National Institute of Standards & Technology. 25 May 2001.
URL: http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf (18 April 2004).

US CERT, Vulnerability Note VU#888801. "SSL/TLS implementations disclose side channel information via PKCS #1 v1.5 version number extension". 23 May 2003. URL: http://www.kb.cert.org/vuls/id/888801 (24 April 2004).

US CERT, Vulnerability Note VU#288574. "Null-pointer assignment during SSL handshake". 26 March 2004
URL: http://www.kb.cert.org/vuls/id/288574 (24 April 2004).

US CERT, Vulnerability Note VU#484726. "OpenSSL does not adequately validate length of Kerberos ticket during SSL/TLS handshake ()". March 26, 2004
URL: http://www.kb.cert.org/vuls/id/484726 (24 April 2004),

US CERT, Vulnerability Note VU#686224. "OpenSSL does not securely handle invalid public key when configured to ignore errors". 1 October 2003.
URL: http://www.kb.cert.org/vuls/id/686224 (24 April 2004).

US CERT, Vulnerability Note VU#867593, "Multiple vendors' web servers enable HTTP TRACE method by default". 14 September 2003.
URL: http://www.kb.cert.org/vuls/id/867593 (25 April 2004).

US Department of Commerce, Bureau of Industry and Security. "Commercial Encryption Export Controls". 6 June 2002.
URL: http://www.bxa.doc.gov/Encryption/ (18 April 2004).