

# Security Evaluation of Microsoft .NET Framework and IBM WebSphere

Release 1.0

June 2003

## **mission**

@stake, Inc., the premier digital consulting firm, provides security services and award-winning products to assess and manage risk in complex enterprise environments. @stake consultants combine technical expertise with a business focus to create comprehensive security solutions to mitigate risks and maximize results. As the first company to develop an empirical model that measures Return On Security Investment (ROSI), @stake keeps security investments in line with business requirements.



[www.astake.com](http://www.astake.com)

196 BROADWAY

CAMBRIDGE, MA 02139

USA

MAIN: 617.621.3500

FAX: 617.621.1738



Copyright ©2003 @stake, Inc., Inc.

All Rights Reserved.

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system or translated into any language in any form by any means without the written permission of @stake, Inc..

While every precaution has been taken in the preparation of this document, @stake, Inc., Inc. assumes no responsibility for errors, omissions, or for damages resulting from the use of the information herein. Products or corporate names may be trademarks or registered trademarks of other companies. They are used only for explanation and to the benefit of the owners, without intent to infringe.

# Table of Contents

---

<b>1 Executive Summary</b>	
1.1 Executive Summary .....	1
<b>2 Objectives and Process</b>	
2.1 Goals and Objectives .....	10
2.2 Work Plan .....	16
2.3 Analytical Process .....	20
2.4 Documentation of Findings .....	24
<b>3 Analysis Baseline</b>	
3.1 Microsoft .NET Framework and Windows Server 2003 .....	29
3.2 IBM WebSphere (Linux) .....	32
3.3 IBM WebSphere (Unix) .....	35
<b>4 Findings and Analysis</b>	
4.1 Scorecard .....	39
4.2 Application Server .....	49
4.2.1 Application Logging Services .....	51
4.2.1 Exception Management .....	53
4.2.1 Logging Privileges .....	57
4.2.1 Log Management .....	61
4.2.2 Authentication and Access Control .....	67
4.2.2 Login Management .....	69
4.2.2 Role Based Access Control .....	76
4.2.2 Web Server Integration .....	91
4.2.3 Communications .....	94

4.2.3	Communication Security .....	96
4.2.3	Network-Accessible Services .....	100
4.2.4	Cryptography .....	104
4.2.4	Cryptographic Hashing .....	107
4.2.4	Encryption Algorithms .....	111
4.2.4	Key Generation .....	115
4.2.4	Random Number Generation .....	118
4.2.4	Secrets Storage .....	121
4.2.4	XML Cryptography .....	124
4.2.5	Database Access .....	128
4.2.5	Database Pool Connection Encryption .....	130
4.2.5	Data Query Safety .....	134
4.2.6	Data Validation .....	142
4.2.6	Common Validators .....	145
4.2.6	Data Sanitization .....	150
4.2.6	Negative Data Validation .....	156
4.2.6	Output Filtering .....	161
4.2.6	Positive Data Validation .....	165
4.2.6	Type Checking .....	170
4.2.7	Information Disclosure .....	175
4.2.7	Error Handling .....	177
4.2.7	Stack Traces and Debugging .....	180
4.2.8	Runtime Container Security .....	182
4.2.8	Code Security .....	184
4.2.8	Runtime Account Privileges .....	206
4.2.9	Web Services .....	211
4.2.9	Credentials Mapping .....	213
4.2.9	SOAP Router Data Validation .....	219
4.3	Host and Operating System .....	229
4.3.1	IP Stack Hardening .....	230

4.3.1 Protocol Settings .....	232
4.3.2 Service Minimization .....	240
4.3.2 Installed Packages .....	242
4.3.2 Network Services .....	248
4.4 Web Server .....	253
4.4.1 Architecture .....	254
4.4.1 Security Partitioning .....	256
4.4.2 Authentication .....	261
4.4.2 Authentication Input Validation .....	264
4.4.2 Authentication Methods .....	269
4.4.2 Credential Handling .....	276
4.4.2 Digital Certificates .....	281
4.4.2 External Authentication .....	289
4.4.2 Platform Integrated Authentication .....	295
4.4.3 Communication Security .....	301
4.4.3 Session Encryption .....	303
4.4.4 Information Disclosure .....	311
4.4.4 Error Messages and Exception Handling .....	313
4.4.4 Logging .....	317
4.4.4 URL Content Protection .....	326
4.4.5 Session Management .....	338
4.4.5 Cookie Handling .....	340
4.4.5 Session Identifier .....	344
4.4.5 Session Lifetime .....	348
<b>5 Coda</b>	
5.1 References .....	352
5.2 About the Authors .....	353
5.3 Colophon .....	355



## 1.1 Executive Summary

---

### Executive Summary

#### Overview

@stake, Inc., an independent security consulting firm, performed an extensive analysis comparing security in the .NET Framework® 1.1, running on Microsoft® Windows® Server 2003, to IBM WebSphere® 5.0, running on both Red Hat Linux Advanced Server 2.1 and a leading commercial distribution of Unix.® This report gives customers a resource for better understanding the differences between the leading application platforms and for guidance on best security practices.

Overall, @stake found that:

- Both platforms provide infrastructure and effective tools for creating and deploying secure applications
- The .NET Framework 1.1 running on Windows Server 2003 scored slightly better with respect to conformance to security best practices
- The Microsoft solution scored even higher with respect to the ease with which developers and administrators can implement secure solutions

The overall differences between the platforms were not large. Based on the results of this study, companies should feel comfortable deploying web and web service applications on WebSphere and on Windows Server 2003 with .NET Framework. In addition, our analysis shows that with appropriate processes and training, applications created for the Microsoft solution can be made as secure as those created for IBM WebSphere.

As a result, @stake recommends that:

- Development organizations that are experimenting with Microsoft Windows Server 2003 and .NET Framework should not allow security concerns to hold back deployments
- Companies that have made strategic commitments to either platform need not switch solely on the basis of security
- Companies seeking to standardize on a web application platform will derive significant security benefits from both, but may find the Microsoft solution easier to secure initially

The balance of this section provides an overview of @stake's methodology, findings, and conclusions.

#### Introduction

The Microsoft Corporation (Microsoft) provides software and services for personal and business computing.

The recently released Windows Server 2003 and .NET Framework Version 1.1 provides an environment for building and deploying Web-based applications and XML Web services.

While the Internet and networking provide enhanced business opportunity, online services and applications dramatically increase the risk of information disclosure and service compromise. Microsoft has stated that the security of its customers and partners is of critical importance. By evaluating the security of Windows and the .NET Framework, Microsoft sought to determine how the overall security posture compares with that of competitive frameworks such as Java® 2 Enterprise Edition (J2EE).

To that end, and in response to customer requests, Microsoft engaged @stake, a leading independent security consulting firm, to perform a competitive security analysis of the .NET Framework versus IBM's WebSphere implementation of the J2EE framework. @stake works with leading companies to improve their abilities to design, develop, test, and maintain the security of their applications and networks. @stake is qualified to be authoritative on the subject of application security best practices. As a firm, @stake has completed several hundred network and application security assessments for clients. @stake is a trusted advisor to six of the world's top ten financial institutions, four of the top ten independent software companies and seven of the world's top ten telecommunication service providers. @stake's published works on application security, notably *The Security of Applications, Not All Are Created Equal*, have been cited by such diverse sources as CIO, The Economist, Forbes, Information Week, CERT/Carnegie-Mellon, The SANS Institute, and Ernst and Young. The @stake project team spent over 1500 man-hours on the competitive analysis, with over 100 test cases applied to both products.

For the analysis, @stake compared three platforms:

- .NET Framework 1.1, running on Windows 2003 Server, hereafter referred to as ".NET Framework" or ".NET"
- IBM WebSphere Application Server 5.0, running on Red Hat Linux Advanced Server 2.1, hereafter referred to as "WebSphere"
- IBM WebSphere Application Server 5.0, running on the current version of a leading commercial vendor of Unix, hereafter referred to as "WebSphere"

@stake evaluated the level of effort required for developers and system administrators to create and deploy solutions that implement security best practices, and to reduce or eliminate most common attack surfaces. In combination with the application servers, we used each respective vendor's development tools and databases:

- Microsoft Visual Studio .NET 2003 and Microsoft SQL Server 2000 for the .NET Framework
- IBM WebSphere Studio Application Developer 5.0 and IBM DB2 Universal Database Version 7.2

Our analysis focused on the level of security provided by default in the platforms, the security features provided by the platforms, and the level of effort required from developers and administrators to implement best security practices using those features.

## Work Plan

During the first two weeks of the engagement, the @stake team developed a comprehensive work plan and analytical process for assessing the security features included with the .NET Framework and WebSphere

deployments. @stake analyzed the level of effort required to minimize all inherent attack surfaces present in out-of-the-box installations of the base operating systems, web servers, and application server environments.

The analytical process also defined techniques for measuring the level of effort and complexity of developing secure applications and web services. For each platform, we defined three high-level requirement categories — application server, web server and host/operating system — each with several major areas of analysis:

- Application Server
  - **Application Logging Services** allow applications to audit security events, errors, unexpected events, and session metrics
  - **Authentication and Access Controls** restrict access to content to designated users
  - **Communications** security protects against host-level system attacks, and prevents exploitation of tunneled communication streams
  - **Cryptography** enables applications to communicate in an undecipherable context, thereby preventing information compromise
  - **Database Access** controls prevent modification, addition, deletion, and theft of data from back-end repositories
  - **Data Validation** ensures that data used for input and output cannot be used to compromise servers or poison information stores
  - **Information Disclosure** refers to the inadvertent leakage of server or configuration information, which may aid in attacks
  - **Runtime Container Security** controls user, group, and code permissions for applications running in the server runtime environment
  - **Web Services security** provides an infrastructure for machines to communicate securely in a platform- and language-independent manner
- Host and Operating System
  - **IP Stack Hardening** eliminates or minimizes available lower-level protocol attacks
  - **Service Minimization** reduces the number of network services listening on a network interface to the absolute minimum
- Web Server
  - **Architecture** permits clean separation of externally facing services and protocols from policies, and prevents information disclosure and compromise
  - **Authentication** guards against un-privileged access to application functionality.
  - **Communication Security** enables the web server to communicate with the application server securely
  - **Information Disclosure** includes inadvertent leakage of server or configuration information, which may aid in attacks
  - **Session Management** permits applications to uniquely track users, and helps ensure that only authorized access the application.

The areas of analysis were further broken down into forty-five (45) distinct topics enumerated in the table of contents. Each topic contained a set of “best practices” that defined the security benchmark for the area, based on security areas that are frequently implemented incorrectly. We defined sixty-seven (67) best practices in total. Finally, each best practice included several test cases designed to measure each platform’s compliance. We defined one-hundred and three (103) test cases in all.

To score the best practices and test cases, we developed a scorecard system to quantify each platform’s:

- Compliance with best practices
- Level of effort to secure a solution, based on the effort associated with completing each test case. Factors affecting level of effort, as we define it, include:
  - Implementation complexity
  - Quality of available documentation and sample code made available from the vendors
  - Developer/administrator skills needed
  - Time to implement

Collectively, the areas of analysis, best practices, test cases and scoring system defined the analytical process for the assessment.

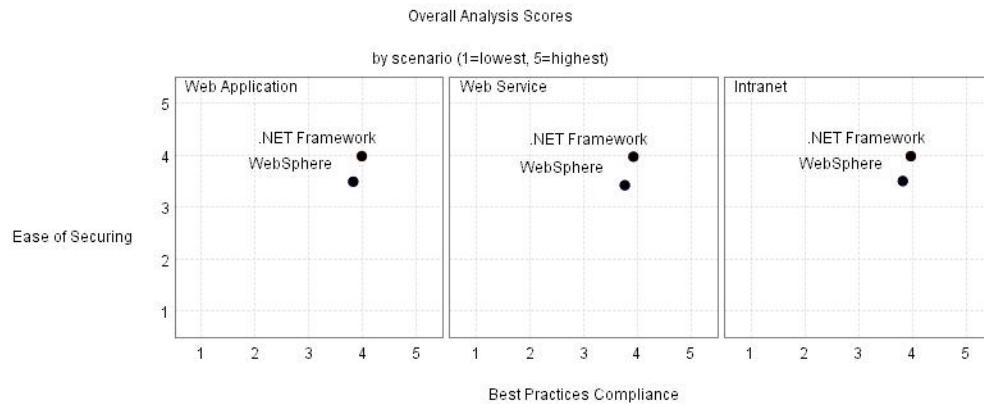
To ensure objectivity, analysts focused on security from three leading technology analyst firms validated the methodology, reviewed the comparative strategy, and provided recommendations and suggestions.

After the completion of all test cases, we calculated overall Best Practice Compliance and Ease of Securing scores for the .NET Framework and WebSphere platforms based on the scorecard results.

In addition, we created weighted scores for three scenarios: web application, web service, and intranet application. Each security area we analyzed received more or less weight depending on the scenario. For example, the security features of SOAP routers are weighted as important for the web service scenario, but are irrelevant to the web application or intranet scenarios. The scores for each scenario, therefore, reflect how well each platform implemented the features appropriate for actual web and web service applications.

## **Findings**

Based on the results of our analysis, @stake scored the .NET Framework higher than WebSphere running on either Unix or Linux. After weighting the raw scores for the three scenarios we considered, .NET Framework running on Windows Server 2003 edged out WebSphere by a narrow margin. The graph below summarizes the scores; each metric is on a five-point scale. For both metrics, one (1) is the lowest possible score, and five (5) is the highest.



@stake found that .NET Framework and WebSphere are strong, credible platforms for developing web applications and web services, as evidenced by the scores for both platforms. Microsoft scored slightly higher on Best Practice Compliance; scores ranged from 3.9-3.96, compared to 3.71-3.8 for WebSphere. For Ease of Securing, the differences were more pronounced: Microsoft's scores consistently came in at or around 4.0, while WebSphere's were nearly a half-point lower (3.44-3.52). We graphed the WebSphere Linux and WebSphere Unix scores together on all three charts, since the differences between the two platform scores were only a few hundredths of a point. For details on the numerical scores, see the Scorecard.

In the next sections, we discuss the strengths and weaknesses of the Microsoft .NET Framework and IBM WebSphere.

### Microsoft .NET Framework

Microsoft Windows Server 2003 with .NET Framework 1.1 provides a strong foundation upon which developers can build secure web applications and web services. @stake found that the .NET Framework scored higher than either of the IBM WebSphere platform combinations we tested. This was true for both best practice compliance and level of effort to secure.

Compared to J2EE, the .NET Framework is a much younger platform and benefits from a comparatively cleaner sheet of paper; it was designed more recently, and has fewer APIs and legacy features. the .NET Framework implements many of the good things that have long been part of the Enterprise Java world, such as a managed runtime environment, separation of web- and application-tier security, role-based access control and declarative privileges. Microsoft has done an effective job implementing these and other features, in addition to leveraging the natural strengths inherent in a more homogeneous application, web, and operating system stack. We believe these advantages contributed to Microsoft's higher score for the best practices compliance compared to WebSphere.

Microsoft also scored better overall with respect to the level of effort required to create secure applications. Areas of particular strength for the .NET Framework (defined as a score of 4 out of a maximum of 5, or higher, for both metrics) included:

- Application server

- Application logging services, particularly log management (albeit only in a Windows-only environment). Microsoft's implementation made application logging simple and effective, requiring minimal coding or environment configuration changes.
  - Integration of web server security with the application server. As might be expected, IIS works well with the .NET framework, resulting in a nearly seamless security integration of web-tier access controls with those of the application server.
  - Support of cryptographic algorithms, particularly those used for generating keys. .NET Framework supports a wide variety of standard cryptographic suites and key generation routines, and makes them convenient for developers to use.
  - Validation of user-submitted input, including validation of data types and “positive” data checking. Microsoft provides user-input filtering in the ASP.NET component of the .NET Framework and includes many pre-configured filters for commonly occurring input requirements.
  - Minimal information leakage when handling errors, with stack trace information hidden from users. Microsoft has largely eliminated the spurious information disclosed by typical error conditions and messages throughout the .NET Framework.
  - Minimal privileges required by the application server at runtime. By default, the application executes with the minimum required privileges to accomplish its tasks. This is in keeping with best practices.
  - For web services, extensibility to support validation of SOAP data. @stake's evaluation team found it relatively easy to apply check user-submitted data within SOAP messages. Since Web services security is a relatively new research frontier, proactive input validation controls are essential safeguards against emerging threats.
- Host and operating system
    - A good default security posture, with few installed packages and attack surfaces. Microsoft restricts the number of services that are “turned-on” by default. This has greatly reduced the number of toe-holds available to a potential attacker.
  - Web server
    - Excellent separation of security policy (as declared by the application) from implementation (as deployed in the infrastructure).
    - Good support for multiple methods of authentication, including Basic, Digest, Passport, Certificate-based, and integrated Windows authentication. Microsoft provided sound documentation and easy-to-use tools to select the most appropriate authentication methods.
    - External authentication through integration with Active Directory and with the Windows operating system.
    - Good support for client-side digital certificates. Microsoft provided concise, straightforward tools and sound documentation on the handling of certificates.
    - Minimal information leakage when handling errors or logging server activity.
    - Effective session management, particularly with respect to using non-predictable session identifiers. The sessions also expire quickly by default.

The strengths of the .NET Framework on Windows Server 2003 far outweighed its weaknesses. We found only one area where it scored of 2.5 or lower in both categories, namely cryptographic key storage. The Data Protection API (API) — which provides storage facilities for secret keys — is not as accessible or as well documented as it might be.

Overall, we found that Microsoft's traditional focus on ease of use, documentation, and general fit-and-finish contributed to its higher scores compared to WebSphere. Much of the sample code and documentation was excellent, and will serve its customers well.

### IBM WebSphere J2EE

IBM WebSphere, like the .NET Framework, provides a strong foundation for building modern, secure applications using web and web services technologies. We rated WebSphere slightly lower than the .NET Framework in most of the areas we analyzed, both with respect to compliance with best practice and to the level of effort required to secure the platform. We believe this is for two reasons. First, WebSphere, as a platform, is composed of a number of packages that do not all originate with IBM. The application platform, for example, uses an HTTP server from Apache, a proprietary SSL stack, an open source SOAP router (Apache SOAP) and code from places like Junit.org, the Jakarta Project, and IBM itself. Because of the number of moving parts, the security model is more complex.

WebSphere also tries to be a good “open” citizen; it works with multiple operating systems, web servers, back-end databases and identity stores. This makes integration harder, and implies the second problem: documentation. IBM is simply not as focused as Microsoft in helping developers create secure applications. While IBM has done an admirable job pulling together quite a bit of material, most of the documentation reflects a bias towards securing infrastructure rather than *code*. IBM should do more to decrease the level of effort required by developers to write secure WebSphere code.

Areas of strength for WebSphere, using the same criteria as for the .NET Framework, included:

- Application server
  - Application logging processes can be implemented securely and with minimal privileges. WebSphere log management proved to be flexible, and logging privileges were very secure by default.
  - Mature, declarative role-based access control model. Role-based access control is easily configured secure out-of-the-box, and supports protected content by role.
  - Readily-implemented support for output filtering, which helps mitigate “cross-site scripting” attacks. JSP tags libraries provided with WebSphere automatically escape sensitive HTML characters when presenting content.
  - Limited information leakage when handling stack traces, with information hidden from users. Hiding servlet errors from the end-user minimizes the information an attacker can use to focus an attack.
  - For web services, validation of SOAP data. @stake readily implemented controls to validate user-submitted data passing through the SOAP router.
- Web server

- Excellent separation of security policy (as declared by the application) from implementation (as deployed in the infrastructure). WebSphere enforces the J2EE servlet specifications, which protects sensitive application configuration files. The web server also runs as a non-privileged user by default.
- Effective session management, particularly with respect to using non-predictable session identifiers. @stake found it straight forward to configure the server session attributes, particularly with respect to the use of secure cookies.

WebSphere had many strong points. Two areas, by the numbers, appear to represent areas of weakness: application server exception management and database access. On an out-of-the box basis, @stake found that it was relatively more difficult to get applications to log events. While various packages can do this (such as Java 2 Standard Edition 1.4's native logging classes, or the ubiquitous Log4J package), IBM's documentation was vague and focused on logging for the server container rather than user-developed applications. With respect to database access, session encryption of database connections was problematic without the installation of additional third party software.

One additional potential problem surfaced during the code security testing portion of the analysis: it appears that WebSphere may not fully implement the Java 2 security specification. In particular, WebSphere does not appear to honor security policy file declarations that require code to be signed. IBM provided no documentation on this functionality; several days of exploration proved fruitless. Support for code-signing is either an incredibly well-kept secret — or just not supported. In either case, IBM should remedy this problem immediately, or provide documentation on how to implement code-signing policies correctly.

With respect to the security features offered in WebSphere, it is worth noting that one of IBM's weaknesses (reliance on external packages, many of them open source) is also a tremendous source of strength. IBM's special style of embrace-and-extend has enabled the WebSphere platform to rapidly add functionality. The WebSphere Studio Application Developer (WSAD) IDE's core, Eclipse, has been donated to the open source community and has been extended with hundreds of aftermarket plug-ins. WSAD shrewdly leverages open source Java tools that we used in the security analysis, like Apache SOAP and the Apache Jakarta Project's Struts framework, a Model-View-Controller (MVC) implementation.

For developers using WebSphere to develop secure applications, IBM's initial integration of the WebSphere Studio IDE with the Struts framework is impressive, if marred by lack of IDE support for Struts Validator, a forms-validation module. In our view, Validator is a hidden gem for developers seeking to safeguard against hostile user input. The other aspects of the integration of Struts into WebSphere Studio provides a glimpse of what IBM could have done competitively, if only it had focused better.

Companies should bear in mind that one area where Microsoft enjoys a momentary advantage, SOAP security, is a rapidly moving target. Both vendors are gearing up to support WS-Security and the as-yet unratified WS-I Basic Profile 1.0 specification, which proposes SOAP extensions for message authentication, signing, and confidentiality. Microsoft has recently released the Web Services Enhancements 1.0 for the .NET Framework. IBM, for its part, is shipping a "technology preview" of the web services features in the upcoming J2EE 1.4 specification, which integrates the J2EE security model with the WS-I Basic Profile. Customers should carefully weigh the costs and benefits of using the web services toolkits in the versions of the products we evaluated. Newer versions, once they have been shown to be compliant with the WS-I Basic Profile, may be a better choice.

## Conclusions and Recommendations

@stake found that both the .NET Framework and WebSphere platforms provide comprehensive tools and infrastructure for building secure applications. Both platforms scored well across nearly all of the security areas we analyzed, with the caveat that web services security is not yet standardized, and support in the platforms for them is still relatively immature.

Although we found that the .NET Framework scored higher overall, the differences were not large. IBM would have increased its scores if there had been more attention paid to helping developers write secure *applications*. As a result, companies may find the .NET Framework to be an easier web application platform to secure initially. For firms deciding on which framework to go with, the right choice largely boils down to factors unrelated to security: namely, the existing developer skill base, and the importance of single-vendor versus open technologies.

The remaining sections describe the methodology we used for the analysis, and contain our findings in detail.

*This document was published in June 2003.*

*Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.*

*WebSphere is a registered trademark of IBM Corporation in the United States, other countries, or both.*

*Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.*

*UNIX is a registered trademark of The Open Group in the United States and other countries.*

*Other company, product and service names may be trademarks or service marks of their respective owners.*

## 2.1 Goals and Objectives

---

### Goals and Objectives

#### Introduction

Microsoft engaged @stake to perform a competitive, independent analysis of the security of two modern web application platforms: Microsoft Windows 2003 with .NET Framework 1.1 and IBM WebSphere Application Server 5.0. During the engagement, @stake analyzed the degree to which each platform adhered to accepted best practices for security, provided strong “out-of-the-box” security, and facilitated rapid implementation through the platforms’ respective Integrated Development Environments.

Identifying “best practices” for web application platforms is difficult. Web applications differ significantly from traditional standalone applications. Most web applications are characterized by multiple tiers, highly decoupled application interfaces and myriad points of security integration. With so many moving parts, any discussion of best practices requires clarification of what a “web application” and “platform” are.

#### Web Application

For the purposes of this analysis, we take the term *web application* to mean a body of distributed software components functioning together as a single coherent application. An idealized view of a web application includes these components:

- **Web server.** Serves static content such as images and HTML. The web server is responsible for serving content requested by the user, and for forwarding dynamic requests to the application server. The web server is typically placed in a public demilitarized zone. *Examples: Internet Information Server, IBM HTTPD (a rebranded version of Apache HTTPD)*
- **Application server.** Fulfills dynamic requests made by the web server on behalf of the user. The application server is typically placed in protected zone behind a firewall, and contains these components:
  - **Dynamic server pages.** Provides interfaces to dynamic runtime objects using a combination of page markup and script code. *Examples: ASP.NET, Java Server Pages (JSP)*
  - **Runtime objects.** Implement the application’s business logic, based on actions triggered by the front-end (web server) or back-end (database server). *Examples: .NET Framework assemblies, Java servlets, Enterprise Java Beans, plain old Java objects (POJOs)*
  - **Managed code runtime container.** Provides a sandboxed execution environment for in-memory runtime objects, and mediates between the objects and the underlying operating system. The runtime container enforces runtime privileges to prevent code from obtaining unauthorized privileges.

*Examples: Microsoft .NET Framework's Common Language Runtime (CLR), IBM WebSphere Application Server (Java Virtual Machine plus servlet and EJB container)*

- **Database server.** Stores application data, typically in a dedicated relational database management system (RDBMS) running on a separate server. *Examples: Microsoft SQL Server, IBM DB2, Oracle*
- **Identity store.** Contains the identities of users that authenticate to, and use, the web application. Identity stores can be local (on the web or application server) or external (centrally administered on a separate server and accessed through industry-standard protocols such as LDAP). *Examples: Active Directory, Sun ONE Directory Server*

### Web Application Platform

The term *web application platform* (or *platform*) is shorthand for all of the components needed to implement and run a web application: the web server, application server, managed code runtime container, database server, and identity store. Most modern web application platforms provide framework services that provide presentation, web, user session management, back-end connectivity, security, and other common functions. The platform's software and services collectively provide the foundation for user-developed runtime objects, server pages, and integration code. The platform:

- Provides a method and context in which code executes
- Ensures the integrity and security of user-supplied objects while system processes they execute
- Provides standardized services, software application programming interfaces (APIs), and convenience classes for common functions
- Exposes methods for interacting with underlying operating system resources

Platforms generally also have an associated Integrated Development Environment (IDE), through which developers create, deploy and debug compliant web application code. Examples of IDEs include Microsoft Visual Studio .NET, IBM WebSphere Studio Application Developer and Sun ONE Studio.

### Best Practices

For the purposes of this analysis, the term *best practices* refers to the methods and techniques used to build and deploy secure web applications on a platform, based on the definitions of *web application* and *platform* from the preceding paragraphs. By *secure* we mean that the application resists attack from common threat vectors, and minimizes the number of attack surfaces available to an opponent. Secure applications minimize the number of failure modes.

@stake is qualified to be authoritative on the subject of application security best practices. As a firm, @stake has completed several hundred network and application security assessments for clients. We are a trusted advisor to six of the world's top ten financial institutions, four of the top ten independent software companies and seven of the world's top ten telecommunication service providers. Our published works on application security, notably *The Security of Applications, Not All Are Created Equal*, have been cited by such diverse sources as *CIO*, *The Economist*, *Forbes*, *Information Week*, CERT/Carnegie-Mellon, The SANS Institute and Ernst and Young.

The best practices matrix used in this analysis is largely based on @stake's internal knowledge base of network and application security best practices, which in turn is culled from client engagement findings data. In addition, to guide our approach we have also relied on documented practices from industry groups such as the Open Web Application Security Project (OWASP).

@stake's documentation of best practices is designed to keep the analysis relatively straightforward in terms of tone, nomenclature and structure. We did not use the kind of documented formalism that one might find in official standards works such as the Common Criteria.

## Goals and Objectives

### Security Best Practices Analysis

@stake evaluated the relative recognized security components common to all web-based application server environments:

- Code Access Security provided by both the .NET Framework's common language runtime (CLR) and the Java Virtual Machine (JVM) running under control of a Java Authentication and Authorization Service (JAAS) security policy
- Declarative Role-Based Security, Authentication, Authorization and Access Control features supported in, and implemented by, the .NET Framework CLR and the WebSphere EJB container
- Auditing and Logging functions available to applications running in the CLR and the WebSphere container
- Communications Security including functionality provided by Secure Socket Layer (SSL), Transport Layer Security (TLS) and Internet Protocol Security (IPSEC)
- Cryptographic Security Functions provided by the .NET Framework, in conjunction with the Windows Data Protection API (DPAPI) and the Java Cryptography Extension (JCE) as integrated into the Java 2 Software Development Kit, Standard Edition, Version 1.4.
- Web forms as supported by ASP.NET and JSP
- Data Integrity and Access provided by ADO.NET and the Java Database Connectivity (JDBC) specification
- Availability and comparative analysis of the granularity of access control mechanisms in the .NET Framework and WebSphere

As part of the product comparison, @stake examined the web and application stacks of each platform. The two figures below depict the conceptual architecture of Microsoft Windows Server 2003 with .NET Framework and IBM WebSphere Application Server, respectively. The diagrams are meant to give a visual comparison between the functional components of the two platforms, and are not necessarily comprehensive.

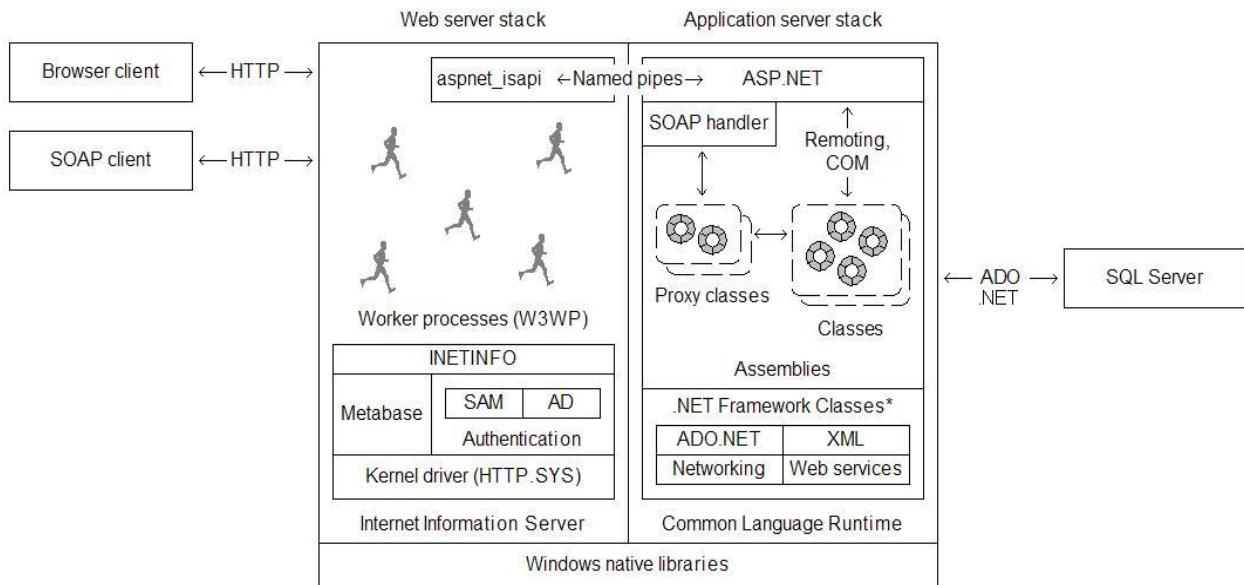


Figure 1: Microsoft Windows Server 2003 with .NET Framework Architectural Diagram

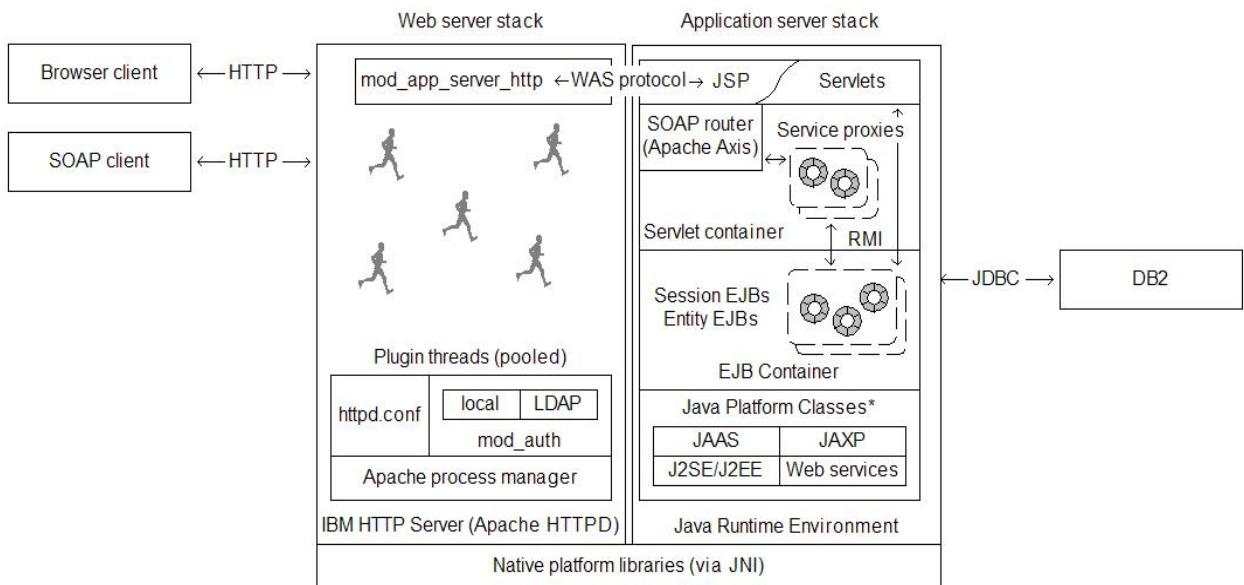


Figure 2: IBM WebSphere J2EE Framework Architectural Diagram

Both platforms are similar architecturally. Each has separate tiers for static HTML serving (IIS v. IBM HTTP Server), web tier dynamic server pages (ASP.NET v. JSP/servlets), managed application runtime (CLR v. J2EE container) and back-end database access (ADO.NET v. JDBC). A full explanation of the architectural components in each platform is beyond the scope of this analysis; see each vendor's documentation for more details.

### Default Security Posture Analysis

@stake evaluated the default degree of security provided by each platform at the time of installation. For the analysis, @stake:

- Identified and enumerated any available remote or external attack surfaces, including open network sockets and network services
- Identified and enumerated any local attack surfaces including avenues for privilege elevations, executables that run with elevated privilege, services that impersonate users of higher privileges, or inappropriately protected objects
- Analyzed the security consequences arising from the interactions and integration of base operating systems and applications servers

### Network Security Level of Effort Analysis

@stake evaluated the relative difficulty and level of effort required to hardening the base operating systems and application server environments for deployed applications. During its analysis, @stake:

- Compared the availability of tools, documentation and best practices on hardening. Sources of documentations were limited to sites maintained by the vendors
- Enumerated and analyzed the security implications of default system and application server user accounts
- Analyzed any inappropriate or insecure object permissions associated with the products, principally with respect to permissions and access-control mechanisms in place upon files, directories, devices, or registry entries. @stake assessed the level of effort required to minimize the security threats posed by insecure object permissions
- Analyzed the remote or externally available services installed by default on the host operating systems, as well as services started or created as a result of installation of application servers. @stake assessed the level of effort required to minimize the security threats posed by inappropriate exposure of external services
- Enumerated and analyzed any management interfaces exposed by the default installation of any component that could lead to a compromise of the products
- Analyzed the changes or modifications required to the base IP stack in order to eliminate or to minimize available attack surfaces. @stake assessed the level of effort required to eliminate or minimize the security threats posed by inappropriate or insecure IP stack configurations
- Assessed and analyzed the modifications required to remove extraneous software packages and functionality from any of the components included with the products. @stake assessed the level of effort required to perform the modifications needed to decrease attack surfaces and improve overall security
- Assessed the level of effort required to implement network access restrictions and policy-based remote access controls

### Application Security Level of Effort Analysis

@stake evaluated the level of effort and relative complexity of developing secure applications and web services using the exposed security functionality of the Products. In order to provide an objective set of evaluation criteria, typical web application security requirements were evaluated. The evaluation focused on

security areas that are frequently implemented incorrectly. Each focus area was evaluated via the creation and execution of several test cases and coding examples to validate analysis results. The areas of focus included:

- User input validation functionality. Input filtering routines were exercised using traditional web applications as well as web service-based applications
- Authorization and access control Application Programming Interfaces (APIs)
- Session management functionality
- Back-end database access classes, and pool connection security
- Exception and error handling functionality
- Cryptography functionality
- Object-level access controls for use in web applications and web services
- Secure deployment of web applications and web services

## 2.2 Work Plan

---

### Work Plan

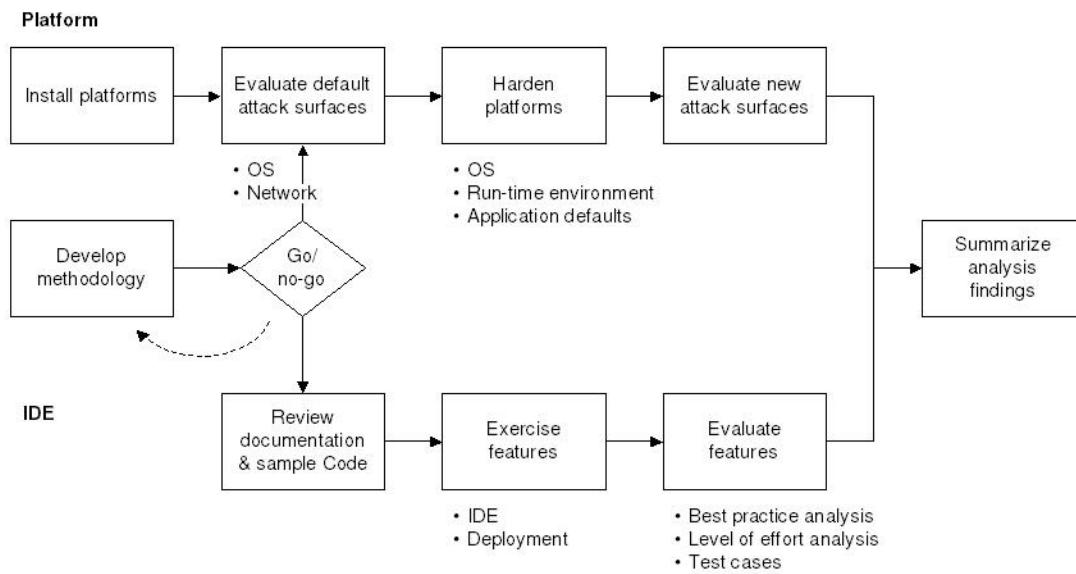
#### Kickoff and Team Structure

@stake and Microsoft jointly planned the execution of the security analysis. The analysis consisted of approximately 1500 man-hours and was conducted by the following @stake consultants:

Name	Project Responsibilities	@stake Role
Andrew Jaquith	Principal author WebSphere team leader Project tools	Program Director
Chris Wysopal	Principal author .NET team leader	Director of Research and Development
Frank Heidt	Principal author Analysis team leader	Managing Security Architect
Brad Arkin	Cryptography analysis	Managing Security Architect
Anthony Barkley	Platform analysis	Managing Security Architect
Fred Bret-Mount	Application analysis	Managing Security Architect
Jason Chan	Platform analysis	Managing Security Architect
Paul Clip	Application analysis	Managing Security Architect
Peter Krautscheid	Application analysis	Managing Security Architect
Todd Lefkowitz	Project Manager	Project Manager
Patrick Mullen	Platform analysis	Senior Security Architect
Kevin Soo Hoo	Methdology and metrics	Senior Security Architect

#### Key Tasks

Prior to the beginning of the analysis phase, @stake developed a detailed work plan for the setup, testing, evaluation and write-up tasks required to complete the analysis. The diagram and descriptions below summarize our approach.



### Install Platforms

- Evaluate and harden three platform environments, each consisting of two computers — one hosting the database, and the other hosting the web server and application server
- Install two client Integrated Development Environments: Visual Studio .NET 2003 and IBM WebSphere Studio Application Developer Version 5 (based on Eclipse)

Details on the architecture and configurations we used are described in the Analysis Baseline section.

### Develop Methodology

- Outline approach and specific stages for platform comparison
- Describe areas of analysis and test cases to be used in each platform evaluation
- Research and determine a level-of-effort metric and criteria

Details on the methodology we developed are described in the Analytical Process section.

### Go/No Go

- Methodology description delivered to Microsoft and 3rd-party security industry analysts for review
- Microsoft decides whether to go forward with the study

### Platform Analysis Activities

#### Evaluate Default Attack Surfaces

- Identify and enumerate all remote or external attack surfaces, including open network sockets and network services

- Identify and enumerate all local attack surfaces, including avenues for privilege elevations, executables that run with elevated privilege, services that impersonate users of higher privileges, or inappropriately protected objects
- Stipulate the security consequences arising from interactions among and integration of base operating systems, applications server, and database engines

#### **Harden Platform**

- Compare available tools, documentation, and best practices on hardening from sites maintained by the products' vendors
- Analyze and determine level-of-effort to minimize inappropriate or insecure object permissions and access-control mechanisms in place upon files, directories, devices, or registry entries
- Determine level of effort to minimize the security threats posed by inappropriate exposure of external services
- Analyze and determine level of effort for changes to the base IP stack to eliminate or to minimize available attack surfaces
- Assess and calculate level of effort required to implement modifications to remove extraneous software packages and functionality from any components, decreasing attack surfaces
- Assess and calculate level of effort required to implement network access restrictions and policy-based remote access controls

#### **Evaluate New Attack Surfaces**

- Identify and enumerate all remaining remote or external attack surfaces after hardening
- Identify and enumerate all local attack surfaces after hardening
- Stipulate any remaining security consequences arising from component interactions after hardening

#### **IDE Analysis Activities**

##### **Review Documentation and Sample Code**

- Evaluate quality of security included in product documentation and example code
- Measure time to find, to read, and to understand documentation for implementing security features

##### **Exercise Features**

- Implement security test cases in the Integrated Development Environment
- Implement best practices (if possible) in the IDE
- Deployment testing within the IDE

##### **Evaluate Features**

- Compare and note implementation time, lines of code written, and other observations of platform and environment features that increase or decrease the programmer's level of effort during code development
- Evaluate compliance against best practices achieved on each platform for each security feature

### Summarize Engagement Findings

- Document findings from default attack surface evaluation
- Document findings from host hardening exercise
- Document findings from documentation review
- Document findings from security feature implementation and evaluation

## 2.3 Analytical Process

---

### Analytical Process

#### Assessment Methods

@stake employed these methods to assess the security of the .NET Framework and WebSphere platforms:

- Construction and execution of test cases
- Penetration testing techniques, including raw data from host hardening tasks
- Manual inspection of configuration and run-time settings
- Inspection of stack traces, where applicable

@stake used these methods to assess the relative security of each platform, not as a means of developing new vulnerabilities. For example, the analysis included a test case for filtering user input. To measure the level of effort required for a developer to successfully implement input filters, @stake first built several sample filters, then validated their effectiveness. During the validation exercise we used a subset of the techniques a potential attacker might use, but only a subset. Full penetration tests of the platforms were not part of @stake's scope of work.

#### Evaluation Criteria

During the analysis phase, @stake evaluated each Area of Analysis topic according to five criteria:

- **Best practice compliance.** For a given analysis topic, to what degree did the platform permit implementation of best practices? Factors influencing best practice compliance include transparent integration (the default behavior enforces best practices automatically), user-assist features in the IDE, and degree of clarity of configuration
- **Implementation complexity.** How difficult was it for the developer to implement the desired feature? Factors influencing implementation complexity include the ease of use of the feature (as implemented in a tool), amount and length of code (if any was needed)
- **Quality of documentation and sample code.** How appropriate was the documentation? If examples were supplied with the documentation, were they sufficient to illustrate the concept, and did they exhibit best practices?
- **Developer competence.** How skilled did the developer need to be in order to implement the security feature? To ensure an apples-to-apples comparison, prior knowledge of the tools platform (Visual Studio .NET, WebSphere Studio Application Developer) was not assumed.

- **Time to implement.** How long did it take to implement the desired security feature or behavior? The ratings take into effect two considerations:
  - *The search cost.* How long did it take for the user to find information on how to use the feature? Was the information shipped with the product or was it found externally on a vendor website or via Google?
  - *The implementation time.* How long did it take, after knowledge assimilation, to configure or code the platform to implement correct behaviors?

To ensure consistency in measurement, @stake defined standardized rating levels for each criterion. The tables below define the levels we used.

### Best Practice Compliance

Rating	Definition
1 - Not possible	Best practice cannot be implemented in the platform, or can be implemented but with prohibitive time constraints. "Prohibitive", in our subjective view, denotes an amount of time or labor sufficiently high that a reasonable person would not, under most circumstances, even entertain the notion of implementing the practice.
2 - Developer implements	Best practice can be implemented only by development of entirely new subsystems or classes.
3 - Developer extends	Platform permits partial implementation of best practice, but full implementation depends on extension of base classes or the use of custom exit routines or interfaces.
4 - Wizard	Best practice can be implemented in a straightforward manner using a checkbox, property sheet, wizard, or simple deployment descriptor.
5 - Transparent	Best practice compliance is built into the platform (and is "inescapable"), or is configurable but turned on by default.

Best practices, as defined by @stake, are based on generally accepted principles for secure application development and deployment. Sources for best practice information include:

- Security industry groups such as the Open Web Application Security Project (OWASP)
- Vendor platform documentation
- @stake's Centers of Excellence
- @stake's internal knowledge base

Where appropriate and necessary, we have cited specific sources for particular best practices.

### Implementation Complexity

Development Rating	Development Definition	Implementation Definition
1 - Large amount of code	Required 500 user-developed lines of code or more (spread for example over 10 classes of 50 lines each).	Required the installation of multiple external applications and configuration of multiple interfaces
2 - Medium amount of code	Required 51-500 user-developed lines of code.	Required the installation and configuration of external software
3 - Small amount of code	Required 0-50 user-developed lines of code.	Required the configuration of multiple interfaces.
4 - Wizard +	Implemented automatically, or via a wizard, but also requires construction of a single small class file (10 lines or less), class annotation or deployment descriptor entry, and/or small manual configuration file changes.	Implemented via a single configuration interface ( <i>i.e.</i> , wizard, GUI, configuration file)
5 - Wizard	Implemented automatically, or via a wizard, checkbox or property sheet.	Implemented automatically, the configuration is default.

### Quality of Documentation and Sample Code

Rating	Definition
1 - Incorrect or Insecure	Documentation provided gives information that might cause the user to build an insecure application. Examples provided are incorrect, cause errors, or implement insecure practices.
2 - Vague or Incomplete	Documentation provided is vague and does not indicate when a potential vulnerability could be introduced. Important information is withheld.
3 - Adequate	Suitable to task, but provides only summary-level information on best or worst practices. No references are provided.
4 - Suitable	Clearly describes how to implement the functionality; if less-than-best practice, describes security implications in detail. May provide a reference to an example provided elsewhere.
5 - Best Practice Documentation	Clearly describes how to implement the functionality using best practice examples. No potential vulnerabilities are introduced, and clear explanations (if appropriate) are provided regarding less secure alternatives.

Because both Microsoft and IBM claim to provide “total solutions” for developers, a key goal of the analysis was to determine how well each vendor lived up to those claims with respect to security. In particular, @stake feels strongly that any credible “solutions” should include accurate patterns and practices for developing real-world Internet-facing web and web service applications. Application patterns, practices and examples provided to customers, if and when implemented on public servers, should not put them at risk.

To that end, for the purposes of the analysis @stake limited its searches to documentation and related materials that could be found as part of the vendors’ software distributions or on the vendors’ websites. @stake recognizes that many readers will find this to be an unrealistic assumption. For example, many Java developers rely heavily on non-IBM resources such as Apache’s Jakarta Project and Sun’s java.sun.com website. However, to analyze how well each vendor provides a total security solution for developers, we did

not consider third-party sources in our evaluation of documentation and sample code quality. Caveat: in a limited number cases where documentation on a particular topic was not available from the vendor, we consulted external websites. This resulted in the vendor receiving the minimum score for the documentation criterion, but allowed the remaining criteria to be evaluated on their merits.

### Developer/Administrator Competence

Rating	Developer Definition	Administrator Definition
1 - Expert	5+ years of experience with web and multi-tier applications. Developer is intimately familiar with managed code implementations and concepts in environments such as J2EE.	5+ years of experience with the base operating system. Extensive experience with security principles and configuration; extensive experience with web system administration.
2 - Expert/intermediate		
3 - Intermediate	3-5 years of experience with web and multi-tier applications. Developer is competent with JSP, ASP, PHP, Perl, and related technologies, and has 2-3 years of J2EE and/or .NET Framework experience.	3-5 years of experience with the base operating system. Administrator has experience with web system administration.
4 - Intermediate/novice		
5 - Novice	0-1 years of experience. Developer has limited experience with web application development, and probably no experience with web services. Some previous knowledge of Java-like languages (including C) is assumed.	0-1 years of experience. Administrator has limited experience with web administration. Some previous knowledge of the platform is assumed.

### Time to Implement

Rating	Definition
1 - High	More than 4 hours
2 - Medium to High	1 to 4 hours
3 - Medium	16-60 minutes
4 - Low to Medium	6-15 minutes
5 - Low	5 minutes or less

Relative skill differences between developers make apples-to-apples comparison of implementation times difficult. Each analysis topic describes the kind and amount of effort required for implementation. In most cases, we deliberately exposed implementors to technologies that they were less familiar with, compared to other aspects of the platform. We did this to simulate what an implementor new to the platform might experience. In most cases we have documented the skill levels of the evaluators relative to each analysis topic.

## 2.4 Documentation of Findings

---

### Documentation of Findings

#### Documentation Formats

At the conclusion of the analysis phase, @stake created this document based on our findings. Key presentation formats we used to present information include:

- Best practice compliance analysis
- Level of effort analysis
- Scorecard
- Evaluation Matrix

Each format is explained below. In addition to these formats, we also include summary tables for each Area of Analysis.

#### Best Practice Compliance Analysis

Each Area of Analysis has an associated Best Practice Analysis table. Found in the Findings and Analysis section, these tables describe each Area of Analysis and indicate relative strengths and weaknesses. For each topic, @stake includes its understanding of best-in-class solutions, potential options available to developers or system administrators for each platform, the relative conformance of each option to the best practices, and recommendations for developers or administrators depending on the scenario. The bottom row of the table includes a calculated Best Practice Compliance score, which is the mean of the best practice scores for the Area of Analysis.

For example:

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=80]. The application server should provide flexible support for external identity stores such as LDAP servers. Proprietary identity stores (e.g., Active Directory) may also be used, provided the access protocols are well-documented and supported by the industry.	IIS 6 offers four options for authenticating against an external store: Basic authentication, Integrated Windows authentication (utilizing either NTLM or Kerberos), Digest authentication for Windows domain servers, and .NET Passport authentication.	<b>Transparent (5).</b> Integrated Windows authentication is enabled by default.	Basic authentication is not recommended even with the addition of SSL encryption since attacks such as cross-site tracing may still be able to recover a user's credentials.

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=81]. If an identity store is used for authentication, binding and lookup operations should be performed using a secure protocol.	Secure protocols for authenticating to an external store are NTLM, Kerberos, and .NET Passport.	<b>Transparent (5).</b> NTLM, Kerberos, and .NET Passport are all suitable choices.	

**Best Practice Compliance score**, based on mean of 2 best practices: 5

In this example, the Best Practice Compliance score of 5 equals  $(5 + 5) / 2$ .

## Level of Effort Analysis

Each Area of Analysis has an associated Level of Effort Analysis table. Found in the Findings and Analysis section, these tables describe the level of effort required by a developer or deployer to bring each platform to the level of best practices. The table estimates the level of effort by documenting the effort needed to fulfill several “test cases.” For each test case, four criteria are used to calculate the overall score for the level of effort: Implementation Complexity, Quality of Documentation and Sample Code, Developer Competence, and Time to Implement. The bottom row of the table includes a calculated Ease of Securing score, which is the mean of all of the test case scores for the Area of Analysis. The result is rounded to the nearest hundredth.

For example:

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=80.1]. Configure the application server to use an external identity store for user and group lookup.	Implementation Complexity	<b>Wizard (5).</b> Enabling any of the authentication mechanisms described above is easily accomplished via IIS' Authentication Methods configuration panel.
	Documentation and Examples	<b>Best practice (5).</b> The various options are clearly explained along with security recommendations.
	Implementor Competence	<b>Novice/intermediate (4).</b> The developer has extensive experience developing web applications and limited skills in configuring/administering IIS.
	Time to Implement	<b>Low (5).</b> The default configuration is secure and can quickly be changed if an alternative authentication method is preferred.
[id=81.1]. Configure secure protocols for identity lookup.	Implementation Complexity	<b>Wizard (5).</b> The default configuration is secure.
	Documentation and Examples	<b>Suitable (4).</b> Documentation is clear but does not give in-depth information on the security features and impacts of the various protocols.
	Implementor Competence	<b>Novice/intermediate (4).</b> The developer has extensive experience developing web applications and limited skills in configuring/administering IIS.
	Time to Implement	<b>Low (5).</b> The default configuration is secure and can quickly be changed if an alternative authentication method is preferred.

### **Level of Effort Analysis**

Test Case	Criteria	Rating
<b>Ease of Securing score</b> , based on mean of 2 test cases: 4.63		

In this example, the Ease of Securing score of 3.88 equals  $((3+5+4+3) + (3+5+4+4)) / (2 \text{ test case} * 4 \text{ metrics each})$ .

Each Area of Analysis section contains a *Notes* section that describes how we conducted our tests. Screenshots and code samples are included in most cases.

### **Scorecard**

Found at the beginning of the Findings and Analysis section, these tables summarize the findings from the *Best Practice Analysis* and *Level of Effort Analysis* tables. Three versions of this table are presented -- one each for a Web Application scenario, Web Service scenario, and Intranet Application scenario. The level of effort scores are weighted differently for each.

Each table is divided into three subsections: Application Server, Host and Operating System, and Web Server. These correspond to the three high-level analysis categories we used. Each row in the table summarizes the results for a particular analysis topic, grouped by area. The relative weight for the topic is shown, followed by the weighted scores for Best Practice Compliance and Ease of Securing for each platform. Scenario weights were assigned based upon criticality of the analysis topic to a specific scenario (i.e. Web Application, Web Service, or Intranet Application scenarios). These weights range from 1 (relatively unimportant) to 3 (extremely important). A scenario weight of zero means that the topic does not apply.

The icons in each row graphically depict the underlying raw score based on a scale of one to five:

- = raw score of 5
- = raw score of 4-4.99
- = raw score of 3-3.99
- = raw score of 2-2.99
- = raw score of 1-1.99

Subtotals are shown at the bottom of each high-level category, as well as an overall total score that is the sum of all three categories. The totals are shown “raw,” and also normalized to a five-point scale. The normalized scores were what we used to create the overall numbers shown in the graph in the Executive Summary.

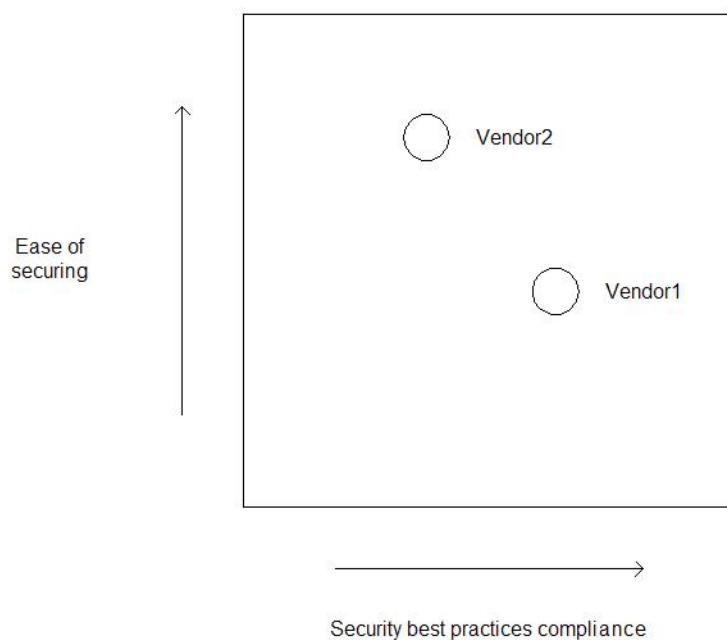
For example, here is a partial excerpt for the Web Application scenario, from the Application Server section. Because table space is at a premium, we have used shorthand to denote the different platforms (*e.g.*, “.NET Windows”):

## Web Application Scorecard

Area	Topic	Topic Weight	Weighted Scores					
			Best Practice Compliance			Ease of Securing		
			.NET Windows	WebSphere Linux	WebSphere Unix	.NET Windows	WebSphere Linux	WebSphere Unix
<b>Application Server</b>								
Application Logging Services	Exception Management <i>1 best practice, 2 test cases</i>	2	6/10	4/10	4/10	7.75/10	4.75/10	4.75/10
	Logging Privileges <i>1 best practice, 3 test cases</i>	2	4/10	8/10	8/10	5.67/10	8/10	8/10
	Log Management <i>1 best practice, 3 test cases</i>	2	8/10	6/10	6/10	8.67/10	6.83/10	6.83/10
Authentication and Access Control	Login Management <i>2 best practices, 2 test cases</i>	3	10.5/15	9/15	9/15	13.13/15	8.63/15	8.63/15
...								
Subtotal for 28 topics (range: 48-240 points)			173.5/240	173.25/240	171.25/240	183.81/240	162.76/240	162.26/240
<b>Subtotal (normalized to 5-point scale)</b>			<b>3.61/5</b>	<b>3.61/5</b>	<b>3.57/5</b>	<b>3.83/5</b>	<b>3.39/5</b>	<b>3.38/5</b>

## Evaluation Matrix

Found in the Executive Summary, this exhibit summarizes the Scorecard in a concise graphical format. Envisioned as a 2x2 matrix, the Evaluation Matrix graphs each platform's degree of security compliance against best practices (on the horizontal axis) and the relative level of effort required for compliance (vertical axis). The scores on each axis are normalized to a five-point scale, and are taken directly from the Scorecard. Three versions of the chart are included, one each for the Web Application, Web Service and Intranet scenarios.



## 3.1 Microsoft .NET Framework and Windows Server 2003

---

### Microsoft .NET Framework

#### Introduction

The test environment for Microsoft .NET consisted of two separate Dell desktop systems, described below. One system hosted the web and application server while the other hosted a Microsoft SQL Server 2000 database. The operating system for both was Microsoft Windows 2003. These test environments were used for an attack surface evaluation and all infrastructure-related testing. The installation and configuration procedures followed to setup the test environments are described below.

#### Hardware Specifications

P4/1.8Ghz Desktop

512MB RAM

15GB Hard Drive

ATI Rage 128 Video Card with 32MB VRAM

Dell E770p Monitor

#### Web and Application Server Setup

##### Windows 2003 Installation

Microsoft Windows 2003 base system was installed using a bootable CD on a 360-day license. In unpartitioned space, the installer formatted a partition with the NTFS file system and installed Windows on it. Next, we personalized the software and entered our product key information. After setting the networking options to allow interaction with our network, we rebooted the machine, completing the installation.

- Welcome to Setup Screen: Pressed “ENTER”
- Windows Licensing Agreement Screen: Pressed “F8” to agree
- Disk Partitioning Setup: Formatted unpartitioned space with NTFS file system for installation of windows. The system then reboots into the Graphical Installation wizard.
- Initial Setup Screens: Personalized computer, entered product key, date, time, username and password.

- Network Settings Screen: Selected the “Custom settings” radio button
- Network Components Screen
  - Selected “Internet Protocol (TCP/IP)”
  - Chose “Properties” to modify settings
  - Selected the “Use the following IP address” radio button
  - Entered “10.2.2.220” for the IP Address
  - Entered “255.255.0.0” for the Subnet Mask
  - Entered “10.2.1.1” for the Default gateway
  - Entered “10.2.2.211” for the Preferred DNS server
  - Chose “OK” to complete the modifications
- Completed installation and rebooted system
- Initial Logon: Logged in as Administrator
- Closed the “Manage Your Server” tool (No server roles are selected at this time)

## IIS 6.0 and ASP.NET Installation

IIS 6.0 and ASP were installed via the “Manage Your Server” tool included with Windows 2003 Server installation.

- Selected “Add or remove a role” to set up the system as an application server
- Selected “Application server (IIS, ASP.NET)” as the server role
- Selected the “FrontPage Server Extensions” check box
- Selected the “Enable ASP.NET” check box
- Selected “Finish” to complete the installation

## Database Setup

### Windows Installation

*Same as that used for the web and application servers’ desktop system. See above.*

### Microsoft SQL Server 2000 Installation

Microsoft SQL Server 2000 Enterprise Edition was installed via its installation CD. A warning from Windows 2003 that it was not compatible with Microsoft SQL Server 2000 SP2 or below was ignored since we planned to install service pack 3. We selected “Create a new instance of SQL Server, or install Client Tools”, accepted the license agreement, entered the CD key, and began the Typical installation of Server and Client Tools. The following settings were applied.

- Instance name: default

- Setup Type: Typical
- Use the same account for each service. Autostart SQL Server Service
- Accept default authentication mode of "Windows Authentication Mode"
- Accept default licensing mode of "Per Seat for"
- Entered "25" for number of devices.

Completed installation and proceeded to install the Microsoft SQL Server 2000 Enterprise Edition Service Pack 3 from Microsoft's website.

## 3.2 IBM WebSphere (Linux)

---

### IBM WebSphere J2EE on Linux

#### Introduction

The test environment for IBM WebSphere J2EE on Linux consisted of two separate Dell desktop systems, described below. One system hosted the web and application server while the other hosted a DB2 database. The operating system for both was RedHat Advanced Server 2.1. These test environments were used for an attack surface evaluation and all infrastructure-related testing. The installation and configuration procedures followed to setup the test environments are described below.

#### Hardware Specifications

P4/1.8Ghz Desktop

512MB RAM

15GB Hard Drive

ATI Rage 128 Video Card with 32MB VRAM

Dell E770p Monitor

#### Web and Application Server Setup

##### Linux Installation

RedHat Advanced Server 2.1 base system was installed using a bootable CD and the graphical mode installer. English, a standard US keyboard, and a three-button mouse were selected for the Advanced Server installation. The installer was allowed to automatically partition the disk and remove all partitions on the system. To create appropriate disk space for WebSphere, an /opt partition of 10GB was created and the /usr was resized to 3GB.

##### Boot Loader Configuration

- Used GRUB as the boot loader
- Installed boot loader on - /dev/hda Master Boot Record (MBR)
- Kernel parameters: left blank

- Did not force use of LBA32 (not normally required) - Not Selected
- Partition: /dev/hda6 Type: ext3
- Selected Default boot image - Selected
- Boot label: RedHat Linux Advanced Server
- Use a GRUB password - Not Selected

#### **Network Configuration - eth0**

- Configure using DHCP - Not Selected
- Activate on boot - Selected
- IP Address - 10.2.2.213
- Netmask - 255.255.0.0
- Network - 10.2.0.0
- Broadcast - 10.2.255.255
- Hostname - linux1
- Gateway - 10.2.1.1
- Primary DNS - 10.2.2.211

#### **Package Group Selection**

- GNOME - Selected
- KDE - Not Selected
- Software Development - Not Selected
- Select individual packages - Not Selected
- Total install size - 938 MB

#### **Other Settings**

- *Firewall configuration:* medium security level was selected along with the default firewall rules.
- *Accounts:* A root password was set and no other accounts were created.
- *Graphics:* Screen resolution of 800 X 600 with high color (16 bits)
- *Desktop:* GNOME Selected with graphical login type.
- *Kickstart:* The /root/anaconda-ks.cfg Kickstart configuration file that is generated from this installation was used to deploy additional Linux test machines to ensure consistency. When using this Kickstart file, the default run level was set to 3 (multi-user, no X). For consistency, this setting was set back to 5 (multi-user, X) prior to additional testing.

#### **WebSphere Installation**

Prior to installation, the instructions in the IBM document “IBM WebSphere Application Server, Version 5: Getting Started” specify the creation and modification of several users and groups, including:

- Add the user mqm.
- Add the group mqbrkrs.
- Grant root privileges to both mqm and mqbrkrs.

WebSphere was installed using the graphical mode installer and the installation CD. Both IBM WebSphere Application Server, version 5.0, and IBM HTTP Server, version 1.3.26, were installed on the /opt partition. After the installation wizard finished, we verified installation in the WebSphere First Steps GUI and then started both the application and web servers.

## **Database Setup**

### **Linux Installation**

*Same as that used for the web and application servers' desktop system. See above.*

### **DB2 Installation**

IBM DB2 Universal Database 8.1 Workgroup Server Edition was installed on a RedHat Advanced Server 2.1 Base System. Prior to installation, we followed instructions from the IBM document “Up and Running with DB2 for Linux” to uninstall the existing Java package from the RedHat base build and to install the pdksh package from Disk 2 of the RedHat Advanced Server 2.1 distribution. Once this step was completed, we proceeded to install DB2 with the graphical mode installer from the installation CD. After we selected the DB2 UDB Workgroup Server Edition from the launchpad dialog, the DB2 Setup Wizard began.

### **Wizard Settings**

- Installation type: Selected typical.
- New user: Selected and new user *dasusr1* created.
- Create a DB2 instance: Selected.
- New user for DB2 instance: Selected and user *db2inst1* created.
- New fenced user: Selected and user *db2grp1* created.
- DB2 Tools Catalog: Selected Do not prepare catalog.
- Health Monitor Notification: Entered *db2inst1* as the new contact.

After copying files and reviewing the installation options, the wizard finished the DB2 installation.

### 3.3 IBM WebSphere (Unix)

---

## IBM WebSphere J2EE on Unix

### Introduction

The test environment for IBM WebSphere J2EE on Unix consisted of two separate systems from a leading provider of Unix servers, described below. One system hosted the web and application server while the other hosted a DB2 database. The operating system for both was the leading commercial distribution of Unix. These test environments were used for an attack surface evaluation and all infrastructure-related testing. The installation and configuration procedures followed to setup the test environments are described below.

### Hardware Specifications

Single 440 MHz processor

512 MB RAM

9GB Hard Drive

### Web and Application Server Setup

#### Unix Installation

The Unix base operating system was installed using a bootable CD and a serial console connection. The disk (/dev/dsk/c0t0d0) was re-partitioned, and the installation software was installed on c0t0d0. A swap slice of 512MB was placed at the beginning of the disk and the system rebooted to continue the installation process. The Web Start Command Line Installation Screen greeted us. We set time zone, root password, no power management options, and the following network configuration:

#### Network Configuration

- Please choose a primary network interface - 1 (hme0)
- Would you like to configure using DHCP? - No
- Host name - sol1
- IP address - 10.2.2.216
- Subnet netmask - 255.255.0.0

- Enable Ipv6 - No
- Default router detection method - Specify one (Detect one is the default)
- Router IP address - 10.2.1.1
- Enable Kerberos - No
- Name service - 5 (None)

Next, we swapped out Installation CD 1 for Installation CD 2 and continued the installation. Finally, we swapped CD2 for the Software Supplement CD and completed the installation.

### WebSphere Installation

To use the graphical mode installer, the SSH daemon on the server was configured to allow root SSH connections and X11 forwarding. We then used a Linux workstation to create an X session on the server to install the WebSphere Application Server and IBM HTTP Web Server.

Prior to installation, instructions in the IBM document “IBM WebSphere Application Server, Version 5: Getting Started” specify the creation and modification of several users and groups. To facilitate the creation of home directories, the automounter map was modified

- The following line was commented out of /etc/auto\_master:
  - /home auto\_home -nobrowse
- The automounter daemon was stopped and restarted with the commands:
  - /etc/init.d/autofs stop
  - /etc/init.d/autofs start
- The mqm and mqbrkrs groups were added
- The root user was added to the mqm and mqbrkrs groups.
- The root user was then logged out and back in to effect group membership changes.

**Kernel Configuration** The IBM document “IBM WebSphere Application Server, Version 5: Getting Started,” stipulates that several kernel parameters be modified prior to installation. The following kernel modifications were made to the /etc/system file, and the system rebooted.

- shmsys:shminfo\_shmmax - 4294967295
- shmsys:shminfo\_shmseg - 1024
- shmsys:shminfo\_shmmni - 1024
- semsys:seminfo\_semaem - 16384
- semsys:seminfo\_semmni - 1024
- semsys:seminfo\_semmap - 1026
- semsys:seminfo\_semmns - 16384
- semsys:seminfo\_semmsl - 100

- semsys:seminfo\_semopm - 100
- semsys:seminfo\_semmnu - 2048
- semsys:seminfo\_semume - 256
- msgsys:msginfo\_msgmap - 1026
- msgsys:msginfo\_msgmax - 4096
- rlim\_fd\_cur - 1024

WebSphere was installed using the Launch Pad utility on the installation CD. Both IBM WebSphere Application Server, version 5.0, and IBM HTTP Server, version 1.3.26, were installed in the /opt directory. After the installation wizard finished, we verified installation in the WebSphere First Steps GUI and then started both the application and web servers.

## **Database Setup**

### **Unix Installation**

*Same as that used for the web and application servers' system. See above.*

### **DB2 Installation**

IBM DB2 Universal Database 8.1 Workgroup Server Edition was installed on the Unix base system. Prior to installation, we followed instructions from the IBM document “Quick Beginnings for DB2 Servers, Version 8” to decompress and untar the installation software image in /export/home directory. Next, several kernel parameters were adjusted to accommodate DB2. For systems with 512MB of RAM, the guide suggested using the settings file kernel.param.512MB, provided on the installation media. However, this particular file was not supplied with the media. Thus, the settings for systems with 512MB and more memory from the “Quick Beginnings: Updating kernel configuration parameters” page on IBM’s website were used instead.

### **Kernel Parameter Adjustments for DB2 Installation**

- msgsys:msginfo\_msgmax - 65535(1)
- msgsys:msginfo\_msgrnb - 65535(1)
- msgsys:msginfo\_msgmap - 258
- msgsys:msginfo\_msgrnbi - 256
- msgsys:msginfo\_msgrsz - 16
- msgsys:msginfo\_msgrql - 1024
- msgsys:msginfo\_msgrseg - 32767(2)
- shmsys:shminfo\_shmmax - 536870912(3)
- shmsys:shminfo\_shmseg - 50
- shmsys:shminfo\_shmmni - 300

- semsys:seminfo\_semmni - 1024
- semsys:seminfo\_semmmap - 1026
- semsys:seminfo\_semmns - 2048
- semsys:seminfo\_semmnu - 2048
- semsys:seminfo\_semume - 50

### User and Group Additions

The instructions in the IBM document “IBM DB2 Universal Database: Installation and Configuration Supplement, Version 8” specify several user and group modifications and additions. To facilitate creation of user home directories, the automounter map was modified.

- This line was commented out of /etc/auto\_master: “/home auto\_home -nobrowse”
- The automounter daemon was stopped and restarted.
- The db2iadm1 (the instance owner group) group was added.
- The db2fadm1 (the fenced user group ) group was added.
- The db2asgrp (the DB2 administration server group) group was added.
- The db2inst1 user (the instance owner) was added.
- The db2fenc1 user (the DB2 fenced user) was added.
- The dasusr1 user (the DB2 administration server user) was added.

Once this step was completed, we proceeded to install DB2.WSE (DB2 Workgroup Server Edition) with the d2\_install tool. Since the db2\_install installation script does not completely set up a working DB2 system, several post-installation steps were performed to create a working DB2 system. These steps were found in the IBM document “IBM DB2 Universal Database: Installation and Configuration Supplement, Version 8” and included:

- Creating a DB2 Administration Server (DAS)
- Creating a DB2 instance
- Updating the DB2 license key
- Setting up TCP/IP communications for DB2

## 4.1 Scorecard

---

### Introduction

This section contains the weighted and summarized scores for each of the scenarios @stake evaluated:

- Web Application
- Web Service
- Intranet

Table links are “clickable” and permit drill-down for specific analysis categories, areas and topics.

The icons in each row graphically depict the underlying raw score based on a scale of one to five:

- = raw score of 5
- = raw score of 4-4.99
- = raw score of 3-3.99
- = raw score of 2-2.99
- = raw score of 1-1.99

Subtotals are shown at the bottom of each high-level category, as well as an overall total score that is the sum of all three categories. The totals are shown “raw,” and also normalized to a five-point scale. The normalized scores were what we used to create the overall numbers shown in the graph in the Executive Summary.

For more details on the scoring methodology, see the Documentation of Findings section.

### Scorecard: Web Application

## Web Application Scorecard

Area	Topic	Topic Weight	Weighted Scores					
			Best Practice Compliance			Ease of Securing		
			.NET Windows	WebSphere Linux	WebSphere Unix	.NET Windows	WebSphere Linux	WebSphere Unix
<b>Application Server</b>								
Application Logging Services	Exception Management <i>1 best practice, 2 test cases</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.75/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4.75/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4.75/10
	Logging Privileges <i>1 best practice, 3 test cases</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 5.67/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10
	Log Management <i>1 best practice, 3 test cases</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8.67/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6.83/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6.83/10
Authentication and Access Control	Login Management <i>2 best practices, 2 test cases</i>	3	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 10.5/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 9/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 9/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 13.13/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8.63/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8.63/15
	Role Based Access Control <i>2 best practices, 6 test cases</i>	3	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 12/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 13.5/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 13.5/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 11.75/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 13.25/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 13.25/15
	Web Server Integration <i>1 best practice, 1 test case</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 10/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 10/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 5.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 5.5/10
Communications	Communication Security <i>1 best practice, 2 test cases</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.25/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.25/10
	Network-Accessible Services <i>1 best practice, 1 test case</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10
Cryptography	Cryptographic Hashing <i>1 best practice, 2 test cases</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.5/10
	Encryption Algorithms <i>1 best practice, 3 test cases</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7/10
	Key Generation <i>1 best practice, 1 test case</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 10/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.5/10
	Random Number Generation <i>1 best practice, 1 test case</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7/10
	Secrets Storage <i>1 best practice, 1 test case</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7/10
	XML Cryptography <i>1 best practice, 2 test cases</i>	0	n/a	n/a	n/a	n/a	n/a	n/a
Database Access	Database Pool Connection Encryption <i>1 best practice, 1 test case</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4/10
	Data Query Safety <i>1 best practice, 2 test cases</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 10/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 10/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8.25/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8.25/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8.25/10

## Web Application Scorecard

Area	Topic	Topic Weight	Weighted Scores					
			Best Practice Compliance			Ease of Securing		
			.NET Windows	WebSphere Linux	WebSphere Unix	.NET Windows	WebSphere Linux	WebSphere Unix
Data Validation	Common Validators 1 best practice, 1 test case	1	3/5	5/5	5/5	3.25/5	3.5/5	3.5/5
	Data Sanitization 1 best practice, 1 test case	2	6/10	8/10	8/10	8/10	6.5/10	6.5/10
	Negative Data Validation 1 best practice, 1 test case	2	6/10	8/10	8/10	5.5/10	7.5/10	7.5/10
	Output Filtering 1 best practice, 1 test case	1	4/5	5/5	5/5	3.75/5	4/5	4/5
	Positive Data Validation 1 best practice, 1 test case	2	8/10	6/10	6/10	9/10	5/10	5/10
	Type Checking 1 best practice, 1 test case	1	5/5	4/5	4/5	4.25/5	3.25/5	3.25/5
Information Disclosure	Error Handling 1 best practice, 1 test case	2	10/10	8/10	8/10	8.5/10	7.5/10	7.5/10
	Stack Traces and Debugging 1 best practice, 1 test case	2	10/10	10/10	10/10	8.5/10	8/10	8/10
Runtime Container Security	Code Security 4 best practices, 5 test cases	1	3.75/5	2.75/5	2.75/5	3.95/5	2.55/5	2.55/5
	Runtime Account Privileges 1 best practice, 1 test case	2	10/10	8/10	8/10	9/10	6/10	6/10
Web Services	Credentials Mapping 1 best practice, 1 test case	0	n/a	n/a	n/a	n/a	n/a	n/a
	SOAP Router Data Validation 2 best practices, 2 test cases	0	n/a	n/a	n/a	n/a	n/a	n/a
Subtotal for 28 topics (range: 48-240 points)			174.25/240	173.25/240	171.25/240	184.41/240	162.76/240	162.26/240
Subtotal (normalized to 5-point scale)			3.63/5	3.61/5	3.57/5	3.84/5	3.39/5	3.38/5
<b>Host and Operating System</b>								
IP Stack Hardening	Protocol Settings 2 best practices, 4 test cases	3	12/15	12/15	12/15	11.63/15	7.69/15	9.19/15
Service Minimization	Installed Packages 1 best practice, 3 test cases	2	10/10	8/10	8/10	8/10	5.5/10	5.67/10
Network Services	1 best practice, 2 test cases	3	12/15	12/15	9/15	10.88/15	10.5/15	7.88/15
Subtotal for 3 topics (range: 8-40 points)			34/40	32/40	29/40	30.5/40	23.69/40	22.73/40
Subtotal (normalized to 5-point scale)			4.25/5	4/5	3.63/5	3.81/5	2.96/5	2.84/5

## Web Application Scorecard

Area	Topic	Topic Weight	Weighted Scores					
			Best Practice Compliance			Ease of Securing		
			.NET Windows	WebSphere Linux	WebSphere Unix	.NET Windows	WebSphere Linux	WebSphere Unix
<b>Web Server</b>								
Architecture	Security Partitioning 3 best practices, 4 test cases	2	10/10	10/10	10/10	8.75/10	8.38/10	8.38/10
Authentication	Authentication Input Validation 1 best practice, 2 test cases	1	3/5	4/5	4/5	4.13/5	3.75/5	3.75/5
	Authentication Methods 1 best practice, 5 test cases	2	8/10	6/10	6/10	8.3/10	5.7/10	5.7/10
	Credential Handling 3 best practices, 3 test cases	2	7.33/10	7.33/10	7.33/10	8.17/10	7.33/10	7.33/10
	Digital Certificates 3 best practices, 5 test cases	2	10/10	6/10	6/10	9.3/10	7.5/10	7.5/10
	External Authentication 2 best practices, 2 test cases	3	15/15	12/15	12/15	13.88/15	11.63/15	11.63/15
Communication Security	Platform Integrated Authentication 1 best practice, 2 test cases	1	5/5	4/5	3/5	5/5	3.75/5	3.63/5
	Session Encryption 3 best practices, 7 test cases	3	11/15	10/15	13/15	11.68/15	11.46/15	12/15
Information Disclosure	Error Messages and Exception Handling 1 best practice, 1 test case	2	10/10	8/10	8/10	9/10	7.5/10	7.5/10
	Logging 4 best practices, 4 test cases	2	8.5/10	8/10	8/10	8.88/10	7.75/10	7.75/10
	URL Content Protection 3 best practices, 5 test cases	2	7.33/10	7.33/10	7.33/10	9.2/10	7.3/10	7.3/10
Session Management	Cookie Handling 2 best practices, 2 test cases	2	7/10	9/10	9/10	7/10	9/10	9/10
	Session Identifier 1 best practice, 1 test case	3	15/15	15/15	15/15	12.75/15	14.25/15	14.25/15
	Session Lifetime 1 best practice, 1 test case	3	15/15	15/15	15/15	14.25/15	11.25/15	11.25/15
Subtotal for 14 topics (range: 30-150 points)			132.17/150	121.67/150	123.67/150	130.27/150	116.55/150	116.96/150
Subtotal (normalized to 5-point scale)			4.41/5	4.06/5	4.12/5	4.34/5	3.88/5	3.9/5
<b>Web Application Totals</b>								
Total for 45 topics (range: 86-430 points)			340.42/430	326.92/430	323.92/430	345.18/430	302.99/430	301.95/430
Total (normalized to 5-point scale)			3.96/5	3.8/5	3.77/5	4.01/5	3.52/5	3.51/5

## Scorecard: Web Service

## Web Service Scorecard

Area	Topic	Topic Weight	Weighted Scores					
			Best Practice Compliance			Ease of Securing		
			.NET Windows	WebSphere Linux	WebSphere Unix	.NET Windows	WebSphere Linux	WebSphere Unix
<b>Application Server</b>								
Application Logging Services	Exception Management <i>1 best practice, 2 test cases</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.75/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4.75/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4.75/10
	Logging Privileges <i>1 best practice, 3 test cases</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 5.67/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10
	Log Management <i>1 best practice, 3 test cases</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8.67/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6.83/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6.83/10
Authentication and Access Control	Login Management <i>2 best practices, 2 test cases</i>	3	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 10.5/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 9/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 9/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 13.13/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8.63/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8.63/15
	Role Based Access Control <i>2 best practices, 6 test cases</i>	3	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 12/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 13.5/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 13.5/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 11.75/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 13.25/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 13.25/15
	Web Server Integration <i>1 best practice, 1 test case</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 10/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 10/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 5.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 5.5/10
Communications	Communication Security <i>1 best practice, 2 test cases</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.25/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.25/10
	Network-Accessible Services <i>1 best practice, 1 test case</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10
Cryptography	Cryptographic Hashing <i>1 best practice, 2 test cases</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.5/10
	Encryption Algorithms <i>1 best practice, 3 test cases</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7/10
	Key Generation <i>1 best practice, 1 test case</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 10/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.5/10
	Random Number Generation <i>1 best practice, 1 test case</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7/10
	Secrets Storage <i>1 best practice, 1 test case</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7/10
	XML Cryptography <i>1 best practice, 2 test cases</i>	3	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 9/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 9/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 9/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 10.5/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.5/15	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 7.5/15
Database Access	Database Pool Connection Encryption <i>1 best practice, 1 test case</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6.5/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 4/10
	Data Query Safety <i>1 best practice, 2 test cases</i>	2	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 6/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 10/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 10/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8.25/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8.25/10	<span style="background-color: #0070C0; border: 1px solid black; color: white; padding: 2px;"> </span> 8.25/10

## Web Service Scorecard

Area	Topic	Topic Weight	Weighted Scores					
			Best Practice Compliance			Ease of Securing		
			.NET Windows	WebSphere Linux	WebSphere Unix	.NET Windows	WebSphere Linux	WebSphere Unix
Data Validation	Common Validators 1 best practice, 1 test case	1	3/5	5/5	5/5	3.25/5	3.5/5	3.5/5
	Data Sanitization 1 best practice, 1 test case	2	6/10	8/10	8/10	8/10	6.5/10	6.5/10
	Negative Data Validation 1 best practice, 1 test case	3	9/15	12/15	12/15	8.25/15	11.25/15	11.25/15
	Output Filtering 1 best practice, 1 test case	1	4/5	5/5	5/5	3.75/5	4/5	4/5
	Positive Data Validation 1 best practice, 1 test case	3	12/15	9/15	9/15	13.5/15	7.5/15	7.5/15
	Type Checking 1 best practice, 1 test case	1	5/5	4/5	4/5	4.25/5	3.25/5	3.25/5
Information Disclosure	Error Handling 1 best practice, 1 test case	2	10/10	8/10	8/10	8.5/10	7.5/10	7.5/10
	Stack Traces and Debugging 1 best practice, 1 test case	2	10/10	10/10	10/10	8.5/10	8/10	8/10
Runtime Container Security	Code Security 4 best practices, 5 test cases	1	3.75/5	2.75/5	2.75/5	3.95/5	2.55/5	2.55/5
	Runtime Account Privileges 1 best practice, 1 test case	2	10/10	8/10	8/10	9/10	6/10	6/10
Web Services	Credentials Mapping 1 best practice, 1 test case	3	9/15	9/15	9/15	12/15	5.25/15	5.25/15
	SOAP Router Data Validation 2 best practices, 2 test cases	3	12/15	12/15	12/15	12.75/15	12.38/15	12.38/15
	Subtotal for 28 topics (range: 59-295 points) Subtotal (normalized to 5-point scale)		211.25/295 3.58/5	210.25/295 3.56/5	208.25/295 3.53/5	226.91/295 3.85/5	194.13/295 3.29/5	193.63/295 3.28/5
<b>Host and Operating System</b>								
IP Stack Hardening	Protocol Settings 2 best practices, 4 test cases	3	12/15	12/15	12/15	11.63/15	7.69/15	9.19/15
Service Minimization	Installed Packages 1 best practice, 3 test cases	2	10/10	8/10	8/10	8/10	5.5/10	5.67/10
	Network Services 1 best practice, 2 test cases	3	12/15	12/15	9/15	10.88/15	10.5/15	7.88/15
Subtotal for 3 topics (range: 8-40 points) Subtotal (normalized to 5-point scale)			34/40 4.25/5	32/40 4/5	29/40 3.63/5	30.5/40 3.81/5	23.69/40 2.96/5	22.73/40 2.84/5

## Web Service Scorecard

Area	Topic	Topic Weight	Weighted Scores					
			Best Practice Compliance			Ease of Securing		
			.NET Windows	WebSphere Linux	WebSphere Unix	.NET Windows	WebSphere Linux	WebSphere Unix
<b>Web Server</b>								
Architecture	Security Partitioning 3 best practices, 4 test cases	2	10/10	10/10	10/10	8.75/10	8.38/10	8.38/10
Authentication	Authentication Input Validation 1 best practice, 2 test cases	1	3/5	4/5	4/5	4.13/5	3.75/5	3.75/5
	Authentication Methods 1 best practice, 5 test cases	2	8/10	6/10	6/10	8.3/10	5.7/10	5.7/10
	Credential Handling 3 best practices, 3 test cases	2	7.33/10	7.33/10	7.33/10	8.17/10	7.33/10	7.33/10
	Digital Certificates 3 best practices, 5 test cases	3	15/15	9/15	9/15	13.95/15	11.25/15	11.25/15
	External Authentication 2 best practices, 2 test cases	3	15/15	12/15	12/15	13.88/15	11.63/15	11.63/15
Communication Security	Platform Integrated Authentication 1 best practice, 2 test cases	1	5/5	4/5	3/5	5/5	3.75/5	3.63/5
	Session Encryption 3 best practices, 7 test cases	3	11/15	10/15	13/15	11.68/15	11.46/15	12/15
	Error Messages and Exception Handling 1 best practice, 1 test case	2	10/10	8/10	8/10	9/10	7.5/10	7.5/10
Information Disclosure	Logging 4 best practices, 4 test cases	2	8.5/10	8/10	8/10	8.88/10	7.75/10	7.75/10
	URL Content Protection 3 best practices, 5 test cases	2	7.33/10	7.33/10	7.33/10	9.2/10	7.3/10	7.3/10
	Cookie Handling 2 best practices, 2 test cases	2	7/10	9/10	9/10	7/10	9/10	9/10
Session Management	Session Identifier 1 best practice, 1 test case	3	15/15	15/15	15/15	12.75/15	14.25/15	14.25/15
	Session Lifetime 1 best practice, 1 test case	3	15/15	15/15	15/15	14.25/15	11.25/15	11.25/15
Subtotal for 14 topics (range: 31-155 points)			137.17/155	124.67/155	126.67/155	134.92/155	120.3/155	120.71/155
Subtotal (normalized to 5-point scale)			4.42/5	4.02/5	4.09/5	4.35/5	3.88/5	3.89/5
<b>Web Service Totals</b>								
Total for 45 topics (range: 98-490 points)			382.42/490	366.92/490	363.92/490	392.33/490	338.12/490	337.07/490
Total (normalized to 5-point scale)			3.9/5	3.74/5	3.71/5	4/5	3.45/5	3.44/5

## Scorecard: Intranet

**Intranet Scorecard**

Area	Topic	Topic Weight	Weighted Scores					
			Best Practice Compliance			Ease of Securing		
			.NET Windows	WebSphere Linux	WebSphere Unix	.NET Windows	WebSphere Linux	WebSphere Unix
<b>Application Server</b>								
Application Logging Services	Exception Management <i>1 best practice, 2 test cases</i>	2	6/10	4/10	4/10	7.75/10	4.75/10	4.75/10
	Logging Privileges <i>1 best practice, 3 test cases</i>	1	2/5	4/5	4/5	2.83/5	4/5	4/5
	Log Management <i>1 best practice, 3 test cases</i>	2	8/10	6/10	6/10	8.67/10	6.83/10	6.83/10
Authentication and Access Control	Login Management <i>2 best practices, 2 test cases</i>	2	7/10	6/10	6/10	8.75/10	5.75/10	5.75/10
	Role Based Access Control <i>2 best practices, 6 test cases</i>	2	8/10	9/10	9/10	7.83/10	8.83/10	8.83/10
	Web Server Integration <i>1 best practice, 1 test case</i>	2	10/10	8/10	8/10	10/10	5.5/10	5.5/10
Communications	Communication Security <i>1 best practice, 2 test cases</i>	2	6/10	8/10	8/10	7.5/10	7.25/10	7.25/10
	Network-Accessible Services <i>1 best practice, 1 test case</i>	1	4/5	4/5	3/5	3.75/5	3.25/5	3/5
Cryptography	Cryptographic Hashing <i>1 best practice, 2 test cases</i>	2	6/10	6/10	6/10	7.5/10	7.5/10	7.5/10
	Encryption Algorithms <i>1 best practice, 3 test cases</i>	2	6/10	6/10	6/10	7/10	7/10	7/10
	Key Generation <i>1 best practice, 1 test case</i>	2	10/10	6/10	6/10	8/10	7.5/10	7.5/10
	Random Number Generation <i>1 best practice, 1 test case</i>	2	6/10	6/10	6/10	7/10	7/10	7/10
	Secrets Storage <i>1 best practice, 1 test case</i>	2	4/10	6/10	6/10	4.5/10	7/10	7/10
	XML Cryptography <i>1 best practice, 2 test cases</i>	0	n/a	n/a	n/a	n/a	n/a	n/a
Database Access	Database Pool Connection Encryption <i>1 best practice, 1 test case</i>	1	3/5	2/5	2/5	3.25/5	2/5	2/5
	Data Query Safety <i>1 best practice, 2 test cases</i>	2	6/10	10/10	10/10	8.25/10	8.25/10	8.25/10

## Intranet Scorecard

Area	Topic	Topic Weight	Weighted Scores					
			Best Practice Compliance			Ease of Securing		
			.NET Windows	WebSphere Linux	WebSphere Unix	.NET Windows	WebSphere Linux	WebSphere Unix
Data Validation	Common Validators 1 best practice, 1 test case	1	■ 3/5	■ 5/5	■ 5/5	■ 3.25/5	■ 3.5/5	■ 3.5/5
	Data Sanitization 1 best practice, 1 test case	2	■ 6/10	■ 8/10	■ 8/10	■ 8/10	■ 6.5/10	■ 6.5/10
	Negative Data Validation 1 best practice, 1 test case	2	■ 6/10	■ 8/10	■ 8/10	■ 5.5/10	■ 7.5/10	■ 7.5/10
	Output Filtering 1 best practice, 1 test case	1	■ 4/5	■ 5/5	■ 5/5	■ 3.75/5	■ 4/5	■ 4/5
	Positive Data Validation 1 best practice, 1 test case	2	■ 8/10	■ 6/10	■ 6/10	■ 9/10	■ 5/10	■ 5/10
	Type Checking 1 best practice, 1 test case	1	■ 5/5	■ 4/5	■ 4/5	■ 4.25/5	■ 3.25/5	■ 3.25/5
Information Disclosure	Error Handling 1 best practice, 1 test case	2	■ 10/10	■ 8/10	■ 8/10	■ 8.5/10	■ 7.5/10	■ 7.5/10
	Stack Traces and Debugging 1 best practice, 1 test case	2	■ 10/10	■ 10/10	■ 10/10	■ 8.5/10	■ 8/10	■ 8/10
Runtime Container Security	Code Security 4 best practices, 5 test cases	1	■ 3.75/5	■ 2.75/5	■ 2.75/5	■ 3.95/5	■ 2.55/5	■ 2.55/5
	Runtime Account Privileges 1 best practice, 1 test case	2	■ 10/10	■ 8/10	■ 8/10	■ 9/10	■ 6/10	■ 6/10
Web Services	Credentials Mapping 1 best practice, 1 test case	0	n/a	n/a	n/a	n/a	n/a	n/a
	SOAP Router Data Validation 2 best practices, 2 test cases	0	n/a	n/a	n/a	n/a	n/a	n/a
Subtotal for 28 topics (range: 43-215 points)			157.75/215	155.75/215	154.75/215	166.28/215	146.22/215	145.97/215
Subtotal (normalized to 5-point scale)			3.67/5	3.62/5	3.6/5	3.87/5	3.4/5	3.39/5
<b>Host and Operating System</b>								
IP Stack Hardening	Protocol Settings 2 best practices, 4 test cases	2	■ 8/10	■ 8/10	■ 8/10	■ 7.75/10	■ 5.13/10	■ 6.13/10
	Installed Packages 1 best practice, 3 test cases	2	■ 10/10	■ 8/10	■ 8/10	■ 8/10	■ 5.5/10	■ 5.67/10
Service Minimization	Network Services 1 best practice, 2 test cases	2	■ 8/10	■ 8/10	■ 6/10	■ 7.25/10	■ 7/10	■ 5.25/10
	Subtotal for 3 topics (range: 6-30 points)		26/30	24/30	22/30	23/30	17.63/30	17.04/30
Subtotal (normalized to 5-point scale)			4.33/5	4/5	3.67/5	3.83/5	2.94/5	2.84/5

**Intranet Scorecard**

Area	Topic	Topic Weight	Weighted Scores					
			Best Practice Compliance			Ease of Securing		
			.NET Windows	WebSphere Linux	WebSphere Unix	.NET Windows	WebSphere Linux	WebSphere Unix
<b>Web Server</b>								
Architecture	Security Partitioning <i>3 best practices, 4 test cases</i>	2	10/10	10/10	10/10	8.75/10	8.38/10	8.38/10
Authentication	Authentication Input Validation <i>1 best practice, 2 test cases</i>	1	3/5	4/5	4/5	4.13/5	3.75/5	3.75/5
	Authentication Methods <i>1 best practice, 5 test cases</i>	2	8/10	6/10	6/10	8.3/10	5.7/10	5.7/10
	Credential Handling <i>3 best practices, 3 test cases</i>	2	7.33/10	7.33/10	7.33/10	8.17/10	7.33/10	7.33/10
	Digital Certificates <i>3 best practices, 5 test cases</i>	1	5/5	3/5	3/5	4.65/5	3.75/5	3.75/5
	External Authentication <i>2 best practices, 2 test cases</i>	2	10/10	8/10	8/10	9.25/10	7.75/10	7.75/10
Communication Security	Platform Integrated Authentication <i>1 best practice, 2 test cases</i>	2	10/10	8/10	6/10	10/10	7.5/10	7.25/10
	Session Encryption <i>3 best practices, 7 test cases</i>	2	7.33/10	6.67/10	8.67/10	7.79/10	7.64/10	8/10
	Error Messages and Exception Handling <i>1 best practice, 1 test case</i>	1	5/5	4/5	4/5	4.5/5	3.75/5	3.75/5
Information Disclosure	Logging <i>4 best practices, 4 test cases</i>	2	8.5/10	8/10	8/10	8.88/10	7.75/10	7.75/10
	URL Content Protection <i>3 best practices, 5 test cases</i>	1	3.67/5	3.67/5	3.67/5	4.6/5	3.65/5	3.65/5
	Cookie Handling <i>2 best practices, 2 test cases</i>	2	7/10	9/10	9/10	7/10	9/10	9/10
Session Management	Session Identifier <i>1 best practice, 1 test case</i>	3	15/15	15/15	15/15	12.75/15	14.25/15	14.25/15
	Session Lifetime <i>1 best practice, 1 test case</i>	1	5/5	5/5	5/5	4.75/5	3.75/5	3.75/5
Subtotal for 14 topics (range: 24-120 points)			104.83/120	97.67/120	97.67/120	103.5/120	93.95/120	94.06/120
<b>Subtotal (normalized to 5-point scale)</b>			<b>4.37/5</b>	<b>4.07/5</b>	<b>4.07/5</b>	<b>4.31/5</b>	<b>3.91/5</b>	<b>3.92/5</b>
<b>Intranet Totals</b>								
Total for 45 topics (range: 73-365 points)			288.58/365	277.42/365	274.42/365	292.79/365	257.79/365	257.07/365
<b>Total (normalized to 5-point scale)</b>			<b>3.95/5</b>	<b>3.8/5</b>	<b>3.76/5</b>	<b>4.01/5</b>	<b>3.53/5</b>	<b>3.52/5</b>

## 4.2 Application Server

---

### Application Server

The application server fulfills dynamic requests made by the web server on behalf of the user. The application server is typically placed in protected zone behind a firewall, and contains these components:

- **Dynamic server pages.** Provides interfaces to dynamic runtime objects using a combination of page markup and script code. *Examples: ASP.NET, Java Server Pages (JSP)*
- **Runtime objects.** Implement the application's business logic, based on actions triggered by the front-end (web server) or back-end (database server). *Examples: .NET Framework assemblies, Java servlets, Enterprise Java Beans, plain old Java objects (POJOs)*
- **Managed code runtime container.** Provides a sandboxed execution environment for in-memory runtime objects, and mediates between the objects and the underlying operating system. The runtime container enforces runtime privileges to prevent code from obtaining unauthorized privileges. *Examples: Microsoft common language runtime (CLR), IBM WebSphere Application Server (Java Virtual Machine plus Servlet and EJB containers)*

Specific Application Server areas of analysis include:

- **Application Logging Services** . Application Logging Services provide an application with a simple to use, yet flexible and robust mechanism for logging output to multiple destinations. Property driven parameters allow an application to easily modify the semantics of logging without changing any of the application's code. Application Logging Services allow an application to audit application events, errors, unexpected events, session metrics, and any other information pertinent to an administrator and/or application user in order to help them use and maintain an application.
- **Authentication and Access Control** . Authentication enables content control by only allowing access to confirmed, designated users. Good Authentication also facilitates accountability of actions. These functions are most important for external applications. Web application and web services are weighted as high and intranet as medium.
- **Communications** . Architecture allows for the proper function of an application without opening additional avenues of attack. Proper architecture is essential in protecting against host level system attacks and preventing compromise of data through tunneled communication streams.
- **Cryptography** . Cryptography enables applications to communicate and send information between other applications, networks, and users in an undecipherable context. Encryption is the most powerful form of cryptography and is pivotal in preventing information leakage, data compromise.
- **Database Access** . Database access limits the connectivity by which applications access the database.

Good database access control curbs entry to a datasource to prevent information theft, modification, addition, and data deletion.

- **Data Validation** . Data validation ensures the integrity of information used by an application. Generally data validation is used for data input and output to ensure proper information is being passed to and from the application to prevent attack surface exposer and the disclosure of sensitive information. For web services it is very important that data be validated properly since their interface specification is published as WSDL which can aid attackers in generating hostile input. Web services are weighted high. Web application and intranet scenarios are weighted medium.
- **Information Disclosure** . Information disclosure is the unauthorized presentation of sensitive or confidential information. Proper application design should safeguard against information disclosure to protect both application end users and intellectual property kept by the application. All application scenarios are weighted medium.
- **Runtime Container Security** . Runtime container security features user, group, and code permissions for an application server runtime environment. Account privileges limit the privileges granted to the runtime environment as a whole, while code access security controls limit the privileges granted to individual applications.
- **Web Services** . Web services provides an infrastructure for parties (computer-to-computer) to communicate with each other securely over an internal network for services in a platform- independent and language-neutral manner. Key components of the platform includes features such as security, distributed transaction management and connection pool management.

The next sections provide further information on each area of analysis, and describe @stake's findings in detail.

## 4.2.1 Application Logging Services

---

### Application Logging Services

Application Logging Services provide an application with a simple to use, yet flexible and robust mechanism for logging output to multiple destinations. Property driven parameters allow an application to easily modify the semantics of logging without changing any of the application's code. Application Logging Services allow an application to audit application events, errors, unexpected events, session metrics, and any other information pertinent to an administrator and/or application user in order to help them use and maintain an application.

The Application Logging Services area of analysis includes these topics:

- Exception Management
- Logging Privileges
- Log Management

The table below describes the topics at a high level, and lists prevailing best practices. The right-hand columns show the weights @stake assigned for each of the scenarios.

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Exception Management</b> . Exception Management allows an application to capture and track erroneous events that could affect proper application processing. A well-designed application should contain exception management mechanisms that not only detect exceptions but audit exception anomalies to enable monitoring and logging of these events. This is weighted medium for all application scenarios.	The monitoring components log security events in the case of failures, errors, unexpected behavior and other exception conditions.	2	2	2
<b>Logging Privileges</b> . Restricted privileges to access the logs are most important in an environment that has applications from multiple organizations running on the same machine such as in a hosted environment. Logging privileges is weighted medium for web applications and web services and low for intranet.	When logging application events, the server process does so in a write-only fashion. Other processes are denied write-access to the service. Logging occurs on a separate, protected server to ensure that logs cannot be tampered with by attackers.	2	2	1
<b>Log Management</b> . Authentication and privilege management should be logged by all application in order to create an audit trail. This is weighted medium for the three application scenarios.	The application server provides logging with sufficient detail to permit reconstruction of system activity.	2	2	2

Scenario weights range from 1 (relatively unimportant) to 3 (extremely important). A scenario weight of zero means that the topic does not apply.

## Findings

The table below summarizes @stake's quantitative findings. For each analysis topic, the two high-level metrics (Best Practice Compliance and Ease of Securing) are shown. These are the arithmetic means of the underlying best practice and test case scores. The Ease of Securing metric is further broken down into four sub-metrics.

Topic	Platform	Best Practice Compliance	Ease of Securing				
			Overall	Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Exception Management <i>1 best practice, 2 test cases</i>	.NET Windows	<b>3</b>	<b>3.88</b>	3	5	4	3.5
	WebSphere Linux	<b>2</b>	<b>2.38</b>	2	2.5	4	1
	WebSphere Unix	<b>2</b>	<b>2.38</b>	2	2.5	4	1
Logging Privileges <i>1 best practice, 3 test cases</i>	.NET Windows	<b>2</b>	<b>2.83</b>	2.33	2.67	4	2.33
	WebSphere Linux	<b>4</b>	<b>4</b>	4.33	3.33	4	4.33
	WebSphere Unix	<b>4</b>	<b>4</b>	4.33	3.33	4	4.33
Log Management <i>1 best practice, 3 test cases</i>	.NET Windows	<b>4</b>	<b>4.33</b>	5	4	4	4.33
	WebSphere Linux	<b>3</b>	<b>3.42</b>	4.33	2.67	3	3.67
	WebSphere Unix	<b>3</b>	<b>3.42</b>	4.33	2.67	3	3.67

This area of analysis was comprised of eight (8) test cases cover the topics of Exception Management, Logging Privileges, and Log Storage Management. Microsoft scored substantially higher in areas of Exception Management and slightly better in Log Storage Management. IBM scored slightly higher in the area of Logging Privileges. Overall, IBM's solution of integrating third party products for certain functionality in the WebSphere environment did not compare favorably to Microsoft's more coherent integration strategy.

The individual sections for each analysis topic describe @stake's findings in further detail.

## 4.2.1.1 Exception Management

---

### Exception Management

Exception Management allows an application to capture and track erroneous events that could affect proper application processing. A well-designed application should contain exception management mechanisms that not only detect exceptions but audit exception anomalies to enable monitoring and logging of these events. This is weighted medium for all application scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Exception Management topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=57]. The monitoring components log security events in the case of failures, errors, unexpected behavior and other exception conditions.	The .NET Framework exposes the Windows event logging functionality to programmers and makes it easy for them to leverage it for both local and remote logging.	<b>Developer extends (3)</b> . Only a handful of lines of code are required to use event logs.	By default, the ASP.NET process does not have permissions to create event sources. This is by design since event source creation requires writing to the registry. Developers should ensure that event sources are created at install time, or administratively, and not by elevating the ASP.NET process' permissions so it can create them during run-time.

**Best Practice Compliance score**, based on mean of 1 best practice: 3

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=57.1]. Configure the server to log security events. Induce a fault in the application server; determine if components behaved as expected.	Implementation Complexity	<b>Small amount of code (3)</b> . A small number of lines of code (i.e. under 10 lines) are required to log to an event.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation in "Building Secure Microsoft ASP.NET Applications" on writing to event logs is very clear and easy to follow. Also mentioned are the privileges required to create event sources in the registry.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive web application development experience but very little with ASP and ASP.NET.
	Time to Implement	<b>Medium (3)</b> . Most of this time was spent finding the appropriate documentation and creating a test event source. The code to write to the event log itself was very quickly written.
[id=57.2]. Configure the server to log security events to another server. Induce a fault in the application server; determine if messages are sent as expected.	Implementation Complexity	<b>Small amount of code (3)</b> . Extending the code in the test case above to write to another server is very simple: the server name needs to be specified when creating the event source.
	Documentation and Examples	<b>Best practice (5)</b> . Information on how to log to a separate server was very easy to find using Visual Studio .NET help function to look up the CreateEventSource method.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive web application development experience but very little with ASP and ASP.NET.
	Time to Implement	<b>Low to Medium (4)</b> . Once the relevant documentation was found, the code changes were trivial.

**Ease of Securing score**, based on mean of 2 test cases: 3.88

### Notes

#### Test Case 57.1

The developer created a small ASP.NET program that enables users to write to the Application event log.

See also: EventLog.zip

#### Test Case 57.2

See the code from the above test case for an example of how to log to another server.

### IBM WebSphere J2EE

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=57]. The monitoring components log security events in the case of failures, errors, unexpected behavior and other exception conditions.	WebSphere supports multiple types of logs. For some log types the event threshold can be varied, as can the physical location and size of the log files. Logging to the syslog daemon is not supported.	<b>Developer implements (2)</b> . The WebSphere logging options are insufficient from a number of perspectives: control over certain logs is incomplete, logging is not integrated into the OS logging facilities, and logging to remote servers is not supported.	A package such as log4j is recommended to address these issues.

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 2, Unix 2

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=57.1]. Configure the server to log security events. Induce a fault in the application server; determine if components behaved as expected.	Implementation Complexity	<b>Medium amount of code (2)</b> . In order to log security events to the syslog daemon, a third-party package such as log4j must be installed.
	Documentation and Examples	<b>Adequate (3)</b> . The documentation explains how to access and configure the existing WebSphere logs but makes no mention of security events nor logging to standard system logs.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering WebSphere application servers.
	Time to Implement	<b>High (1)</b> . Installing, configuring, and integrating log4j into the application architecture will take some time but only needs to be performed once.
[id=57.2]. Configure the server to log security events to another server. Induce a fault in the application server; determine if messages are sent as expected.	Implementation Complexity	<b>Medium amount of code (2)</b> . Logging to another server should be accomplished through a syslog (or syslog-like) facility and therefore requires the use of log4j.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Logging to separate servers is not documented.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering WebSphere application servers.
	Time to Implement	<b>High (1)</b> . Installing, configuring, and integrating log4j into the application architecture will take some time but only needs to be performed once.

**Ease of Securing scores**, based on mean of 2 test cases: Linux 2.38, Unix 2.38

## Notes

### Test Case 57.1

WebSphere supports the following log types:

- JVM logs: these are redirected System.out and System.err streams. There is one set of logs for each JVM.
- Process logs: are created by redirecting the stdout and stderr streams of the JVM process. There is

one set of process logs for each JVM.

- IBM Service log: contains both messages written to `System.out` and WebSphere-specific diagnostic messages. Only one service log per application server exists. It is written in binary format and can be analyzed with a purpose-made tool, the Log Analyzer (located in `bin/waslogbr`).

See also: [log4j main page](#), and [log4j's SyslogAppender](#).

## 4.2.1.2 Logging Privileges

---

### Logging Privileges

Restricted privileges to access the logs are most important in an environment that has applications from multiple organizations running on the same machine such as in a hosted environment. Logging privileges is weighted medium for web applications and web services and low for intranet. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 1

The sections that follow contain the results of @stake's analysis of the Logging Privileges topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=58]. When logging application events, the server process does so in a write-only fashion. Other processes are denied write-access to the service. Logging occurs on a separate, protected server to ensure that logs cannot be tampered with by attackers.	Windows 2003 and .NET support both local and remote logging.	<b>Developer implements (2)</b> . Windows' log capabilities are very comprehensive from a functionality perspective, however some of security benefits (e.g. log isolation) will only truly be reaped when log entries are sent to another server.	For a higher level of confidence in the accuracy of the log entries (which may be esp. necessary for regulatory compliance), logging should be performed to a remote server.

**Best Practice Compliance score**, based on mean of 1 best practice: 2

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=58.1]. Configure the application logging service for write-only access.	Implementation Complexity	<b>Wizard (5)</b> . By default, logs are protected because no users have write access to the log files and Windows exposes no way to programmatically modify log entries. By default, all users can read and add Application log entries. System logs are world-readable but require administrator privileges to write to them. Users wishing to read or clear the Security log need additional rights.
	Documentation and Examples	<b>Suitable (4)</b> . Documentation is clear but, given its copious nature, requires some searching before the relevant docs are found.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web application and limited skills in configuring/administering Windows servers.
	Time to Implement	<b>Low (5)</b> . No configuration changes need to be made.
[id=58.2]. Configure the application logging service to deny write access to other processes.	Implementation Complexity	<b>Large amount of code (1)</b> . In the amount of time allocated to this task, the developer was unable to find a way to prevent some local processes from adding entries to existing and/or custom log files.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Specific information on this topic was not found. The above answer was arrived at through empirical tests.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web application and limited skills in configuring/administering Windows servers.
	Time to Implement	<b>High (1)</b> . Much time was spent looking for documentation on this topic. As none was found the developer tested various permission combinations to deny log access to certain processes.
[id=58.3]. Attempt to access log stores using a process other than the server's; verify that the file cannot be modified.	Implementation Complexity	<b>Large amount of code (1)</b> . As mentioned above, preventing log access requires logging to an external server
	Documentation and Examples	<b>Vague or incomplete (2)</b> . See comments above.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web application and limited skills in configuring/administering Windows servers.
	Time to Implement	<b>High (1)</b> . Work on this test case and the one above it was combined.
<b>Ease of Securing score</b> , based on mean of 3 test cases: 2.83		

## Notes

### Test Case 58.1

Although an attacker may not be able to modify the logs, he or she could create enough fake entries in the Application log as to force it to either fill-up or rotate. The first case will typically generate operator alerts but the attacker's activities will no longer be monitored. In the second case, an attacker will create many entries once an attack has occurred in order to cover their tracks.

### Test Case 58.2

In order to deny log access to other processes, logs should be written to a remote server that enforces authenticated access.

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=58]. When logging application events, the server process does so in a write-only fashion. Other processes are denied write-access to the service. Logging occurs on a separate, protected server to ensure that logs cannot be tampered with by attackers.	By default, the WebSphere 5.0 logs are owned by root, readable by all (owner/group/other), but only writable by root.	<b>Wizard (4)</b> . Making the log files writable-only requires executing the chmod command.	For most users' needs, the default log settings are secure. For increased security, consider logging to a remote host.

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 4, Unix 4

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=58.1]. Configure the application logging service for write-only access.	Implementation Complexity	<b>Small amount of code (3)</b> . The user must find the specific logs to protect (e.g. service logs, process logs, JVM logs, etc.) and run chmod to prevent other users from reading them thus ensuring their write-only nature.
	Documentation and Examples	<b>Suitable (4)</b> . The documentation gives clear explanations of each log and its purpose. The help files installed with the application server itself have limited information but the IBM website makes up for this.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web application and limited skills in configuring/administering WebSphere application servers.
	Time to Implement	<b>Medium (3)</b> . The majority of the time was spent reading the documentation in order to understand which log files to change.
[id=58.2]. Configure the application logging service to deny write access to other processes.	Implementation Complexity	<b>Wizard (5)</b> . No changes need be made since only root can write to the logs.
	Documentation and Examples	<b>Adequate (3)</b> . No specific information is available on this topic. However any Unix administrator should be able to verify that non-root processes cannot access the logs.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web application and limited skills in configuring/administering WebSphere application servers.
	Time to Implement	<b>Low (5)</b> . Once the previous test case was performed, this case was trivial to execute.

### Level of Effort Analysis

---

Test Case	Criteria	Rating
[id=58.3]. Attempt to access log stores using a process other than the server's; verify that the file cannot be modified.	Implementation Complexity	<b>Wizard (5)</b> . No changes need be made since only root can write to the logs.
	Documentation and Examples	<b>Adequate (3)</b> . No specific information is available on this topic. However any Unix administrator should be able to verify that non-root processes cannot access the logs.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web application and limited skills in configuring/administering WebSphere application servers.
	Time to Implement	<b>Low (5)</b> . Once the previous test case was performed, this case was trivial to execute.

**Ease of Securing scores**, based on mean of 3 test cases: Linux 4, Unix 4

---

### 4.2.1.3 Log Management

---

## Log Management

Authentication and privilege management should be logged by all application in order to create an audit trail. This is weighted medium for the three application scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Log Management topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

## Microsoft .NET

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=56]. The application server provides logging with sufficient detail to permit reconstruction of system activity.		<b>Wizard (4)</b> . Windows 2003 logs authentication activity out of the box and only one configuration change is required to fulfill the other two test cases.	

**Best Practice Compliance score**, based on mean of 1 best practice: 4

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=56.1]. Ensure that logging functionality provides authentication activity logging.	Implementation Complexity	<b>Wizard (5)</b> . Authentication activity is logged by default.
	Documentation and Examples	<b>Suitable (4)</b> . Documentation is comprehensive but the developer found it easier to use the tools directly rather than read documentation.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . The developer has extensive experience developing web application and limited skills in configuring/administering Windows servers.
	Time to Implement	<b>Low (5)</b> . No actions needed to be taken other than verify that log entries were indeed being created for authentication events.
[id=56.2]. Ensure that logging functionality provides account/role creation/deletion logging.	Implementation Complexity	<b>Wizard (5)</b> . Enabling the logging of account/role operations only requires some configuration settings.
	Documentation and Examples	<b>Suitable (4)</b> . Documentation is comprehensive but the developer found it easier to use the tools directly rather than read documentation.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . The developer has extensive experience developing web application and limited skills in configuring/administering Windows servers.
	Time to Implement	<b>Medium (3)</b> . Although an easy configuration, some time was spent finding the area where changes must be made (i.e. in Administrative Tools/Local Security Policy).
[id=56.3]. Ensure that logging functionality provides for monitoring changes in role and privileges.	Implementation Complexity	<b>Wizard (5)</b> . Audit changes in privilege levels requires the same configuration change as the one above (auditing of account/role operations).
	Documentation and Examples	<b>Suitable (4)</b> . Documentation is comprehensive but the developer found it easier to use the tools directly rather than read documentation.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . The developer has extensive experience developing web application and limited skills in configuring/administering Windows servers.
	Time to Implement	<b>Low (5)</b> . No changes needed to be made.

Ease of Securing score, based on mean of 3 test cases: 4.33

## IBM WebSphere J2EE

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=56]. The application server provides logging with sufficient detail to permit reconstruction of system activity.	Several difficulties arise when attempting to create audit events in an EJB compliant J2EE environment. The notes section of the following test cases will more fully explain the challenges of implementing a robust logging and audit frame work. Due to difficulties encountered during the research of this best practice, a least common denominator approach to application logging was taken. The test cases in this best practice area will be implemented using the <code>log()</code> of the base <code>javax.servlet.GenericServlet</code> class.	<b>Developer extends (3)</b> . Basic logging and audit functionality is possible with a moderate amount of code using the built in logging facility of the <code>GenericServlet</code> . However, depending on the size and scope of a deployed application the primitive logging capabilities of the Servlet and its derivatives lack scalability and flexibility.	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 3, Unix 3

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=56.1]. Ensure that logging functionality provides authentication activity logging.	Implementation Complexity	<b>Small amount of code (3)</b> . A rating of three is given only in keeping to the letter of the test-case. Implementing a simple authentication log was accomplished in less than a dozen lines of code. Though it is impossible to extrapolate the code complexity of more sophisticated logging schemes from such simple requirements, one could assume at least a two order of magnitude increase
	Documentation and Examples	<b>Suitable (4)</b> . Documentation on implementing Servlet logging is widely available. The vendor sites provided suitable documentation to implement the test case.
	Implementor Competence	<b>Intermediate (3)</b> . The tester has wide experience in configuring and deploying web based applications.
	Time to Implement	<b>Medium (3)</b> . The required modifications to a single Java class were accomplished in 30 minutes.
[id=56.2]. Ensure that logging functionality provides account/role creation/deletion logging.	Implementation Complexity	<b>Wizard (5)</b> . Using a properties sheet wizard it was possible to configure the server to generate trace messages to audit role change events,
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Documentation on setting tracing and logging was barely adequate to accomplish the goals of the test case.
	Implementor Competence	<b>Intermediate (3)</b> . The tester has wide experience in configuring and deploying web based applications.
	Time to Implement	<b>Low to Medium (4)</b> . The required to the server configuration file required less than 5 minutes.

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=56.3]. Ensure that logging functionality provides for monitoring changes in role and privileges.	Implementation Complexity	<b>Wizard (5)</b> . Using a properties sheet wizard it was possible to configure the server to generate trace messages to audit role change events,
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Documentation on setting tracing and logging was barely adequate to accomplish the goals of the test case.
	Implementor Competence	<b>Intermediate (3)</b> . The tester has wide experience in configuring and deploying web based applications.
	Time to Implement	<b>Low to Medium (4)</b> . The required to the server configuration file required less than 5 minutes.

**Ease of Securing scores**, based on mean of 3 test cases: Linux 3.42, Unix 3.42

### Notes

#### Test Case 56.1

One of the primary difficulties with implementing a logging framework for an EJB compliant application is the restrictions on calling primitive file input/output operators from the confines of the container.

The EJB 1.1 Specification , section 18.1.2 (Programming Restrictions) states the following:

*An enterprise bean must not use the java.io package to attempt to access files and directories in the file system.*

*The file system APIs are not well-suited for business components to access data. Business components should use a resource manager API, such as JDBC API, to store data.*

To overcome this basic restriction, logging frameworks must perform logging operations via a resource connection supported by the EJB programming model. EJB servers may support several resource connection options, which can be used to log events as shown below:

- JDBC (javax.sql.DataSource) : Write log events in to a relational database.
- URL (java.net.URL) : Write log events to a custom logging server or post them to a web server.
- JMS (javax.jms.QueueConnectionFactory | javax.jms.TopicConnectionFactory) : Send the log events as messages.

Each of these methods have their benefits and liabilities. All of them were deemed to be to complex to implement given the overall time restrictions of the comparison project.

Other possible logging solutions that were not explored: Log4j and other logging solutions

log4j

The Jakarta Commons Logging Project

The `java.util.logging` class

Below is a source code example created by extending the `LogonAction` class created during a previous test case:

```

boolean validated = false;

        try {
            validated = isUserLogon(username,password);
        } catch (UserDirectoryException ude) {
            ActionErrors errors = new ActionErrors();
            errors.add(ActionErrors.GLOBAL_ERROR,
            new ActionError("error.logon.connect"));
            saveErrors(request,errors);
            return (new ActionForward(mapping.getInput()));
        }
        if (!validated) {
            // credentials don't match
            ActionErrors errors = new ActionErrors();
            errors.add(ActionErrors.GLOBAL_ERROR,
            new ActionError("error.logon.invalid"));
            saveErrors(request,errors);
            // return to input page
            return (new ActionForward(mapping.getInput()));
        }

        // Save our logged-in user in the session,
        // because we use it again later.
        HttpSession session = request.getSession();
        session.setAttribute(Constants.USER_KEY, form);

        // Log this event, if appropriate
        if (servlet.getDebug() >= Constants.DEBUG) {
            StringBuffer message =
                new StringBuffer("LogonAction: User ''");
            message.append(username);
            message.append('\' logged on in session ');
            message.append(session.getId());
            servlet.log(message.toString());
        }
        return (mapping.findForward(Constants.SUCCESS));
    }
}

```

## Test Case 56.2

Setting the following server configuration properties results in role change event notification to the log file:

```

Current trace specification =
com.ibm.ws.security.core.SecurityCollaborator=all=enabled
com.ibm.ws.security.role.RoleBasedConfiguratorImpl=all=enabled
com.ibm.ws.security.role.RoleBasedModule=all=enabled
com.ibm.ws.security.web.WebAuthenticator=all=enabled
com.ibm.ws.security.web.WebCollaborator=all=enabled
com.ibm.ws.security.web.WebSecurityContext=all=enabled

```



## 4.2.2 Authentication and Access Control

---

### Authentication and Access Control

Authentication enables content control by only allowing access to confirmed, designated users. Good Authentication also facilitates accountability of actions. These functions are most important for external applications. Web application and web services are weighted as high and intranet as medium.

The Authentication and Access Control area of analysis includes these topics:

- Login Management
- Role Based Access Control
- Web Server Integration

The table below describes the topics at a high level, and lists prevailing best practices. The right-hand columns show the weights @stake assigned for each of the scenarios.

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Login Management</b> . Login management functions are most important for external applications. Web application and web services are weighted as high and intranet as medium.	All users are authenticated before being given access to restricted information and functionality. There is only one mechanism through which users can authenticate to the system. Upon logout, users are no longer able to access any of the application's protected functionality.	3	3	2
<b>Role Based Access Control</b> . Roles are a powerful way to manage access control. They are most important for external applications. Web application and web services are weighted as high and intranet as medium.	Each role, upon creation, defaults to a deny-all stance, i.e. roles are created with no permissions. This helps ensure that roles will be given the minimum privileges necessary. Role-to-principal binding is supported. Ideally this is done declaratively, via a mechanism which does not exist on the web tier.	3	3	2
<b>Web Server Integration</b> . Applications should be able to leverage the built in authentication of the web server. This is weighted as medium for all application scenarios.	Application framework uses the built-in authentication provided by the web server.	2	2	2

Scenario weights range from 1 (relatively unimportant) to 3 (extremely important). A scenario weight of zero means that the topic does not apply.

### Findings

The table below summarizes @stake's quantitative findings. For each analysis topic, the two high-level metrics (Best Practice Compliance and Ease of Securing) are shown. These are the arithmetic means of the underlying best practice and test case scores. The Ease of Securing metric is further broken down into four sub-metrics.

Topic	Platform	Best Practice Compliance	Ease of Securing				
			Overall	Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Login Management <i>2 best practices, 2 test cases</i>	.NET Windows	<b>3.5</b>	<b>4.38</b>	4	4	4.5	5
	WebSphere Linux	<b>3</b>	<b>2.88</b>	3	3	3	2.5
	WebSphere Unix	<b>3</b>	<b>2.88</b>	3	3	3	2.5
Role Based Access Control <i>2 best practices, 6 test cases</i>	.NET Windows	<b>4</b>	<b>3.92</b>	3.5	4.17	4	4
	WebSphere Linux	<b>4.5</b>	<b>4.42</b>	4.83	4	4.5	4.33
	WebSphere Unix	<b>4.5</b>	<b>4.42</b>	4.83	4	4.5	4.33
Web Server Integration <i>1 best practice, 1 test case</i>	.NET Windows	<b>5</b>	<b>5</b>	5	5	5	5
	WebSphere Linux	<b>4</b>	<b>2.75</b>	3	2	3	3
	WebSphere Unix	<b>4</b>	<b>2.75</b>	3	2	3	3

This area of analysis was comprised of nine (9) test cases covering the platform's ability to securely deal with, users, groups and roles. Of the test cases executed, Microsoft scored slightly higher in the topics of Web Server Integration and Login Management whereas IBM was found to score slightly higher in the topic of Role Based Access Control

The individual sections for each analysis topic describe @stake's findings in further detail.

## 4.2.2.1 Login Management

---

### Login Management

Login management functions are most important for external applications. Web application and web services are weighted as high and intranet as medium. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 3
- Web Service: 3
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Login Management topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=49]. All users are authenticated before being given access to restricted information and functionality. There is only one mechanism through which users can authenticate to the system.	Authentication methods are configured through a checkbox in the configuration manager.	<b>Wizard (4) .</b>	
[id=50]. Upon logout, users are no longer able to access any of the application's protected functionality.	.NET provides a method to be called to log out a user and delete its session.	<b>Developer extends (3) .</b>	No recommendation.

**Best Practice Compliance score**, based on mean of 2 best practices: 3.5

### Level of Effort Analysis

---

Test Case	Criteria	Rating
[id=49.1]. Configure the container for user authentication.	Implementation Complexity	<b>Wizard (5)</b> . Implementation is performed through checkboxes.
	Documentation and Examples	<b>Best practice (5)</b> . Documentation is adequate.
	Implementor Competence	<b>Novice (5)</b> . No development work is necessary.
	Time to Implement	<b>Low (5)</b> . Implementation is performed through checkboxes.
[id=50.1]. Log out user; verify non-access to application.	Implementation Complexity	<b>Small amount of code (3)</b> . Session deletion is implemented via one function call.
	Documentation and Examples	<b>Adequate (3)</b> . Documentation was sufficient to execute the test case but no mention could be found of the need to implement logout functionality in a application.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . The developer has extensive web application development experience but very little with ASP and ASP.NET.
	Time to Implement	<b>Low (5)</b> . The amount of time required is trivial.

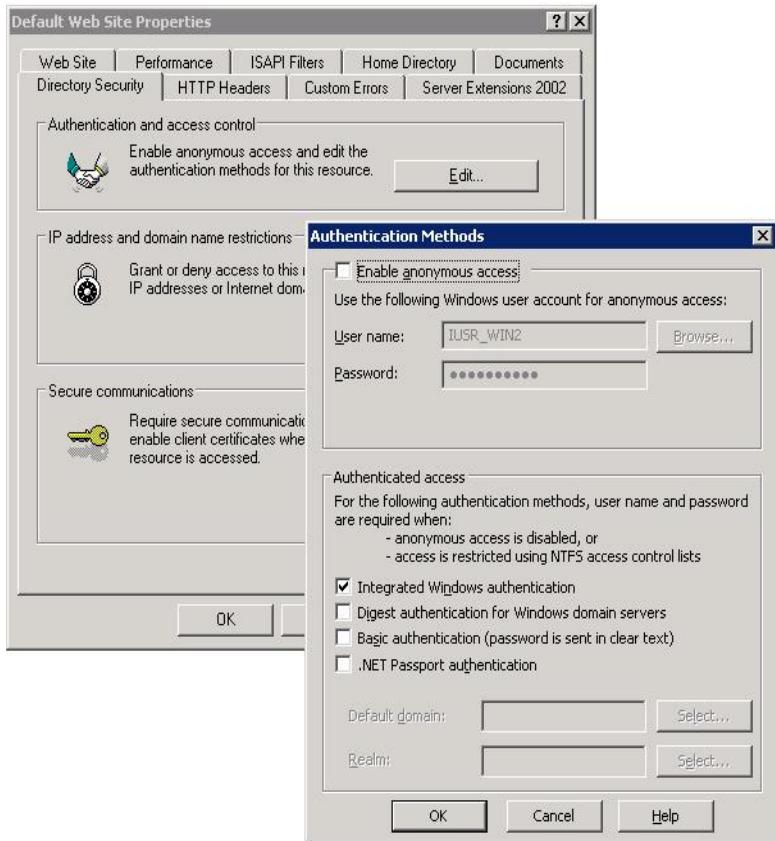
**Ease of Securing score**, based on mean of 2 test cases: 4.38

---

### Notes

#### Test Case 49.1

Authentication is implemented through the use of checkboxes.



## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=49]. All users are authenticated before being given access to restricted information and functionality. There is only one mechanism through which users can authenticate to the system.	The WebSphere Application Server can be configured to authenticate users by using one of several authentication mechanisms. Given the wide array of methods available a least common denominator approach to user authentication was taken. The following test case assumed a Local Operating System User Registry in conjunction with HTTP-BasicAuth	<b>Developer extends (3).</b> Implementing what was assumed to be the simplest user authentication scheme proved to be an extremely difficult task. Documentation was scant at best, often confusing, and neglected to mention several implementation pitfalls.	

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=50]. Upon logout, users are no longer able to access any of the application's protected functionality.	The Java HttpSession interface is used by a servlet container to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user. It may be programmatically destroyed or invalidated at any time by the container. The test case within this best practice will be implemented using this paradigm.	<b>Developer extends (3)</b> . A moderate amount of code was expended to implement this functionality. Using Session Invalidation as a means of logging a client off is a well understood programming technique in Servlet based applications.	

**Best Practice Compliance scores**, based on mean of 2 best practices: Linux 3, Unix 3

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=49.1]. Configure the container for user authentication.	Implementation Complexity	<b>Small amount of code (3)</b> . Configuring the WebSphere container for user authentication was an extremely complex task requiring the modification of 2 files in 7 distinct steps. This was accomplished using WebSphere Application Developer Studio. A further set of complex tasks had to be done using the Application Server administration console in order to complete the test case.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Documentation on performing the configuration tasks was geared entirely to more sophisticated methods of user authentication (e.g. LDAP and DB2). The scant documentation found for performing Operating System integrated authentication was often confusing, out of date, and in some cases incorrect.
	Implementor Competence	<b>Intermediate (3)</b> . The developer has wide experience in the development and deployment of web applications
	Time to Implement	<b>High (1)</b> . Given the lack of sound documentation performing this test-case took an inordinate amount of effort
[id=50.1]. Log out user; verify non-access to application.	Implementation Complexity	<b>Small amount of code (3)</b> . Code complexity to implement session invalidation is low.
	Documentation and Examples	<b>Suitable (4)</b> . API documentation, the Servlet specification and numerous examples are available on the sites of both IBM and the Apache Group
	Implementor Competence	<b>Intermediate (3)</b> . The developer has wide experience in the development and deployment of web applications
	Time to Implement	<b>Low to Medium (4)</b> . The analyst was able to accomplish this test case in under 15 minutes

**Ease of Securing scores**, based on mean of 2 test cases: Linux 2.88, Unix 2.88

## Notes

Test Case 49.1

### test\_case\_49\_1 Web Deployment Descriptor

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app id="WebApp">
    <display-name>TestCase_49_1Web</display-name>
    <servlet>
        <servlet-name>index</servlet-name>
        <display-name>index</display-name>
        <jsp-file>/index.jsp</jsp-file>
    </servlet>
    <servlet>
        <servlet-name>protected</servlet-name>
        <display-name>protected</display-name>
        <jsp-file>/protected.jsp</jsp-file>
        <security-role-ref>
            <role-name>admin</role-name>
            <role-link>admin</role-link>
        </security-role-ref>
    </servlet>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
        <welcome-file>default.html</welcome-file>
        <welcome-file>default.htm</welcome-file>
        <welcome-file>default.jsp</welcome-file>
    </welcome-file-list>
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>Protected Resources</web-resource-name>
            <description></description>
            <url-pattern>/protected.jsp</url-pattern>
            <http-method>
                GET</http-method>
            <http-method>
                PUT</http-method>
            <http-method>
                HEAD</http-method>
            <http-method>
                TRACE</http-method>
            <http-method>
                POST</http-method>
            <http-method>
                DELETE</http-method>
            <http-method>
                OPTIONS</http-method>
        </web-resource-collection>
        <auth-constraint>
            <description></description>
            <role-name>admin</role-name>
        </auth-constraint>
    </security-constraint>
    <login-config>
        <auth-method>BASIC</auth-method>
```

```

<realm-name>test_case_49_1-realm</realm-name>
</login-config>
<security-role>
    <description>Test Case 49.1 admin role</description>
    <role-name>admin</role-name>
</security-role>
</web-app>

```

### test\_case\_49\_1 Application Deployment Descriptor

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE application PUBLIC "-//Sun Microsystems, Inc.//DTD J2EE Application 1.3//EN"
"http://java.sun.com/dtd/application_1_3.dtd">
<application id="Application_ID">
    <display-name>TestCase_49_1</display-name>
    <module id="JavaClientModule_1049939953487">
        <java>TestCase_49_1Client.jar</java>
    </module>
    <module id="EjbModule_1049939953497">
        <ejb>TestCase_49_1EJB.jar</ejb>
    </module>
    <module id="WebModule_1049939953507">
        <web>
            <web-uri>TestCase_49_1Web.war</web-uri>
            <context-root>TestCase_49_1Web</context-root>
        </web>
    </module>
    <security-role>
        <description>Test Case 49.1 admin role</description>
        <role-name>admin</role-name>
    </security-role>
</application>

```

### Documentation Sources

The IBM web site provided Redbook documentation describing The WebSphere authentication model

Further documentation regarding user authentication may be found in the IBM WebSphere V5.0 Security WebSphere Handbook

### Test Case 50.1

The code below is an example of using `ActionForm` target of a JSP page to effect session invalidation. Any attempt to access a protected resource in a application thereafter throws an exception

```

1 package com.atstake.test_case;
2
3 import java.io.IOException;
4 import javax.servlet.RequestDispatcher;
5 import javax.servlet.ServletException;

```

```
6 import javax.servlet.http.HttpServletRequest;
7 import javax.servlet.http.HttpSession;
8 import javax.servlet.http.HttpServletResponse;
9 import org.apache.struts.action.Action;
10 import org.apache.struts.action.ActionForm;
11 import org.apache.struts.action.ActionForward;
12 import org.apache.struts.action.ActionMapping;
13 import org.apache.struts.action.ActionServlet;
14 import org.apache.struts.util.MessageResources;

15 public final class LogoffAction extends Action {
16     public ActionForward perform(ActionMapping mapping,
17         ActionForm form,
18         HttpServletRequest request,
19         HttpServletResponse response)
20         throws IOException, ServletException {
21
22     HttpSession session = request.getSession();
23     LogonForm user = (LogonForm)
24         session.getAttribute(Constants.USER_KEY);
25
26     // unbind the user, then nuke the session.
27
28     session.removeAttribute(Constants.USER_KEY);
29     session.invalidate();
30
31     // Return success
32     return (mapping.findForward(Constants.SUCCESS));
33 }
34 }
```

## Documentation Sources

Information regarding HttpSession

## 4.2.2.2 Role Based Access Control

---

### Role Based Access Control

Roles are a powerful way to manage access control. They are most important for external applications. Web application and web services are weighted as high and intranet as medium. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 3
- Web Service: 3
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Role Based Access Control topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=11]. Each role, upon creation, defaults to a deny-all stance, i.e. roles are created with no permissions. This helps ensure that roles will be given the minimum privileges necessary.	All role-based authentication is built on top of the Windows group paradigm.	<b>Transparent (5)</b> . Windows group creation defaults to a deny-all stance.	Using the appropriate MMC snap-in, an administrator can easily create new groups.
[id=12]. Role-to-principal binding is supported. Ideally this is done declaratively, via a mechanism which does not exist on the web tier.	<p>Role-to-user binding can be performed differently depending on the user count:</p> <ul style="list-style-type: none"> <li>• Static users - The role can be assigned through the OS user management wizards.</li> <li>• Dynamic users - This includes public web sites. In this case, the developer can choose to implement an Active Directory or SQL server store.</li> </ul> <p>Both cases are remarkably well supported by .NET.</p>	<b>Developer extends (3)</b> . The .NET Framework provides for role-based functionality but leaves too many implementation details up to the developer.	

**Best Practice Compliance score**, based on mean of 2 best practices: 4

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=11.1]. Create a role to protect a given file.	Implementation Complexity	<b>Wizard (5)</b> . Group creation is trivial to accomplish.
	Documentation and Examples	<b>Best practice (5)</b> . Documentation is clear.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited Windows administration.
	Time to Implement	<b>Low (5)</b> .
[id=11.2]. Create a role to protect a given web page.	Implementation Complexity	<b>Small amount of code (3)</b> . This can be achieved via file system permissions and ASP.NET configuration. File system permissions are trivial to set but ASP.NET configurations are more complex to specify.
	Documentation and Examples	<b>Adequate (3)</b> . No documentation specific to sub-folder authorization
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited Windows administration.
	Time to Implement	<b>Low to Medium (4)</b> . Once found, the configuration options are easy to set.
[id=11.3]. Create a role to protect a given web service.	Implementation Complexity	<b>Wizard+ (4)</b> . The steps involved with securing a web service are identical to those involved in securing a web application.
	Documentation and Examples	<b>Suitable (4)</b> . Documentation includes best practices and implementation patterns.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited Windows administration.
	Time to Implement	<b>Low to Medium (4)</b> . Configuration changes are quick to make.
[id=11.4]. Create a role to protect a given object.	Implementation Complexity	<b>Small amount of code (3)</b> . This level of role-based authorization must be implemented programmatically.
	Documentation and Examples	<b>Best practice (5)</b> . See step-by-step instructions in link below.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited Windows administration.
	Time to Implement	<b>Medium (3)</b> . Some coding is necessary to support role-based access control which, ideally, could have been auto-generated.
[id=11.5]. Create a role to protect a given method.	Implementation Complexity	<b>Small amount of code (3)</b> . A declarative security statement can be used to implement this test case
	Documentation and Examples	<b>Best practice (5)</b> . Very detailed best practice documentation available. See below for link.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited .NET exposure.
	Time to Implement	<b>Low (5)</b> . This is a very simple feature to add.

### Level of Effort Analysis

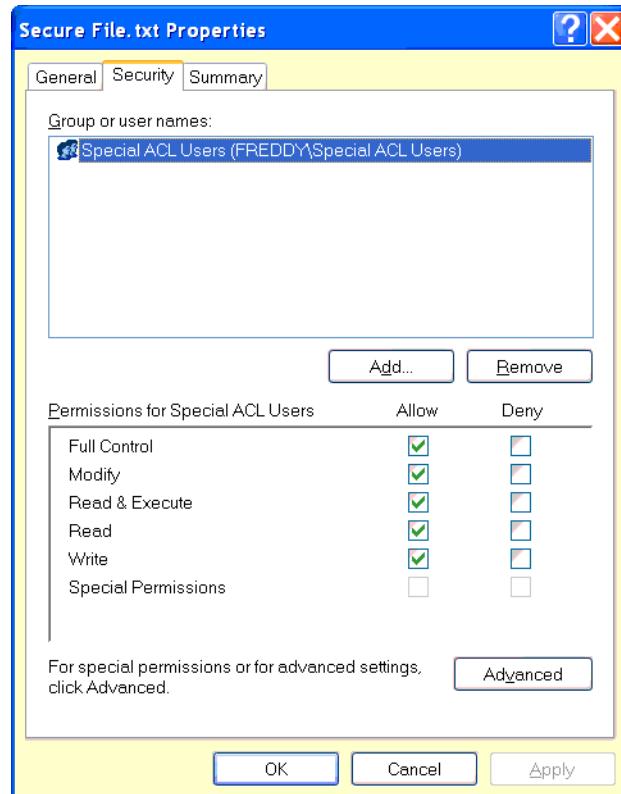
Test Case	Criteria	Rating
[id=12.1]. Create a principal (user); demonstrate binding to a role.	Implementation Complexity	<b>Small amount of code (3)</b> . Some code still needs to be written to execute this test case.
	Documentation and Examples	<b>Adequate (3)</b> . @stake could find several samples on this topic, but no conceptual documentation covering both architecture and security best practices.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited .NET exposure.
	Time to Implement	<b>Medium (3)</b> . The code was relatively quick to implement.

**Ease of Securing score**, based on mean of 6 test cases: 3.92

### Notes

#### Test Case 11.1

Once the new role is created, the administrator can protect a file by assigning the new role to a file's security descriptor:



### Test Case 11.2

Role based access control on a specific URL can be achieved multiple ways:

- Windows Access Control Lists - Using IIS impersonation and setting an appropriate ACL on the actual file if the web page is a file. This requires impersonation.
- URL Authorization - Using web.config.

The following example grants access to authenticated users, while denying it to everyone but "atstake1" for the subdirectory "SubDir".

```

<authentication mode="Forms" >
    <forms name="ManualLogin" path="/" 
        loginUrl="ManualLogin.aspx" protection="All" timeout="1">
    </forms>
</authentication>
<authorization>
    <deny users="?" />
</authorization>
</system.web>
<!-- Configuration for the "SubDir" subdirectory. -->
<location path="SubDir">
    <system.web>
        <authorization>
            <allow users="atstake1" />
            <deny users="*" />
        </authorization>
    </system.web>
</location>
</configuration>

```

See also: Designing Distributed Applications with Visual Studio .NET - Authorization

.NET Framework Developer's Guide - Configuration <location> Settings

### Test Case 11.4

Using the IPrincipal interface, querying for roles associated to the current user is trivial.

The requirements for implementing Application\_AuthenticateRequest could be more transparent, as it requires coding that could be implemented in the default behavior:

```

Private Sub Global_AuthenticateRequest(ByVal sender As Object, ByVal e As
System.EventArgs)
    Handles MyBase.AuthenticateRequest
    // Extract the forms authentication cookie
    string cookieName = FormsAuthentication.FormsCookieName;
    HttpCookie authCookie = Context.Request.Cookies[cookieName];

    if(null == authCookie)

```

```

{
// There is no authentication cookie.
return;
}

FormsAuthenticationTicket authTicket = null;
try
{
authTicket = FormsAuthentication.Decrypt(authCookie.Value);
}
catch(Exception ex)
{
// Log exception details (omitted for simplicity)
return;
}

if (null == authTicket)
{
// Cookie failed to decrypt.
return;
}

// When the ticket was created, the UserData property was assigned a
// pipe delimited string of role names.
string[] roles = authTicket.UserData.Split(new char[]{'|'});

// Create an Identity object
FormsIdentity id = new FormsIdentity( authTicket );

// This principal will flow throughout the request.
GenericPrincipal principal = new GenericPrincipal(id, roles);
// Attach the new principal object to the current HttpContext object
Context.User = principal;
End Sub

```

See also: How To: Create GenericPrincipal Objects with Forms Authentication

### Test Case 11.5

The following code will ensure that only authenticated HR users can access the web service method called WhoAmI( ):

```

[WebMethod]
[PrincipalPermissionAttribute(SecurityAction.Demand,
    Authenticated=true,
    Role="HR")]
public string WhoAmI() {
    return "Running as User : " +
        Thread.CurrentPrincipal.Identity.Name;
}

```

See also: HTTP Security and ASP.NET Web Services

### Test Case 12.1

The role information can be stored with the authentication ticket:

```
// Create the authentication ticket
FormsAuthenticationTicket authTicket = new
    FormsAuthenticationTicket(1,           // version
                            txtUserName.Text,      // user name
                            DateTime.Now,          // creation
                            DateTime.Now.AddMinutes(60), // Expiration
                            false,                 // Persistent
                            roles );               // User data
```

Retrieving it is then as simple as

```
IPrincipal p = HttpContext.Current.User;
Response.Write( "Authenticated Identity is: " +
                p.Identity.Name );
Response.Write( "<p>" );

if ( p.IsInRole("Senior Manager") )
```

Retrieving the role associated to a specific user still needs to be implemented. In the case of a Database store, it would involve writing a stored procedure to retrieve the roles associated with a specific user. This can be done at the same time as credentials are verified. See How To: Use Forms Authentication with SQL Server 2000 for an example of credential verification through stored procedure call that can be extended to support roles.

See also: Visual Studio Samples: Duwamish 7.0 - Security

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=11]. Each role, upon creation, defaults to a deny-all stance, i.e. roles are created with no permissions. This helps ensure that roles will be given the minimum privileges necessary.	Roles are created in WebSphere studio using the property sheets for the application deployment descriptors. Developers can create web and EJB components to create roles independently from the application server administrator, who maps them to an external authentication store at deploy-time.	<b>Transparent (5)</b> . Until the application server administrator explicitly maps an application role to named users or groups in the authentication store, the container prohibits access to content protected by the role.	None; WebSphere implements best practices.

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=12]. Role-to-principal binding is supported. Ideally this is done declaratively, via a mechanism which does not exist on the web tier.	Principals (users) are mapped to roles at deploy-time by the WebSphere application server administrator. When the application is installed, security roles are mapped to groups (and optionally, to named users) in the external identity store. Mappings can also be modified post-installation.	<b>Wizard (4)</b> . The Administration Console provides a simple wizard-like interface for installing applications (.war and .ear files). Administrators can look up the users and groups at deploy-time to verify that the right principals are being mapped.	No recommendations; WebSphere implements the best practice.

**Best Practice Compliance scores**, based on mean of 2 best practices: Linux 4.5, Unix 4.5

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=11.1]. Create a role to protect a given file.	Implementation Complexity	<b>Wizard (5)</b> . Role-based file security is defined in the web tier deployment descriptor (web.xml), which is implemented through several property sheet-style pages. Developers can assign roles to particular URL patterns, specify the authentication method (Basic, Digest, form-based, client certificates) and optionally mandate the use of SSL.
	Documentation and Examples	<b>Suitable (4)</b> . Documentation is clear and thorough. All options are described, although the examples provided do not explicitly demonstrate SSL configuration.
	Implementor Competence	<b>Novice (5)</b> . The property sheet interface is very simple and clean. Resolution of role across a larger web application, however, might require more than just novice skills.
	Time to Implement	<b>Low (5)</b> . Declaring roles in the web.xml deployment descriptor is point-and-click simple. Note that in order for the roles to be implemented at run-time, the server administrator must bind the roles to real users at deploy time.
[id=11.2]. Create a role to protect a given web page.	Implementation Complexity	<b>Wizard (5)</b> . Roles protecting web pages under the application web root are protected in the same manner as for files; see above.
	Documentation and Examples	<b>Suitable (4)</b> . See above.
	Implementor Competence	<b>Novice (5)</b> . See above.
	Time to Implement	<b>Low (5)</b> . See above

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=11.3]. Create a role to protect a given web service.	Implementation Complexity	<b>Wizard+ (4)</b> . The method we chose to implement (adding a <security-constraint> to the web.xml file for the SOAP router Servlets.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . IBM's documentation and website mention many ways to deploy web services, but the information is disorganized. Important information on the method we used was adequate at best, and did not discuss the "secure SOAP messaging" option (apparently, this is WS-Security compliant) in any detail. Limitations of the method we chose were not mentioned.
	Implementor Competence	<b>Novice/intermediate (4)</b> . This reviewer has significant experience with Java, but very little with web services. The wizards presumed basic knowledge of SOAP terminology in general, and was aided by some specific knowledge of Apache SOAP, the package upon which the generated code relies.
	Time to Implement	<b>Medium to High (2)</b> . Although running the wizards to generate the web services took only a minute or two, deciding which of the multiple possible approaches to use took several hours.
[id=11.4]. Create a role to protect a given object.	Implementation Complexity	<b>Wizard (5)</b> . Enterprise Java Beans (EJBs) are the preferred method for implementing protected objects in J2EE; the <ejb-jar.xml> file provides a declarative method for defining roles-based access to EJBs. WebSphere Studio provides an easy-to-use property sheet for configuring EJB role assignments.
	Documentation and Examples	<b>Best practice (5)</b> . The Redbook documentation was very good on procedures, although a bit light on examples. Fortunately, this is a well-understood and supported part of the J2EE specification, and examples abound on IBM's website.
	Implementor Competence	<b>Novice/intermediate (4)</b> . This evaluator has approximately two years of J2EE experience, but very little is required (other than conceptual knowledge of how the J2EE role-based security model works).
	Time to Implement	<b>Low (5)</b> . Changing the deployment descriptor and re-deploying the application took only a few minutes.
[id=11.5]. Create a role to protect a given method.	Implementation Complexity	<b>Wizard (5)</b> . Security for individual object methods (in this case, EJB local and remote methods) is configured in a manner identical to that used for the EJB itself: through the property sheets.
	Documentation and Examples	<b>Best practice (5)</b> . See the test case for object security, above.
	Implementor Competence	<b>Novice/intermediate (4)</b> . See the test case for object security, above.
	Time to Implement	<b>Low (5)</b> . See the test case for object security, above.
[id=12.1]. Create a principal (user); demonstrate binding to a role.	Implementation Complexity	<b>Wizard (5)</b> . The Administration Console interface is simple and clean.
	Documentation and Examples	<b>Suitable (4)</b> . The examples and documentation were clear and concise.
	Implementor Competence	<b>Novice (5)</b> . The mapping tool provides dynamic user/group lookup functions, making it easy to spot errors. We expect that even very inexperienced server administrators should be able to quickly perform the mapping tasks.
	Time to Implement	<b>Low to Medium (4)</b> . Assuming that external authentication has been set up correctly, role-to-principal mapping takes only a few minutes to complete.

### Level of Effort Analysis

Test Case	Criteria	Rating
<b>Ease of Securing scores</b> , based on mean of 6 test cases: Linux 4.42, Unix 4.42		

### Notes

#### Test Case 11.1

The <security-constraint> and <login-config> portions of the web.xml file allow developers to declare security constraints and authentication methods for files and server resources under the application webroot:

```

<security-constraint>
    <web-resource-collection>
        <web-resource-name>Protected Pages</web-resource-name>
        <description></description>
        <url-pattern>
            /protected.jsp</url-pattern>
        <http-method>
            GET</http-method>
        <http-method>
            PUT</http-method>
        <http-method>
            HEAD</http-method>
        <http-method>
            TRACE</http-method>
        <http-method>
            POST</http-method>
        <http-method>
            DELETE</http-method>
        <http-method>
            OPTIONS</http-method>
    </web-resource-collection>
    <auth-constraint>
        <description></description>
        <role-name>admin</role-name>
    </auth-constraint>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>Best Practices</realm-name>
</login-config>
<security-role>
    <description>For administrators.</description>
    <role-name>admin</role-name>
</security-role>

```

Based on the constraints declared in this file, the WebSphere application server protects files under the application webroot. For files outside the webroot (*e.g.*, static content external to the application), the administrator of the web server would restrict access by modifying the IBM HTTPD httpd.conf file. This is not really part of the application context, but the Security Redbook provides examples on how to do this anyhow.

Note that the J2EE Servlet specification 2.3 provides for three types of ‘transport guarantees’ for session security. **None** (no session security *a.k.a.* plain old HTTP) is the default. The other two, **Integral** and **confidential** are essentially both synonymous with SSL and can generally be used interchangeably across Servlet containers; Jakarta Tomcat, for example, also treats Integral and Confidential as meaning SSL. The reason for the synonymity is unclear to us; it is likely an artifact of an earlier revision in the Servlet specification.

### Test Case 11.3

IBM, confusingly, provides at least four methods for generating web services. WebSphere Studio 5.0 includes a series of wizards that will generate web services from Java beans, Enterprise Java Beans, and database schema. We used this method in our tests. The others web service technologies, described in the **WebSphere Version 5 Web Services Handbook** are:

- IBM WebSphere SDK for Web Services (WSDK)
- IBM Web Services Toolkit (WSTK)
- IBM WebSphere Web Services for J2EE Technology Preview

Because this analysis assessed the out-of-the box security features of WebSphere, we elected to use the wizards built-in to WebSphere Studio. However, in the near future one can expect IBM to provide a much more elegant, tightly-integrated solution that complies with JSR-109 (a forthcoming deployment descriptor format which will be introduced with J2EE 1.4).

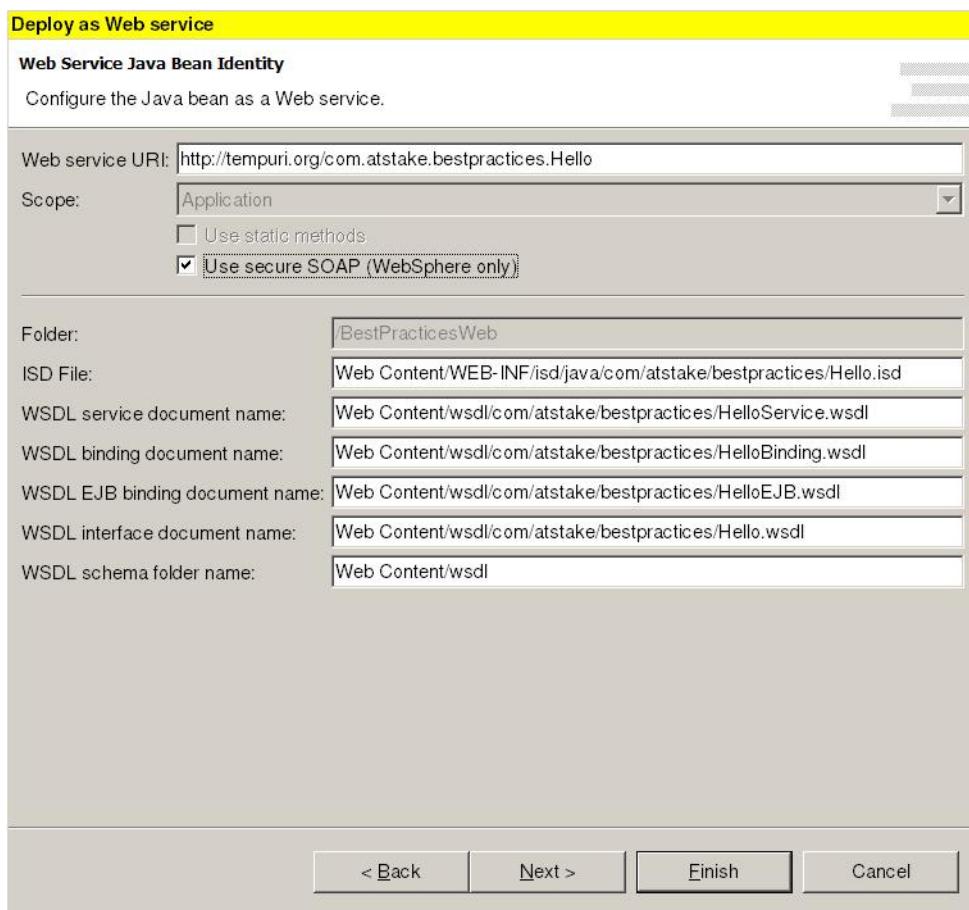
To create a web service, we build a small stateless EJB session bean and ran the web service wizard to generate services that corresponded to methods in the EJB. The practice of using EJBs as the base class for web services aligns well with JSR-109, which mandates the use of web-service-specific deployment descriptors for EJBs.

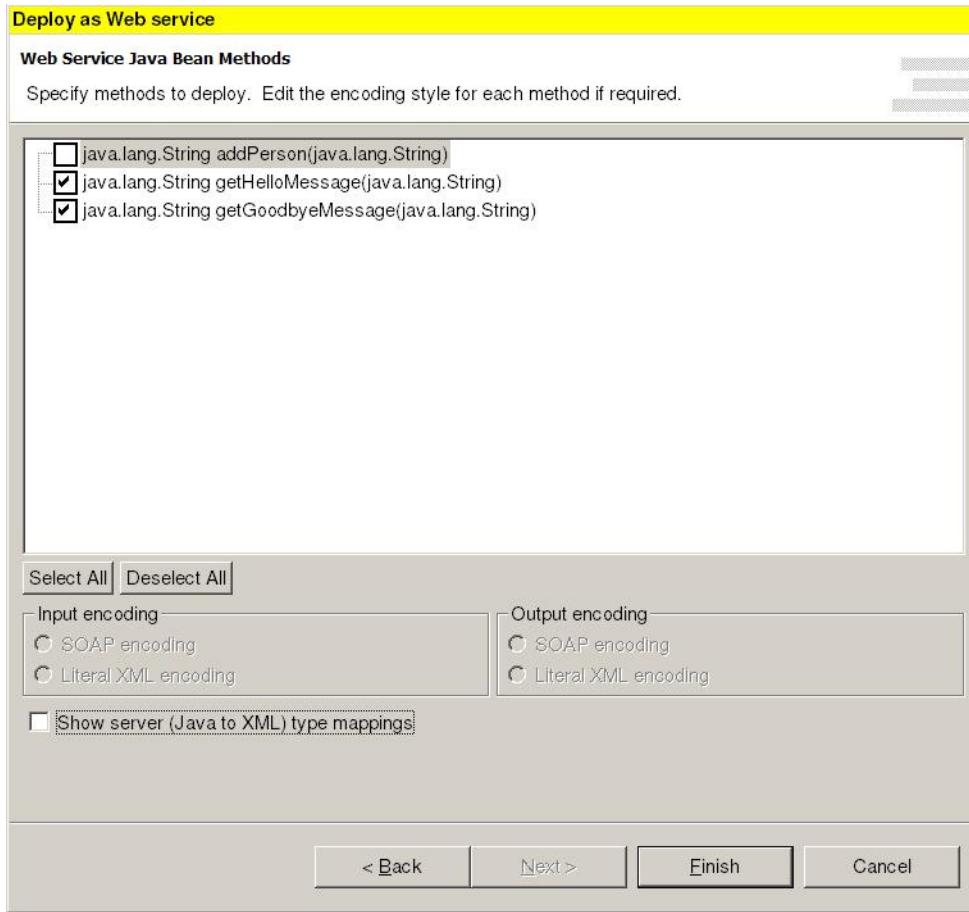
Generating the web service was easy. The Hello EJB session bean had two method signatures that we elected to expose as web services:

- `java.lang.String getHelloMessage(java.lang.String userName)`
- `java.lang.String getGoodbyeMessage(java.lang.String userName)`

The first method was protected by role **admin**; the second was unprotected. We assigned the role the standard J2EE EJB way, by modifying the ejb-jar deployment descriptor’s property sheet. Then we simply generated the service by right-clicking the HelloBean and selecting **Deploy as Web Service** from the context menu. The wizard stepped us through the process, providing a series of default settings and suggested web service bindings. One quirk: the developer needed to be in the J2EE Perspective and J2EE Hierarchy view in order for the EJB to be recognized as such by the wizard.

Here are two screen shots from the wizard:



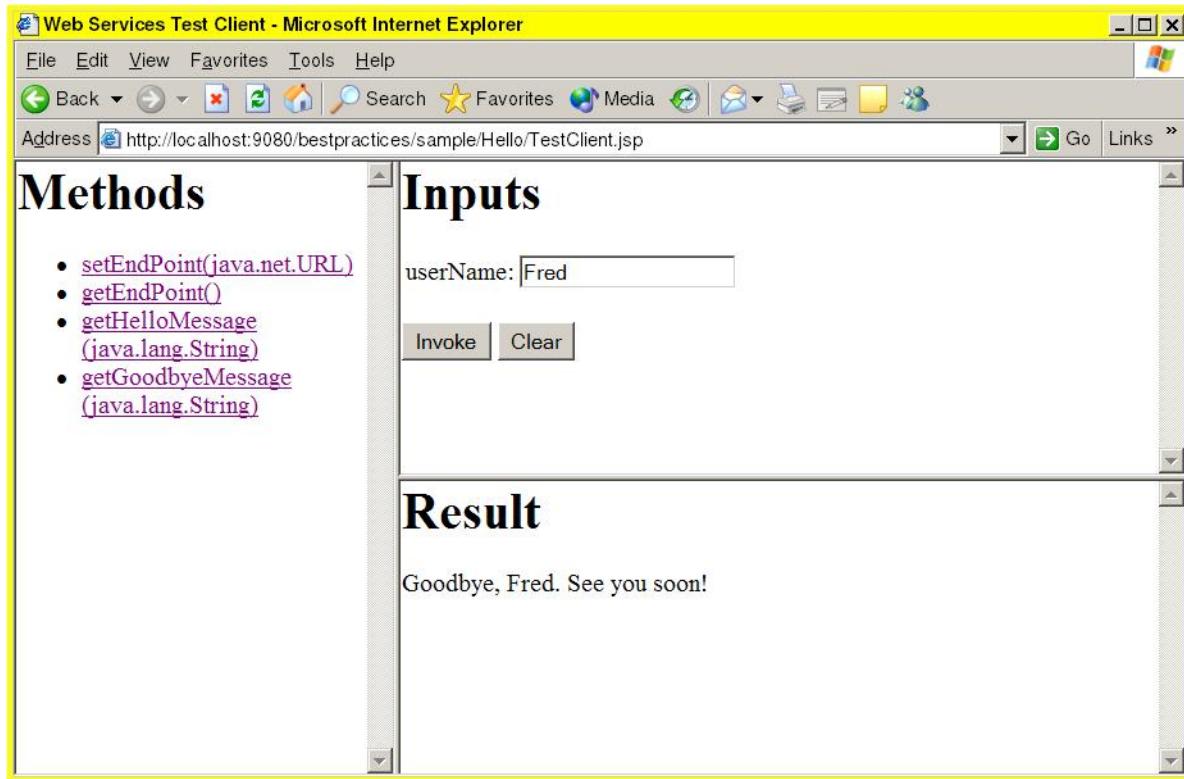


After completion of the wizard, WebSphere generated files and classes to implement the web services. When used with the default settings (**Use secure SOAP** is unchecked) WebSphere studio generated these directories in the web project's Web Context folder: /admin (9 files), /log (1 files), and /wsdl (4 files). WebSphere also placed two files in the web context root, three Apache SOAP-related runtime jars in the /WEB-INF/lib directory, and a web services descriptor file (Hello.wsdl) in /WEB-INF/wsdl. Finally, our web.xml deployment descriptor was modified to add two Servlets: rpcrouter and messagerouter, to map incoming POSTs to the SOAP RPC and message router, respectively.

A second web services wizard (Generate Sample Application), created a small set of JSPs to test the web services in /sample/Hello (4 files).

To protect the web service, we added a <security-constraint> to the web.xml deployment descriptor to protect the /rpcrouter and /messagerouter Servlets used for routing SOAP messages.

Here is a screen shot showing a successful web service call, after authentication to the application server:



While we implemented the test case successfully, the approach we used has several limitations, notably that it lacks role granularity for a given soap router instance.

A far more elegant and secure solution would be to use <ejb-jar.xml> deployment descriptor for role assignments, which would mean that object and method (and by extension, web service) security could be enforced at all levels of the container, rather than at just the web tier. Indeed, this is the goal of JSR-109. We found several mentions of support for an experimental JSR-109 compliant "technology preview" of JSR-109 toolkit, but we did not attempt to implement it.

On a side note, Chapter 15 of *WebSphere Version 5 Web Services Handbook* contains this sentence: "There is some confusion about the different packages for developing Web services on top of WebSphere Application Server and in this chapter we explain the differences between [the packages]." Indeed.

#### Test Case 11.4

WebSphere Studio provides an easy-to-use property sheet for role-based access control to EJBs. A typical role declaration, contained in the <ejb-jar.xml> file bundled into the EJB JAR file at application assembly time, looks like this:

```
<ejb-jar>
  ...
<assembly-descriptor>
  <security-role>
```

```
<description>Administrator Only</description>
<role-name>admin</role-name>
</security-role>
<method-permission>
    <role-name>admin</role-name>
    <method>
        <ejb-name>Hello</ejb-name>
        <method-intf>Remote</method-intf>
        <method-name>getHelloMessage</method-name>
        <method-params>
            <method-param>java.lang.String</method-param>
        </method-params>
    </method>
</method-permission>
</assembly-descriptor>
</ejb-jar>
```

See also: The *WebSphere V5.0 Security Handbook*, Chapter 5, provides excellent procedural detail on how to configure role-based security for EJBs. To enable object security, WebSphere Global Security *must* be enabled.

### Test Case 12.1

The WebSphere Administration Console provides the method through which users in the external identity store are mapped to application roles. Role-to-principal mapping assumes that the identity store for the J2EE container (as a whole) has been defined. This is done once per server; see the analysis topic External Authentication for details on how this is done.

Assuming the identity store has been defined, the Administration Console provides a straightforward method for mapping roles to principals:

The screenshot shows the WebSphere Administrative Console interface. On the left, a sidebar lists navigation options: Home, Save, Preferences, Logout, Help, User ID: websphere, ARJ-HOOVER, Servers, Applications (with links to Enterprise Applications and Install New Application), Resources, Security, Environment, System Administration, and Troubleshooting.

The main content area displays the 'Enterprise Applications > BestPractices > Mapping Users to Roles' page. It includes instructions: 'Map security roles to users/groups' and 'Each role defined in the application or module must be mapped to a user or group from the domain's user registry.' Below these are two buttons: 'Lookup users' and 'Lookup groups'. A table titled 'Role' lists one entry: 'admin' under 'Role', 'Everyone?' under 'Everyone?', and 'All Authenticated?' under 'All Authenticated?'. The 'Mapped Users' column is empty, and the 'Mapped Groups' column contains 'cn=admin,dc=atstake,dc=com'. At the bottom are 'OK' and 'Cancel' buttons.

Below this is the 'WebSphere Status' screen, which shows runtime messages. The message count is Total All Messages: 516, with 36 new and 36 total errors, 27 new and 27 total warnings, and 453 new and 453 total informational messages. There are buttons for 'Clear All' and 'Preferences'.

A second screen (not shown) contains a user and group lookup screen that the application server administrator can use to verify the run-time users and groups that will be read from the external identity store.

### 4.2.2.3 Web Server Integration

---

#### Web Server Integration

Applications should be able to leverage the built in authentication of the web server. This is weighted as medium for all application scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Web Server Integration topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

#### Microsoft .NET

##### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=51]. Application framework uses the built-in authentication provided by the web server.	.NET is tightly integrated with the Web server and operating system, making full use of existing authentication mechanisms.	<b>Transparent (5)</b> . Built-in authentication mechanisms are used directly.	

**Best Practice Compliance score**, based on mean of 1 best practice: 5

##### Level of Effort Analysis

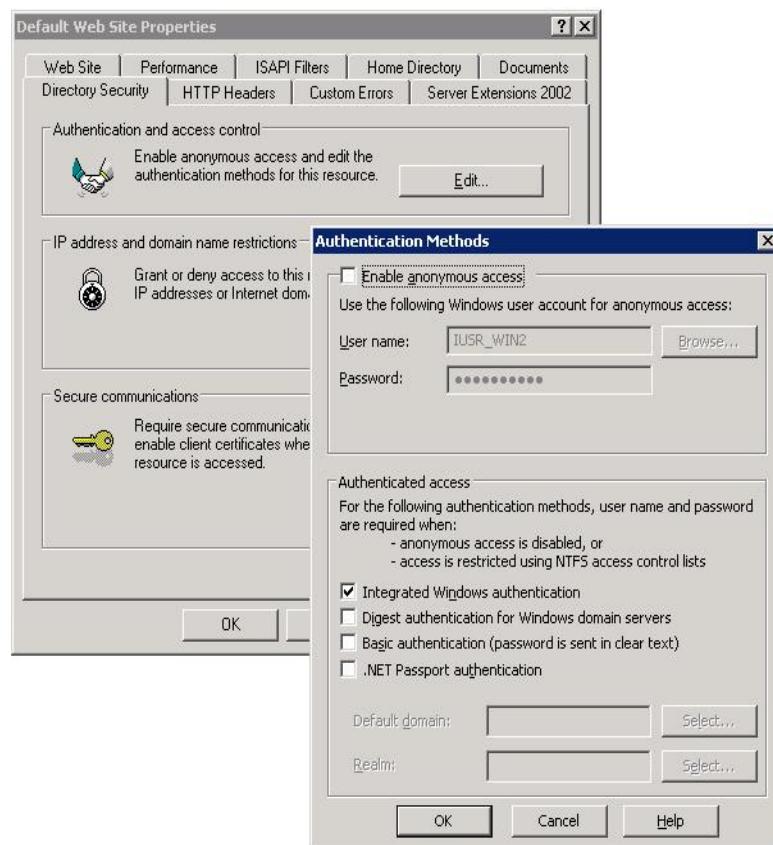
Test Case	Criteria	Rating
[id=51.1]. Configure authentication to use web server's built-in authentication.	Implementation Complexity	<b>Wizard (5)</b> . Implementation is performed through checkboxes.
	Documentation and Examples	<b>Best practice (5)</b> . Documentation is adequate.
	Implementor Competence	<b>Novice (5)</b> . No development work is necessary.
	Time to Implement	<b>Low (5)</b> . Implementation is performed through checkboxes.

**Ease of Securing score**, based on mean of 1 test case: 5

## Notes

### Test Case 51.1

Authentication is implemented through the use of checkboxes.



## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=51]. Application framework uses the built-in authentication provided by the web server.	The WebSphere Application Server can be configured to authenticate by means of the build in Web Server Authentication methods  The following test case assumed a Local Operating System User Registry in conjunction with HTTP-BasicAuth	<b>Wizard (4) .</b>	

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
<b>Best Practice Compliance scores</b> , based on mean of 1 best practice: Linux 4, Unix 4			

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=51.1]. Configure authentication to use web server's built-in authentication.	Implementation Complexity	<b>Small amount of code (3)</b> . Authentication methods are specified during the application development using a somewhat counter-intuitive set of dialogs, property sheets and configuration files greatly adding to the overall complexity of the task.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Documentation on configuring the container to perform integrated Web Authentication was scant.
	Implementor Competence	<b>Intermediate (3)</b> . The developer has wide experience in the development and deployment of web applications
	Time to Implement	<b>Medium (3)</b> . using lessons learned in previous test cases, the analyst was able to accomplish this test case in under 1 hour

**Ease of Securing scores**, based on mean of 1 test case: Linux 2.75, Unix 2.75

### Notes

#### Test Case 51.1

WebSphere supplies a wide range of integrated authentication methods for accessing Web resources protected by an authorization constraint. A client must authenticate by using the configured authentication mechanism. A Web application can authenticate a user to a Web server by using one of the following mechanisms: HTTP basic authentication, HTTP digest authentication, HTTPS client authentication, and form-based authentication.

#### Documentation Sources

The IBM web site provided Redbook documentation describing The WebSphere authentication model

Further documentation regarding user authentication may be found in the IBM WebSphere V5.0 Security WebSphere Handbook

## 4.2.3 Communications

---

### Communications

Architecture allows for the proper function of an application without opening additional avenues of attack. Proper architecture is essential in protecting against host level system attacks and preventing compromise of data through tunneled communication streams.

The Communications area of analysis includes these topics:

- Communication Security
- Network-Accessible Services

The table below describes the topics at a high level, and lists prevailing best practices. The right-hand columns show the weights @stake assigned for each of the scenarios.

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Communication Security</b> . Communication Security covers the security-related options present in the various ways an external source communicates with the application. Options included integrity, authentication, encryption, and anti-replay capabilities. This is weighted medium for all application scenarios.	The application only accepts SSLv3 and TLSv1 encryption. Minimum key size is 128 bits.	2	2	2
<b>Network-Accessible Services</b> . The network accessible services are those ports that are accessible by some system other than the local host. These ports are opened by some internal process and typically are associated with a specific service. It is important that access to these ports be limited for applications that are external. This feature is weighted medium for external applications and low for intranet applications.	Use base operating system network filtering utilities or host hardening techniques to limit access to platform components e.g., database servers, the directory server, etc.	2	2	1

Scenario weights range from 1 (relatively unimportant) to 3 (extremely important). A scenario weight of zero means that the topic does not apply.

### Findings

The table below summarizes @stake's quantitative findings. For each analysis topic, the two high-level metrics (Best Practice Compliance and Ease of Securing) are shown. These are the arithmetic means of the

underlying best practice and test case scores. The Ease of Securing metric is further broken down into four sub-metrics.

Topic	Platform	Best Practice Compliance	Ease of Securing				
			Overall	Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Communication Security <i>1 best practice, 2 test cases</i>	.NET Windows	<b>3</b>	<b>3.75</b>	4	3	4	4
	WebSphere Linux	<b>4</b>	<b>3.63</b>	5	2	4	3.5
	WebSphere Unix	<b>4</b>	<b>3.63</b>	5	2	4	3.5
Network-Accessible Services <i>1 best practice, 1 test case</i>	.NET Windows	<b>4</b>	<b>3.75</b>	4	5	2	4
	WebSphere Linux	<b>4</b>	<b>3.25</b>	4	4	1	4
	WebSphere Unix	<b>3</b>	<b>3</b>	3	4	1	4

@stake executed three (3) test cases designed to evaluate each platform's ability to encrypt communications using SSL and IPSec. @stake found that .NET Framework and WebSphere provide good support for securing HTTP using SSL, although we noted a few cases where documentation could be improved. For securing network ports used by applications, both platforms can be configured to use IPSec.

The individual sections for each analysis topic describe @stake's findings in further detail.

## 4.2.3.1 Communication Security

---

### Communication Security

Communication Security covers the security-related options present in the various ways an external source communicates with the application. Options included integrity, authentication, encryption, and anti-replay capabilities. This is weighted medium for all application scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Communication Security topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### **Best Practice Compliance Analysis**

Best Practice	Options	Rating	Recommendation
[id=55]. The application only accepts SSLv3 and TLSv1 encryption. Minimum key size is 128 bits.	Support for selecting 128bit encryption is built into the management console but changing the settings requires manual edits in the registry and restarting IIS.	<b>Developer extends (3)</b> . Although the Crypto API does provide the necessary flexibility to specify protocols and ciphers, IIS does not provide an UI to easily configure them.	No recommendation.

**Best Practice Compliance score**, based on mean of 1 best practice: 3

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=55.1]. Configure server to use SSLv3 and TLSv1 only.	Implementation Complexity	<b>Small amount of code (3)</b> . Manual registry edits are required.
	Documentation and Examples	<b>Adequate (3)</b> . Finding the appropriate documentation required more efforts than expected. @stake suggests the knowledge base article mentioned below be referenced more often in the product web site.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited Windows and IIS administration.
	Time to Implement	<b>Medium (3)</b> . Once the knowledge base article was found, a few minutes were sufficient to implement the changes.
[id=55.2]. Configure servers to set key size no less than 128 bits.	Implementation Complexity	<b>Wizard (5)</b> . The web site security properties provide a checkbox to enforce a minimum of 128 bits.
	Documentation and Examples	<b>Adequate (3)</b> . More information could be provided on the consequences of enabling the checkbox (such as browser support).
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited Windows and IIS administration.
	Time to Implement	<b>Low (5)</b> . The change is trivial to make.

**Ease of Securing score**, based on mean of 2 test cases: 3.75

### Notes

#### Test Case 55.1

See Microsoft Knowledge Base Article - 245030: How to Restrict the Use of Certain Cryptographic Algorithms and Protocols in Schannel.dll for a step-by-step description of the process.

#### Test Case 55.2

@stake noticed two issues with the configuration UI:

- A web site cannot only deliver HTTPS content as it requires a non-SSL TCP port. One can disable it by "requiring SSL" on the security property page. This suggests a single web site serve both insecure and secure content when best practices dictate two distinct sites to limit the risk of information disclosure.
- In order to force 128 bit encryption, one has to check "Require secure channel". @stake failed to understand the relationship between these two checkboxes.

### IBM WebSphere J2EE

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=55]. The application only accepts SSLv3 and TLSv1 encryption. Minimum key size is 128 bits.	The WebSphere HTTPS Transport (port 9443) only supports SSLv3, however the Administrative Console enables users to choose between low (no encryption), medium (40-bit), and high (128-bit) encryption. Users can also pick the individual ciphers they wish to enable.	<b>Wizard (4)</b> . Implementing the best practice is fairly simple through the Administration Console. However the console is misleading and could result in users believing they have enabled strong encryption while WebSphere is still configured to accept weak ciphers.	Ensure that strong ciphers are explicitly specified.

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 4, Unix 4

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=55.1]. Configure server to use SSLv3 and TLSv1 only.	Implementation Complexity	<b>Wizard (5)</b> . SSL v3 is supported by default. Although TLS v1 is not, the developer feels SSL v3 is similar enough to TLS v1 protocol to award full marks in this category.
	Documentation and Examples	<b>Adequate (3)</b> . The documentation explains the meaning of the administrative settings but does not highlight any security issues associated with them.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web application and limited skills in configuring/administering WebSphere application servers.
	Time to Implement	<b>Low (5)</b> . No changes need be made.
[id=55.2]. Configure servers to set key size no less than 128 bits.	Implementation Complexity	<b>Wizard (5)</b> . Restricting SSL ciphers to 128-bit encryption entails selecting the appropriate cipher suites from the Administrative Console.
	Documentation and Examples	<b>Incorrect or insecure (1)</b> . The documentation states that when the Security Level is set to high, only 128-bit ciphers are accepted. This is not the case: the developer was still able to establish SSL sessions using a weak cipher such as DES. Only when strong ciphers were explicitly selected in the Cipher Suites fields and the application server restarted, did the server only accept 128-bit encrypted SSL connections.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web application and limited skills in configuring/administering WebSphere application servers.
	Time to Implement	<b>Medium to High (2)</b> . Much time was lost searching through documentation, restarting servers, verifying SSL cipher suites, etc.

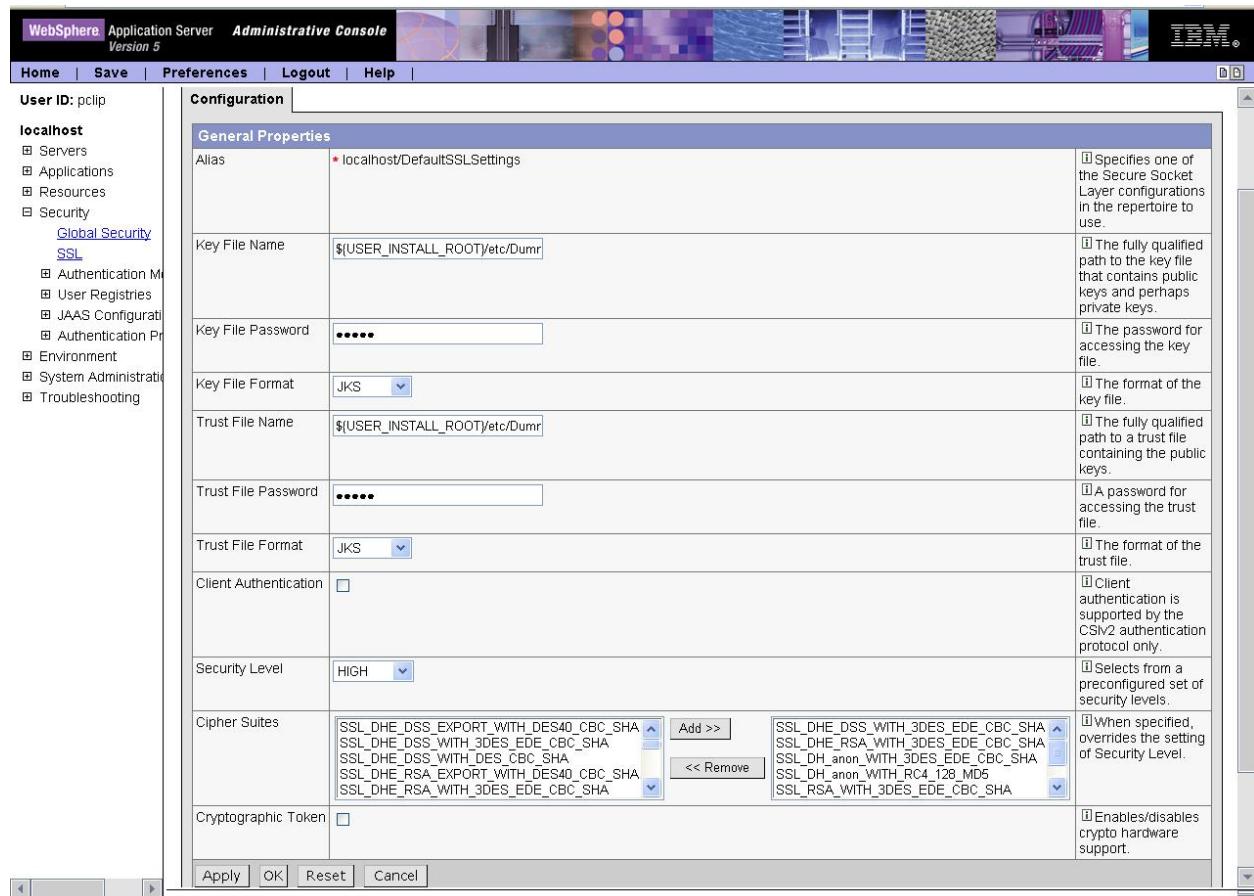
**Ease of Securing scores**, based on mean of 2 test cases: Linux 3.63, Unix 3.63

## Notes

### Test Case 55.2

As can be seen from the Administrative Console below, although the Security Level is set to high by default, it does not prevent using weak ciphers. Instead strong ciphers needed to be explicitly specified in the Cipher

Suites field.



## 4.2.3.2 Network-Accessible Services

---

### Network-Accessible Services

The network accessible services are those ports that are accessible by some system other than the local host. These ports are opened by some internal process and typically are associated with a specific service. It is important that access to these ports be limited for applications that are external. This feature is weighted medium for external applications and low for intranet applications. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 1

The sections that follow contain the results of @stake's analysis of the Network-Accessible Services topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=54]. Use base operating system network filtering utilities or host hardening techniques to limit access to platform components e.g., database servers, the directory server, etc.	Windows 2003 includes the Internet Connection Firewall (ICF), the Routing and Remote Access Basic Firewall, Internet Protocol Security (IPSec), and TCP/IP Filtering. The ICF provides a way to restrict inbound TCP/IP traffic for a single host or server running Internet Connection Sharing (ICS). The Routing and Remote Access service Basic Firewall is a stateful firewall, which combines dynamic packet filtering of network traffic with a set of static packet filters. IPSec provides confidentiality, authentication, and data integrity between two IPSec endpoints. The TCP/IP Filtering provides a way to restrict inbound TCP/IP traffic based on destination TCP or UDP port and IP protocol. Windows 2003 includes the Security Templates snap-in for managing configuration templates that can be used to disable unnecessary services. The templates can be applied to an individual machine with the Security Configuration and Analysis snap-in or to all computers in the Active Directory through Group Policy.	<b>Wizard (4)</b> . The Internet Connection Firewall only requires a single checkbox to be enabled, however, the user must select one of the existing service definitions or create a new one to allow inbound connectivity.	Administrators should consider the Internet Connections Firewall (ICF) if they only need to restrict inbound TCP/IP traffic based on destination IP address and port. However, if the requirements include filtering on protocol, source and destination IP address, or port, then administrators should consider the Routing and Remote Access Basic Firewall. The Basic Firewall provides a similar level of functionality as that in the ICF with the addition of static packet filters for more granular inbound and outbound restrictions. In addition to either ICF or Basic Firewall, administrators should apply an appropriate Security Template that disables all unnecessary services and reduces available attack surfaces. If the server requires an additional level of security that includes confidentiality, authentication, and data integrity between two endpoints, then Internet Protocol Security (IPSec) should be used to filter inbound and outbound traffic and provide the necessary encryption.

**Best Practice Compliance score**, based on mean of 1 best practice: 4

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=54.1]. Configure base operating system to restrict access to back-end database server.	Implementation Complexity	<b>Wizard+ (4)</b> . A local help file was available through the Help and Support Center that explained how to enable and configure filtering.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation available locally in the Help and Support Center is clear and easy to follow for Internet Connection Firewall (ICF), Basic Firewall, Internet Protocol Security (IPSec), and TCP/IP Filtering.
	Implementor Competence	<b>Expert/intermediate (2)</b> . The administrator has extensive experience with the Windows OS family. However, the Internet Connection Firewall (ICF) and Basic Firewall are new to Windows XP and Windows 2003; therefore expectations relative to pre-existing knowledge of the products are low.
	Time to Implement	<b>Low to Medium (4)</b> . Only a few minutes were required to locate a local help file for the Internet Connection Firewall (ICF) and then enable and configure filtering. A few additional minutes were required to identify and configure the custom service definition for Microsoft SQL Server 2000.

**Ease of Securing score**, based on mean of 1 test case: 3.75

## Notes

### Test Case 54.1

The information for this test case was found in the local Help and Support Center and included the following articles: Protecting your home or small office network using Internet Connection Firewall, Securing your network with Basic Firewall, Special IPSec considerations, and TCP/IP Security Features. The information about SQL Server 2000 requirements for a firewall were found in the SQL Server Books online in the article Connections to SQL Server Over the Internet.

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
<i>Linux.</i> [id=54]. Use base operating system network filtering utilities or host hardening techniques to limit access to platform components e.g., database servers, the directory server, etc.	RedHat Advanced Server provides several firewall packages capable of filtering network traffic and limiting access to platform components; ipchains and iptables. These firewall tools can be configured via configuration files or simple interfaces (lokkit). Additionally, the firewall is enabled by default, and custom rules can be configured during installation. TCPWrappers can be used to provide varying levels of access control and can be configured through manual file editing.	<b>Wizard (4)</b> . The firewall is enabled by default during installation. The firewall would have to be customized to accommodate specific services, such as the database server.	The native firewall capabilities should be employed to control access to configured services. iptables, the newer firewall technology implemented in the 2.4 kernel, should be used when possible.
<i>Unix.</i> [id=54]. Use base operating system network filtering utilities or host hardening techniques to limit access to platform components e.g., database servers, the directory server, etc.	The Unix vendor's current distribution provides host-based firewalling capabilities through its IPSec implementation. IPSec can be used to implement basic packet filtering as well as IPSec VPN functionality. TCPWrappers can be used to provide varying levels of access control and can be configured through manual file editing. Additionally, the Unix vendor provides (HOST-BASED FIREWALL), a stateful, packet-filtering firewall package with the current distribution distribution.	<b>Developer extends (3)</b> . The IPSec implementation, though it is primarily a VPN tool, can be used to control access to platform components.	Consider implementing basic packet filtering through the native IPSec implementation. If more fine-grained or complex access control requirements exist, consider installing and configuring the (HOST-BASED FIREWALL) firewall package. For inetd based network services and other libwrap compatible services, use TCPWrappers to implement access controls.

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 4, Unix 3

### Level of Effort Analysis

Test Case	Criteria	Rating
<i>Linux.</i> [id=54.1]. Configure base operating system to restrict access to back-end database server.	Implementation Complexity	<b>Wizard+ (4)</b> . The firewall is already enabled and can be configured during the initial system installation. A simple interface exists for firewall configuration and customization.
	Documentation and Examples	<b>Suitable (4)</b> . System manual (man) pages, hard copy, and online documentation exist to support firewall configuration.
	Implementor Competence	<b>Expert (1)</b> . The implementer is very familiar with Linux, Unix operating systems, security, and firewall practices.
	Time to Implement	<b>Low to Medium (4)</b> . A few minutes to review man pages and online documentation were required to devise the proper firewall policy. The system does not need to be rebooted to effect changes.
<i>Unix.</i> [id=54.1]. Configure base operating system to restrict access to back-end database server.	Implementation Complexity	<b>Small amount of code (3)</b> . An IPSec policy can be created rather easily to restrict access to platform components.
	Documentation and Examples	<b>Suitable (4)</b> . The online Systems Administrator Guide has an appropriate article that describes securing a web server with IPSec. Though this document is written for the previous version of the vendor's Unix distribution, it is applicable to the current version as well.
	Implementor Competence	<b>Expert (1)</b> . The implementer is very familiar with the vendor's distribution of Unix, Unix operating systems in general, security, and firewall practices.
	Time to Implement	<b>Low to Medium (4)</b> . This IPSec configuration takes only a few minutes to research and implement.

**Ease of Securing scores**, based on mean of 1 test case: Linux 3.25, Unix 3

### Notes

#### Linux Test Case 54.1

The relevant configuration files for ipchains and iptables are /etc/sysconfig/ipchains and /etc/sysconfig/iptables, respectively.

#### Unix Test Case 54.1

The default IPSec policy file is /etc/inet/ipsecinit.conf. The online document *System Administration Guide, Volume 3* provides specific examples for using IPSec to limit traffic to allowed services.

## 4.2.4 Cryptography

---

### Cryptography

Cryptography enables applications to communicate and send information between other applications, networks, and users in an undecipherable context. Encryption is the most powerful form of cryptography and is pivotal in preventing information leakage, data compromise.

The Cryptography area of analysis includes these topics:

- Cryptographic Hashing
- Encryption Algorithms
- Key Generation
- Random Number Generation
- Secrets Storage
- XML Cryptography

The table below describes the topics at a high level, and lists prevailing best practices. The right-hand columns show the weights @stake assigned for each of the scenarios.

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Cryptographic Hashing</b> . Cryptography enables applications to communicate and send information between other applications, networks, and users in an undecipherable context. Encryption is the most powerful form of cryptography and is pivotal in preventing information leakage, data compromise.	When hashing is required, platform supports the use of strong, industry standard-algorithms such as SHA1 and MD5.	2	2	2
<b>Encryption Algorithms</b> . Support for industry standard encryption algorithms is an important feature of an application platform. Without it developers will either attempt to integrate a third-party cryptographic library or even develop their own algorithms. Both of these practices are known to have security pitfalls. Encryption algorithms is weighted as medium for all three application scenarios.	When encryption is used, platform supports the use of strong, industry standard-algorithms such as AES, RSA, 3DES, Blowfish, etc. Minimum key size of 128 bits (1024 bits for RSA) is used.	2	2	2

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Key Generation</b> . Secure key generation is an important feature of a secure application platform. Without it developers will either attempt to integrate a third-party cryptographic library or even develop their own algorithms. Key generation is weighted medium for all scenarios.	Key generation uses a quality PRNG. Private portions of key pairs are generated within a secure container and are never exposed to user level processes.	2	2	2
<b>Random Number Generation</b> . Quality random number generation is an important feature of an application platform. Without it, developers will either attempt to integrate a third-party cryptographic library or even develop their own algorithms. Both of these practices are known to have security pitfalls. Random number generation is weighted as medium for all three application scenarios.	Pseudo-random number generators (PRNG) must be based on high-entropy sources and must be statistically random.	2	2	2
<b>Secrets Storage</b> . Secrets must be managed properly or they lose their value for security. Proper secret storage is weighted equally for all three application scenarios.	Keys and credentials necessary for the application to function are protected from attack. This includes not storing keys in plaintext on the hard drive, in source code, within externally-accessible media, or other exposed mechanisms.	2	2	2
<b>XML Cryptography</b> . XML cryptography support is an important part of building a secure web service. It is ranked high for the web service scenario and not applicable for the other two scenarios.	Transparent or built-in support for XML-digital signatures and XML-encryption. Portions of XML messages can be selectively encrypted and signed.	0	3	0

Scenario weights range from 1 (relatively unimportant) to 3 (extremely important). A scenario weight of zero means that the topic does not apply.

## Findings

The table below summarizes @stake's quantitative findings. For each analysis topic, the two high-level metrics (Best Practice Compliance and Ease of Securing) are shown. These are the arithmetic means of the underlying best practice and test case scores. The Ease of Securing metric is further broken down into four sub-metrics.

Topic	Platform	Best Practice Compliance	Ease of Securing				
			Overall	Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Cryptographic Hashing <i>1 best practice, 2 test cases</i>	.NET Windows	3	3.75	3	5	3	4
	WebSphere Linux	3	3.75	3	5	3	4
	WebSphere Unix	3	3.75	3	5	3	4

Topic	Platform	Best Practice Compliance	Overall	Ease of Securing			
				Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Encryption Algorithms <i>1 best practice, 3 test cases</i>	.NET Windows	<b>3</b>	<b>3.5</b>	3	4.33	4	2.67
	WebSphere Linux	<b>3</b>	<b>3.5</b>	3	5	3	3
	WebSphere Unix	<b>3</b>	<b>3.5</b>	3	5	3	3
Key Generation <i>1 best practice, 1 test case</i>	.NET Windows	<b>5</b>	<b>4</b>	5	3	4	4
	WebSphere Linux	<b>3</b>	<b>3.75</b>	3	5	3	4
	WebSphere Unix	<b>3</b>	<b>3.75</b>	3	5	3	4
Random Number Generation <i>1 best practice, 1 test case</i>	.NET Windows	<b>3</b>	<b>3.5</b>	3	4	3	4
	WebSphere Linux	<b>3</b>	<b>3.5</b>	3	5	2	4
	WebSphere Unix	<b>3</b>	<b>3.5</b>	3	5	2	4
Secrets Storage <i>1 best practice, 1 test case</i>	.NET Windows	<b>2</b>	<b>2.25</b>	2	2	4	1
	WebSphere Linux	<b>3</b>	<b>3.5</b>	3	4	4	3
	WebSphere Unix	<b>3</b>	<b>3.5</b>	3	4	4	3
XML Cryptography <i>1 best practice, 2 test cases</i>	.NET Windows	<b>3</b>	<b>3.5</b>	3	4	4	3
	WebSphere Linux	<b>3</b>	<b>2.5</b>	3	2	4	1
	WebSphere Unix	<b>3</b>	<b>2.5</b>	3	2	4	1

This area of analysis was comprised of ten (10) test cases covering cryptographic functionality required for the secure operation of web based applications and web services. Both vendors scored similarly across the board with one exception. Microsoft did not score as well on areas of key material storage. This was due in part to incomplete documentation and the large amount of effort required to implement the stated best practices.

The individual sections for each analysis topic describe @stake's findings in further detail.

## 4.2.4.1 Cryptographic Hashing

---

### Cryptographic Hashing

Cryptography enables applications to communicate and send information between other applications, networks, and users in an undecipherable context. Encryption is the most powerful form of cryptography and is pivotal in preventing information leakage, data compromise. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Cryptographic Hashing topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=44]. When hashing is required, platform supports the use of strong, industry standard-algorithms such as SHA1 and MD5.	.NET supports cryptographic hash functions via the crypto API.	<b>Developer extends (3)</b> . Some coding is required to leverage the API.	

**Best Practice Compliance score**, based on mean of 1 best practice: 3

---

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=44.1]. Implement a hash using SHA1.	Implementation Complexity	<b>Small amount of code (3)</b> . The implementation is a straightforward use of the SHA1 CryptoServiceProvider.
	Documentation and Examples	<b>Best practice (5)</b> . Clear cut examples in the documentation provide a clear path towards correct implementation.
	Implementor Competence	<b>Intermediate (3)</b> . The developer is very familiar with cryptographic hashes, but does not have experience with the Microsoft Crypto API.
	Time to Implement	<b>Low to Medium (4)</b> . Quick process requiring a documentation lookup to verify the API format. Coding is limited to just a couple lines.
[id=44.2]. Implement a hash using MD5.	Implementation Complexity	<b>Small amount of code (3)</b> . The implementation is a straightforward use of the MD5 CryptoServiceProvider.
	Documentation and Examples	<b>Best practice (5)</b> . Clear cut examples in the documentation provide a clear path towards correct implementation.
	Implementor Competence	<b>Intermediate (3)</b> . The developer is very familiar with cryptographic hashes, but does not have experience with the Microsoft Crypto API.
	Time to Implement	<b>Low to Medium (4)</b> . Quick process requiring a documentation lookup to verify the API format. Coding is limited to just a couple lines.

**Ease of Securing score**, based on mean of 2 test cases: 3.75

### Notes

#### Test Case 44.1

See hash\_functions.cs for the full source.

```
HashAlgorithm ha;
ha = new SHA1CryptoServiceProvider();
result = ha.ComputeHash(md_input);
```

#### Test Case 44.2

See hash\_functions.cs for the full source.

```
HashAlgorithm ha;
ha = new MD5CryptoServiceProvider();
result = ha.ComputeHash(md_input);
```

### IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=44]. When hashing is required, platform supports the use of strong, industry standard-algorithms such as SHA1 and MD5.	J2EE supports cryptographic hash functions via the crypto API.	<b>Developer extends (3)</b> . Some coding is required to leverage the API.	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 3, Unix 3

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=44.1]. Implement a hash using SHA1.	Implementation Complexity	<b>Small amount of code (3)</b> . The implementation is a straightforward use of the MessageDigest object.
	Documentation and Examples	<b>Best practice (5)</b> . Clear cut examples in the documentation provide a clear path towards correct implementation.
	Implementor Competence	<b>Intermediate (3)</b> . The developer is very familiar with cryptographic hashes and the Java implementation.
	Time to Implement	<b>Low to Medium (4)</b> . Quick process requiring a documentation lookup to verify the API format. Coding is limited to just a few lines.
[id=44.2]. Implement a hash using MD5.	Implementation Complexity	<b>Small amount of code (3)</b> . The implementation is a straightforward use of the MessageDigest object.
	Documentation and Examples	<b>Best practice (5)</b> . Clear cut examples in the documentation provide a clear path towards correct implementation.
	Implementor Competence	<b>Intermediate (3)</b> . The developer is very familiar with cryptographic hashes and the Java implementation.
	Time to Implement	<b>Low to Medium (4)</b> . Quick process requiring a documentation lookup to verify the API format. Coding is limited to just a few lines.

**Ease of Securing scores**, based on mean of 2 test cases: Linux 3.75, Unix 3.75

### Notes

#### Test Case 44.1

See hash\_functions.java for the full source.

```
MessageDigest md = MessageDigest.getInstance( "SHA" );
md.update(md_input);
byte[] md_out = md.digest();
```

#### Test Case 44.2

See hash\_functions.java for the full source.

```
MessageDigest md = MessageDigest.getInstance( "MD5" );
md.update(md_input);
byte[] md_out = md.digest();
```

## 4.2.4.2 Encryption Algorithms

---

### Encryption Algorithms

Support for industry standard encryption algorithms is an important feature of an application platform. Without it developers will either attempt to integrate a third-party cryptographic library or even develop their own algorithms. Both of these practices are known to have security pitfalls. Encryption algorithms is weighted as medium for all three application scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Encryption Algorithms topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=45]. When encryption is used, platform supports the use of strong, industry standard-algorithms such as AES, RSA, 3DES, Blowfish, etc. Minimum key size of 128 bits (1024 bits for RSA) is used.	.NET supports cryptographic encryption functions via the crypto API.	<b>Developer extends (3)</b> . Some coding is required to leverage the API.	

**Best Practice Compliance score**, based on mean of 1 best practice: 3

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=45.1]. Encrypt text or a message using AES.	Implementation Complexity	<b>Small amount of code (3)</b> . The implementation is a straightforward use of the AES CryptoServiceProvider.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation provides clear cut examples.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . The developer is very familiar with block cipher encryption, but has no experience with the C# interface to the MSFT Crypto API.
	Time to Implement	<b>Medium (3)</b> . The implementation was a quick process to look up the API and adapt the examples.
[id=45.2]. Encrypt text or a message using 3DES.	Implementation Complexity	<b>( 3 )</b> . The implementation is a straightforward use of the TripleDES CryptoServiceProvider.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation provides clear cut examples.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . The developer is very familiar with block cipher encryption, but has no experience with the C# interface to the MSFT Crypto API.
	Time to Implement	<b>Medium (3)</b> . The implementation was a quick process to look up the API and adapt the examples.
[id=45.3]. Encrypt text or a message using RSA.	Implementation Complexity	<b>Small amount of code (3)</b> . Very little code is required to leverage the RSA CryptoServiceProvider.
	Documentation and Examples	<b>Adequate (3)</b> . The documentation did not adequately demonstrate how to export and import RSA keys using the RSAParameters object. The developer was unable to use the ExportParameters and ImportParameters methods. As a result, the ToXmlString was used to extract and move crypto keys.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . The developer is very familiar with RSA encryption, but has no experience with the C# interface to the MSFT Crypto API.
	Time to Implement	<b>Medium to High (2)</b> . A significant amount of time was spent attempting to use Export/ImportParameters API before resorting to XML key export format.

**Ease of Securing score**, based on mean of 3 test cases: 3.5

### Notes

#### Test Case 45.1

See AES.cs for the full source.

#### Test Case 45.2

See TripleDES.cs for the full source.

### Test Case 45.3

See RSAEncryption.cs for the full source.

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=45]. When encryption is used, platform supports the use of strong, industry standard-algorithms such as AES, RSA, 3DES, Blowfish, etc. Minimum key size of 128 bits (1024 bits for RSA) is used.	J2EE supports cryptographic encryption functions via the crypto API.	<b>Developer extends (3)</b> . Some coding is required to leverage the API.	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 3, Unix 3

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=45.1]. Encrypt text or a message using AES.	Implementation Complexity	<b>Small amount of code (3)</b> . The implementation is made straightforward using the cipher object.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation is clear and sample code provides a clean implementation.
	Implementor Competence	<b>Intermediate (3)</b> . The developer is very familiar with the Java cipher API.
	Time to Implement	<b>Medium (3)</b> . The implementation required just a few minutes to look up the API and adapt the sample code.
[id=45.2]. Encrypt text or a message using 3DES.	Implementation Complexity	<b>Small amount of code (3)</b> . The implementation is made straightforward using the cipher object.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation is clear and sample code provides a clean implementation.
	Implementor Competence	<b>Intermediate (3)</b> . The developer is very familiar with the Java cipher API.
	Time to Implement	<b>Medium (3)</b> . The implementation required just a few minutes to look up the API and adapt the sample code.

### Level of Effort Analysis

---

Test Case	Criteria	Rating
[id=45.3]. Encrypt text or a message using RSA.	Implementation Complexity	<b>Small amount of code (3)</b> . The implementation is made straightforward using the cipher object.
	Documentation and Examples	<b>Best practice (5)</b> . Documentation is clear and sample code provides a clean implementation.
	Implementor Competence	<b>Intermediate (3)</b> . The developer is very familiar with the Java cipher API.
	Time to Implement	<b>Medium (3)</b> . The implementation required just a few minutes to look up the API and evaluate the sample code.

**Ease of Securing scores**, based on mean of 3 test cases: Linux 3.5, Unix 3.5

---

### Notes

#### Test Case 45.1

See encryption\_functions.java for the full source.

#### Test Case 45.2

See encryption\_functions.java for the full source.

#### Test Case 45.3

See SampleRSACrypto.java for the full source.

#### 4.2.4.3 Key Generation

---

### Key Generation

Secure key generation is an important feature of a secure application platform. Without it developers will either attempt to integrate a third-party cryptographic library or even develop their own algorithms. Key generation is weighted medium for all scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Key Generation topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=47]. Key generation uses a quality PRNG. Private portions of key pairs are generated within a secure container and are never exposed to user level processes.	.NET supports secure cryptographic key generation via the crypto API.	<b>Transparent (5)</b> . The keys are generated automatically when the RSACryptoServiceProvider object is instantiated.	The documentation should be enhanced to include discussion of how the key generation is performed, including what RNG provider is used.

**Best Practice Compliance score**, based on mean of 1 best practice: 5

---

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=47.1]. Generate a key pair.	Implementation Complexity	<b>Wizard (5)</b> . The keys are generated automatically when the object is instantiated.
	Documentation and Examples	<b>Adequate (3)</b> . The documentation was not very clear regarding how the RSA keys are generated or what RNG source is used. The developer was unable to successfully export the keys using the ExportParameters method.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer is experienced with RSA and public key cryptography, but has no experience with the C# interface to the Microsoft crypto API.
	Time to Implement	<b>Low to Medium (4)</b> . The developer spent a few minutes reading the documentation in an attempt to understand how the keys are generated when the object is instantiated.

**Ease of Securing score**, based on mean of 1 test case: 4

### Notes

#### Test Case 47.1

See RSAEncryption.cs for the full source.

```
RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();
```

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=47]. Key generation uses a quality PRNG. Private portions of key pairs are generated within a secure container and are never exposed to user level processes.	J2EE supports secure cryptographic key generation via the crypto API.	<b>Developer extends (3)</b> . A small amount of code is required to generate key material.	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 3, Unix 3

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=47.1]. Generate a key pair.	Implementation Complexity	<b>Small amount of code (3)</b> . The implementation is a straightforward use of the API.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation is accurate and complete
	Implementor Competence	<b>Intermediate (3)</b> . The developer is very familiar with the Java cipher API
	Time to Implement	<b>Low to Medium (4)</b> . The developer required just a few minutes to look up the API and evaluate the sample code.

**Ease of Securing scores**, based on mean of 1 test case: Linux 3.75, Unix 3.75

### Notes

#### Test Case 47.1

See SampleRSACrypto.java for the full source.

```
rsaKeyPairGen = KeyPairGenerator.getInstance( "RSA" , "IBMJCE" );
rsaKeyPairGen.initialize(1024);
rsaKeyPair = rsaKeyPairGen.generateKeyPair();
```

## 4.2.4.4 Random Number Generation

---

### Random Number Generation

Quality random number generation is an important feature of an application platform. Without it, developers will either attempt to integrate a third-party cryptographic library or even develop their own algorithms. Both of these practices are known to have security pitfalls. Random number generation is weighted as medium for all three application scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Random Number Generation topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=46]. Pseudo-random number generators (PRNG) must be based on high-entropy sources and must be statistically random.	.NET supports cryptographic random number generation via the RNGCryptoServiceProvider.	<b>Developer extends (3)</b> . Some coding is required to leverage the API.	The documentation should be enhanced to include details regarding the RNG Crypto Service Provider's underlying implementation, including the entropy collection process.

**Best Practice Compliance score**, based on mean of 1 best practice: 3

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=46.1]. Verify that numbers are statistically random using an industry-accepted statistical test.	Implementation Complexity	<b>Small amount of code (3)</b> . Very easy implementation using the RNG Service Provider.
	Documentation and Examples	<b>Suitable (4)</b> . The documentation regarding use of the RNG Service Provider is very clear. However, there are no details describing the underlying RNG seeding routine or entropy collection.
	Implementor Competence	<b>Intermediate (3)</b> . The developer has extensive experience with cryptographic RNG implementations, but has not previously used the RNG supplied by the Microsoft Crypto Service Provider.
	Time to Implement	<b>Low to Medium (4)</b> . Only one line is required to initialize the PRNG with a second line to generate output.

**Ease of Securing score**, based on mean of 1 test case: 3.5

### Notes

#### Test Case 46.1

See `rng.cs` for the full source.

```
RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider();
rng.GetBytes(random); // The array is now filled with cryptographically strong random
bytes
```

See also: A 97 MB sample of random output was generated and tested using Maurer's Universal Statistical Test (MUST) for cryptographic key generators. (See [http://www.freebsd.org/doc/en\\_US.ISO8859-1/articles/ipsec-must/code.html](http://www.freebsd.org/doc/en_US.ISO8859-1/articles/ipsec-must/code.html) for an implementation.) The test was unable to detect any measurable difference between the Microsoft PRNG output and a theoretically perfect random bit stream.

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=46]. Pseudo-random number generators (PRNG) must be based on high-entropy sources and must be statistically random.	J2EE supports cryptographic random number generation via the <code>java.security.SecureRandom</code> class.	<b>Developer extends (3)</b> . Some coding is required to leverage the API.	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 3, Unix 3

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=46.1]. Verify that numbers are statistically random using an industry-accepted statistical test.	Implementation Complexity	<b>Small amount of code (3)</b> . Very easy implementation using the SecureRandom object
	Documentation and Examples	<b>Best practice (5)</b> . The javadoc documentation clearly explains how to leverage the API. The documentation also discusses internal implementation details of the PRNG seeding function
	Implementor Competence	<b>Expert/intermediate (2)</b> . The developer has extensive experience with cryptographic RNG implementations, including the SecureRandom object.
	Time to Implement	<b>Low to Medium (4)</b> . Only one line is required to initialize the PRNG with a second line to generate output.

**Ease of Securing scores**, based on mean of 1 test case: Linux 3.5, Unix 3.5

### Notes

#### Test Case 46.1

See PRNG.java for the full source.

```
byte[] nextBytes = new byte[20];
try {
    SecureRandom random = SecureRandom.getInstance("SHA1PRNG");
    random.nextBytes(nextBytes);
} catch (NoSuchAlgorithmException nsa) {
    System.out.println(nsa);
}
```

See also: A 20 MB sample of random output was generated and tested using Maurer's Universal Statistical Test (MUST) for cryptographic key generators. (See [http://www.freebsd.org/doc/en\\_US.ISO8859-1/articles/ipsec-must/code.html](http://www.freebsd.org/doc/en_US.ISO8859-1/articles/ipsec-must/code.html) for an implementation.) The test was unable to detect any measurable difference between the J2EE PRNG output and a theoretically perfect random bit stream.

## 4.2.4.5 Secrets Storage

---

### Secrets Storage

Secrets must be managed properly or they lose their value for security. Proper secret storage is weighted equally for all three application scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Secrets Storage topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=48]. Keys and credentials necessary for the application to function are protected from attack. This includes not storing keys in plaintext on the hard drive, in source code, within externally-accessible media, or other exposed mechanisms.	The Data Protection API (DPAPI) supports the ability to protect data using the user password and optional additional entropy. This data may be easily stored on the local machine in the Registry or in files. Additional code is required to expose the DPAPI to C# applications.	<b>Developer implements (2)</b>	The DataProtection class described by <a href="http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dotnet/html/90000000000000000000000000000000.asp">http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dotnet/html/90000000000000000000000000000000.asp</a> should be made accessible as a C# object to improve access to the DPAPI.

**Best Practice Compliance score**, based on mean of 1 best practice: 2

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=48.1]. Store keying material in secure storage location.	Implementation Complexity	<b>Medium amount of code (2)</b> . A non-trivial amount of code must be written to first expose the Win32 API to C# code. Once this code is written, the API is very simple to use. This is an issue of providing built-in support for the DPAPI in C#.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . The developer spent a significant amount of time investigating the best method for protection sensitive data in .NET. The first relevant solution discovered was the Windows Certificate Store, which is poorly supported in .NET. (Read-only access is supported using the Web Services Enhancement.) Once the DPAPI was discovered, confusion was encountered regarding how to actually leverage the API from applications implemented in C#.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has no experience working with Microsoft certificate stores or the DPAPI, but is very familiar with public key certificates and the storage of sensitive data.
	Time to Implement	<b>High (1)</b> . A significant amount of time was required to discover the documentation describing the X509CertificateStore and the DPAPI. Once the appropriate documentation describing the DPAPI was discovered the implementation of the DataProtection class required to leverage the Win32 API was time consuming. However, the code required to actually use the Win32 API is just one line to encrypt and one line to decrypt data.

**Ease of Securing score**, based on mean of 1 test case: 2.25

### Notes

#### Test Case 48.1

Inspired by <http://www.c-sharpcorner.com/Code/2003/Jan/DigitalCertIII.asp>

```

string storename = "MY";
X509CertificateStore.StoreLocation location = X509CertificateStore.LocalMachine;
X509CertificateStore.StoreProvider provider =
X509CertificateStore.StoreProvider.System;
X509CertificateStore store = new X509CertificateStore(provider, location,
storename);
bool fopen = store.OpenRead();

Console.WriteLine("Certificates: " + store.Certificates.Count);

string subjectName = "CN=test";

X509CertificateCollection cers =
store.FindCertificateBySubjectName(subjectName);
Console.WriteLine("Certs with '{0}' subjectname: {1}", subjectName, cers.Count);

//Certs inside cers collection may be removed and operated on individually

```

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=48]. Keys and credentials necessary for the application to function are protected from attack. This includes not storing keys in plaintext on the hard drive, in source code, within externally-accessible media, or other exposed mechanisms.	J2EE supports the secure storage of Public Key Certificates via the Java Key Store.	<b>Developer extends (3)</b> . The Key Store object provides a simple API for manipulating the certificate key store.	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 3, Unix 3

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=48.1]. Store keying material in secure storage location.	Implementation Complexity	<b>Small amount of code (3)</b> . The API provides a simple interface for accessing the Key Store.
	Documentation and Examples	<b>Suitable (4)</b> . The API documentation clearly explains how to use the Key Store object.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . The developer has not worked extensively with public key stores of any type and has no experience working with the Java key store object.
	Time to Implement	<b>Medium (3)</b> . Accessing an existing certificate requires only three lines of code. The majority of time was spent investigating how to use the Java keytool program for generating and manipulating the keystore via the command line.

**Ease of Securing scores**, based on mean of 1 test case: Linux 3.5, Unix 3.5

### Notes

#### Test Case 48.1

See SecureKeyStore.java for the full source.

```
KeyStore keyStore = KeyStore.getInstance("JKS");
keyStore.load(new FileInputStream(".keystore"), keyPass);
key = keyStore.getKey("alias", keyPass);
```

## 4.2.4.6 XML Cryptography

---

### XML Cryptography

XML cryptography support is an important part of building a secure web service. It is ranked high for the web service scenario and not applicable for the other two scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 0
- Web Service: 3
- Intranet: 0

The sections that follow contain the results of @stake's analysis of the XML Cryptography topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=43]. Transparent or built-in support for XML-digital signatures and XML-encryption. Portions of XML messages can be selectively encrypted and signed.	.NET supports the encryption and digital signing of data via SOAP input and output filters.	<b>Developer extends (3)</b> .	

**Best Practice Compliance score**, based on mean of 1 best practice: 3

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=43.1]. Implement a signed XML message fragment.	Implementation Complexity	<b>Small amount of code (3)</b> . The implementation requires configuring security elements within the SoapContext object to indicate the certificate to be used for signing and the signature elements to be created. This information is used by input and output filters to determine which fields require digital signing/validation and what key material is required to perform the operation.
	Documentation and Examples	<b>Suitable (4)</b> . The documentation located at <a href="http://msdn.microsoft.com/webservices/building/wse/default.aspx?pull=/library/en-us/dnwssecur/html/wssecauthwse.asp">http://msdn.microsoft.com/webservices/building/wse/default.aspx?pull=/library/en-us/dnwssecur/html/wssecauthwse.asp</a> provides a very clear example of how to implement digital signing and signature validation of XML messages using the SOAP filters.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has no experience working with XML, SOAP, or Web Services. The developer has significant experience with digital signatures.
	Time to Implement	<b>Medium (3)</b> . Development time was limited to searching for documentation and extending sample code.
[id=43.2]. Implement an encrypted XML message fragment.	Implementation Complexity	<b>Small amount of code (3)</b> . The implementation requires configuring security elements within the SoapContext object to indicate the encryption algorithm and keying material. This information is used by input and output filters to determine which fields require encryption/decryption and what key material is required to perform the operation.
	Documentation and Examples	<b>Suitable (4)</b> . The documentation located at <a href="http://msdn.microsoft.com/webservices/building/wse/default.aspx?pull=/library/en-us/dnwebsrv/html/wseencryption.asp">http://msdn.microsoft.com/webservices/building/wse/default.aspx?pull=/library/en-us/dnwebsrv/html/wseencryption.asp</a> provides a very clear example of how to implement XML encryption using the SOAP filters.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has no experience working with XML, SOAP, or Web Services. The developer has significant experience with data encryption.
	Time to Implement	<b>Medium (3)</b> . Development time was limited to searching for documentation and extending sample code.

**Ease of Securing score**, based on mean of 2 test cases: 3.5

## Notes

### Test Case 43.1

See the example code in

<http://msdn.microsoft.com/webservices/building/wse/default.aspx?pull=/library/en-us/dnwssecur/html/wssecauthwse.asp> for an illustration of how to perform digital signing of XML.

### Test Case 43.2

See the example code in

<http://msdn.microsoft.com/webservices/building/wse/default.aspx?pull=/library/en-us/dnwebsrv/html/wseencryption.asp> for an illustration of how to perform encryption of XML data.

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=43]. Transparent or built-in support for XML-digital signatures and XML-encryption. Portions of XML messages can be selectively encrypted and signed.	The XML Security Suite offered by IBM supports digital signatures and data encryption of XML tagged data. This package is available for download from <a href="http://www.alphaworks.ibm.com/tech/xmlsecuritysuite">http://www.alphaworks.ibm.com/tech/xmlsecuritysuite</a>	<b>Developer extends (3)</b> . The API supported by the IBM XML Security Suite provides the functionality necessary to perform digital signatures and data encryption of XML tagged data.	The documentation is limited to javadoc API information and a few HTML files describing sample applications demonstrating the use of the API. The API javadoc is very poor, giving little description of the underlying implementation or proper use of the API. This documentation should be improved to the standards of other Java APIs.

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 3, Unix 3

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=43.1]. Implement a signed XML message fragment.	Implementation Complexity	<b>Small amount of code (3)</b> . A small amount of code is required to leverage the XML Security Suite API to perform digital signing and digital signature verification of XML data.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . The documentation is limited to javadoc API information and a few HTML files describing sample applications demonstrating the use of the API. The API javadoc is very poor, giving little description of the underlying implementation or proper use of the API.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has no experience working with XML, but does have significant experience working with digital signatures.
	Time to Implement	<b>High (1)</b> . The developer spent a significant amount of time trying to understand the javadoc API and reading the documentation associated with the sample code. Difficulties were encountered in making the sample applications function correctly. The poor javadoc style and lack of detailed documentation associated with the samples greatly increased the difficulty in using the XML Security Suite API.

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=43.2]. Implement an encrypted XML message fragment.	Implementation Complexity	<b>Small amount of code (3)</b> . A small amount of code is required to leverage the XML Security Suite API to perform data encryption and decryption of XML data.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . The documentation is limited to javadoc API information and a few HTML files describing sample applications demonstrating the use of the API. The API javadoc is very poor, giving little description of the underlying implementation or proper use of the API.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has no experience working with XML, but does have significant experience working with data encryption.
	Time to Implement	<b>High (1)</b> . The developer spent a significant amount of time trying to understand the javadoc API and reading the documentation associated with the sample code. Difficulties were encountered in making the sample applications function correctly. The poor javadoc style and lack of detailed documentation associated with the samples greatly increased the difficulty in using the XML Security Suite API.

**Ease of Securing scores**, based on mean of 2 test cases: Linux 2.5, Unix 2.5

### Notes

#### Test Case 43.1

See XMLSig.java for the full source.

#### Test Case 43.2

See DOMCipher.java for the full source.

## 4.2.5 Database Access

---

### Database Access

Database access limits the connectivity by which applications access the database. Good database access control curbs entry to a datasource to prevent information theft, modification, addition, and data deletion.

The Database Access area of analysis includes these topics:

- Database Pool Connection Encryption
- Data Query Safety

The table below describes the topics at a high level, and lists prevailing best practices. The right-hand columns show the weights @stake assigned for each of the scenarios.

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Database Pool Connection Encryption</b> . Encryption of the database connection ensures the privacy of the data accessed by the application. This is more important in external applications which may be hosted at an Internet provider. External application scenarios are weighted medium and the intranet scenario low.	Applications that use database connection pools to access remote resources should do so over an encrypted channel.	2	2	1
<b>Data Query Safety</b> . Querying the database with stored procedures or prepared statements can help prevent SQL poisoning attacks. It is weighted medium for all three application scenarios.	Stored procedures and/or prepared statements are the only mechanism used by the application to access the database. Dynamic queries that accept user-submitted parameters are not used.	2	2	2

Scenario weights range from 1 (relatively unimportant) to 3 (extremely important). A scenario weight of zero means that the topic does not apply.

### Findings

The table below summarizes @stake's quantitative findings. For each analysis topic, the two high-level metrics (Best Practice Compliance and Ease of Securing) are shown. These are the arithmetic means of the underlying best practice and test case scores. The Ease of Securing metric is further broken down into four sub-metrics.

Topic	Platform	Best Practice Compliance	Overall	Ease of Securing			
				Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Database Pool Connection Encryption <i>1 best practice, 1 test case</i>	.NET Windows	<b>3</b>	<b>3.25</b>	3	4	4	2
	WebSphere Linux	<b>2</b>	<b>2</b>	1	2	4	1
	WebSphere Unix	<b>2</b>	<b>2</b>	1	2	4	1
Data Query Safety <i>1 best practice, 2 test cases</i>	.NET Windows	<b>3</b>	<b>4.13</b>	4.5	4	4	4
	WebSphere Linux	<b>5</b>	<b>4.13</b>	5	2.5	4	5
	WebSphere Unix	<b>5</b>	<b>4.13</b>	5	2.5	4	5

This area of the analysis was comprised of three (3) test cases designed to evaluate the platform's ability to create, execute, and maintain secure database connections. Microsoft was found to score higher under all test cases. Native support for encrypted database connections, as well as superior documentation and code samples were responsible for the disparity in scores in this area of analysis.

The individual sections for each analysis topic describe @stake's findings in further detail.

## 4.2.5.1 Database Pool Connection Encryption

---

### Database Pool Connection Encryption

Encryption of the database connection ensures the privacy of the data accessed by the application. This is more important in external applications which may be hosted at an Internet provider. External application scenarios are weighted medium and the intranet scenario low. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 1

The sections that follow contain the results of @stake's analysis of the Database Pool Connection Encryption topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### **Best Practice Compliance Analysis**

Best Practice	Options	Rating	Recommendation
[id=63]. Applications that use database connection pools to access remote resources should do so over an encrypted channel.	SQL Server supports SSL encrypted connections. The database server can be configured to force all connections to be encrypted or to allow clients the option of SSL-encrypting their connections.	<b>Developer extends (3).</b> SSL-encrypted database connections require simple configuration steps to enable.	

**Best Practice Compliance score**, based on mean of 1 best practice: 3

### Level of Effort Analysis

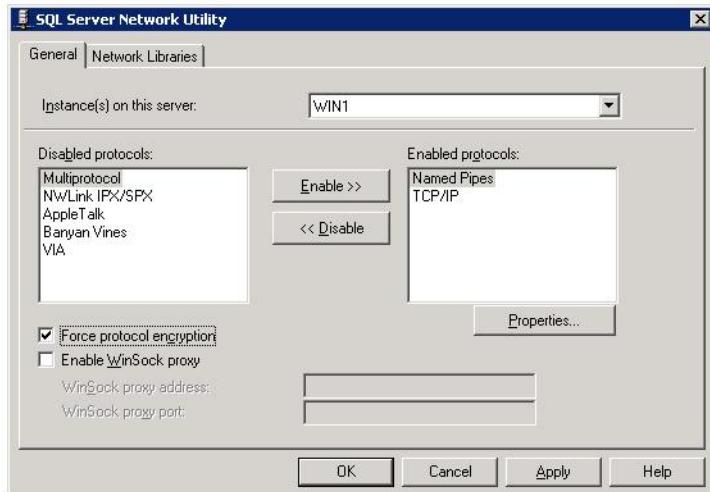
Test Case	Criteria	Rating
[id=63.1]. Implement an encrypted channel for database pooling.	Implementation Complexity	<b>Small amount of code (3)</b> . Supporting SSL encryption mainly consists of importing the appropriate certificates on client and server systems and enabling the encrypted connections.
	Documentation and Examples	<b>Suitable (4)</b> . The documentation clearly explains how to implement secure connections.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering IIS and MS SQL Server.
	Time to Implement	<b>Medium to High (2)</b> . While fairly straightforward, enabling SSL requires a number of steps to be performed.

**Ease of Securing score**, based on mean of 1 test case: 3.25

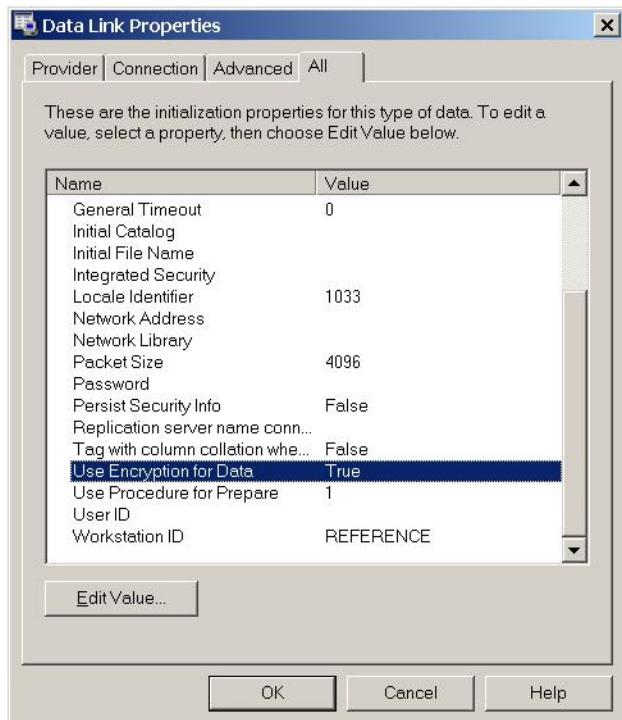
### Notes

#### Test Case 63.1

Configuring SQL Server to only accept encrypted connections is performed through the Server Network Utility:



If the database server doesn't force all connections to be encrypted, clients can still secure their communications through the Data Link Properties:



Finally, IPSec can also be used to encrypt communications between servers, as the second link below describes.

See also: How To: Use SSL to Secure Communication with SQL Server 2000 and How To: Use IPSec to Provide Secure Communication Between Two Servers .

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=63]. Applications that use database connection pools to access remote resources should do so over an encrypted channel.	WebSphere and DB2 do not support creating encrypted JDBC connections. Alternatives include using stunnel or IPSec.	<b>Developer implements (2)</b> . Significant effort needs to be expended in order to encrypt communications between the application and database servers.	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 2, Unix 2

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=63.1]. Implement an encrypted channel for database pooling.	Implementation Complexity	<b>Large amount of code (1)</b> . Configuring IPSec or stunnel on the application and database servers is a fairly lengthy process.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Very little documentation could be found on this topic.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering WebSphere application servers.
	Time to Implement	<b>High (1)</b> . Much time was spent searching IBM, the Unix vendor's web site, and other web sites for relevant documentation.

**Ease of Securing scores**, based on mean of 1 test case: Linux 2, Unix 2

### Notes

#### Test Case 63.1

While DB2 does not appear to support encrypted connections, it does of course support encrypted authentication credentials. IPSec or stunnel (which creates an SSL connection between two endpoints) could be used to secure data flowing between application and database servers.

## 4.2.5.2 Data Query Safety

---

### Data Query Safety

Querying the database with stored procedures or prepared statements can help prevent SQL poisoning attacks. It is weighted medium for all three application scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Data Query Safety topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=62]. Stored procedures and/or prepared statements are the only mechanism used by the application to access the database. Dynamic queries that accept user-submitted parameters are not used.		<b>Developer extends (3)</b> . The Web Data Form Wizard facilitates greatly building a web form bound to a database. Unfortunately, even though ADO.NET provides for stored procedure usage, one has to implement its usage manually.	No recommendation.

**Best Practice Compliance score**, based on mean of 1 best practice: 3

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=62.1]. Implement a database access class that uses a stored procedure or prepared statement.	Implementation Complexity	<b>Wizard+ (4)</b> . The test case implementation requires some manual steps in addition to using the wizard.
	Documentation and Examples	<b>Adequate (3)</b> . Very limited documentation is available on how to appropriately use stored procedures with HTML controls.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive web application development experience but very little with ASP and ASP.NET.
	Time to Implement	<b>Medium (3)</b> . The Web Data Form Wizard is very easy to use but stored procedures need to be manually linked to form elements.
[id=62.2]. Verify that SQL poisoning attacks are ineffective.	Implementation Complexity	<b>Wizard (5)</b> . Using stored procedures and/or parameterize queries prevents SQL injection.
	Documentation and Examples	<b>Best practice (5)</b> . In addition to documentation on writing stored procedures and parameterize statements, the security advantages and implications of both are also covered.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive web application development experience but very little with ASP and ASP.NET.
	Time to Implement	<b>Low (5)</b> . Testing for SQL injection

**Ease of Securing score**, based on mean of 2 test cases: 4.13

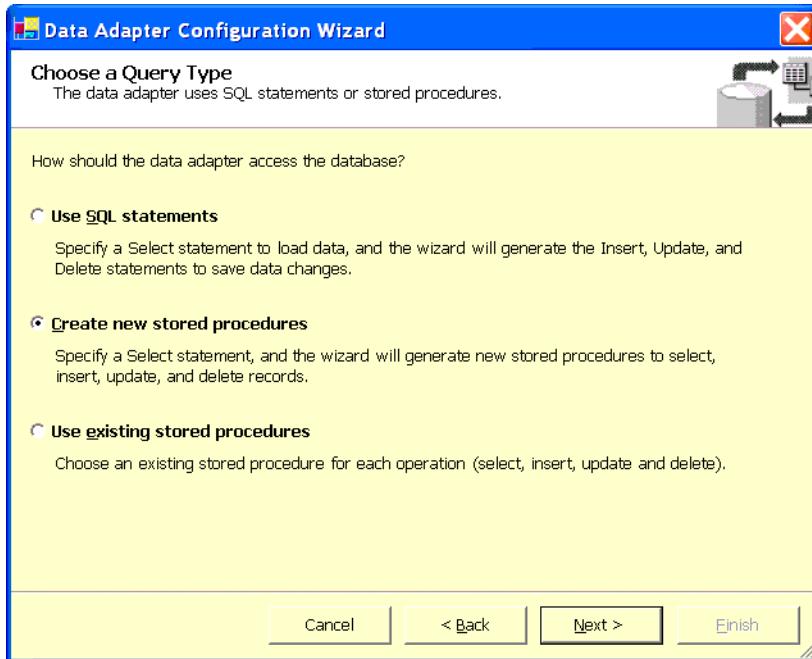
### Notes

#### Test Case 62.1

All queries generated by the data form wizard are dynamic SQL statements potentially susceptible to SQL injection attacks.

Nowhere did @stake find documentation stating unequivocally that one should do stringent input validation and use stored procedures to its fullest extent in order to build defense in depth against security vulnerabilities.

Generating a web form using a stored procedure is relatively straightforward. By drag and dropping a new SQLDataAdapter, this will start a wizard where one can choose to create new stored procedures or use existing ones.



See also: .NET Data Access Architecture Guide

.NET Reflection - Dynamically Bind Your Data Layer to Stored Procedures and SQL Commands Using .NET Metadata and Reflection

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=62]. Stored procedures and/or prepared statements are the only mechanism used by the application to access the database. Dynamic queries that accept user-submitted parameters are not used.	WebSphere offers both Java prepared statements and stored procedures via JDBC. It also provides "DB beans," which are a type of generated JDBC wrapper.	<b>Transparent (5)</b> . DB beans hide SQL details from calling classes and protect against SQL injection attacks. Developers can quickly and easily generate lightweight Java objects that implement parameterized prepared statements.	IBM should provide more documentation for developers describing data query safety issues, particularly with respect to connection string storage. The documentation should explain how WebSphere's DB bean implementation protect against SQL injection attacks.

Best Practice Compliance scores, based on mean of 1 best practice: Linux 5, Unix 5

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=62.1]. Implement a database access class that uses a stored procedure or prepared statement.	Implementation Complexity	<b>Wizard (5)</b> . Database connections and query prototyping are done through wizard interfaces or direct SQL. DB beans are generated directly from database queries through a simple wizard.
	Documentation and Examples	<b>Adequate (3)</b> . Although the functional documentation for using the various database helpers and utilities was quite good, there is only limited information on data query safety.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive web application development experience, but little with WebSphere Application Server directly.
	Time to Implement	<b>Low (5)</b> . Prototyping a parameterized SQL query and generating an associated DB bean that implemented a prepared statement took less than five minutes.
[id=62.2]. Verify that SQL poisoning attacks are ineffective.	Implementation Complexity	<b>Wizard (5)</b> . As described above, WebSphere Studio's DB bean implementation provides transparent protection against SQL injection.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Neither WebSphere Studio concerning the risks of and possible solutions to SQL injection.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive web application development experience but little with WebSphere Application Server directly.
	Time to Implement	<b>Low (5)</b> . An attempt to inject SQL was unsuccessful.

**Ease of Securing scores**, based on mean of 2 test cases: Linux 4.13, Unix 4.13

### Notes

#### Test Case 62.1

The documentation provides extremely limited guidance on the topic of data query safety, although it does encourage the use of stored procedures instead of prepared statements. Here is the one and only mention of data query safety anywhere in WebSphere Studio's online help:

By including database privileges with stored procedures that use static SQL, the database administrator (DBA) can improve security. The DBA or developer who builds the stored procedure must have the database privileges that the stored procedure requires. Users of the client applications that call the stored procedure do not need such privileges. This can reduce the number of users who require privileges.

Fortunately, the SQL utilities that are provided in WebSphere Studio render the need for documentation largely irrelevant since the generated classes provide a high degree of safety. Studio encourages developers to develop database access classes as follows:

1. Create a connection to the database, then create schema and tables as needed
2. Create a *SQL Statement* or *SQL Stored Procedure* (via a GUI)

3. Once the desired query is prototyped and working, create a *DB bean*
4. Use the generated DB bean in the application

The SQL Statement wizard (**New... Data... SQL Statement**) is well-designed, and works similarly to many popular GUI query builders. Microsoft Access users, for instance, will feel right at home. A more direct approach using straight SQL is also available, and will likely be preferred by most developers. The example given in the online help provides helpful guidance on how to use parameters to narrow down the query.

Once the SELECT statement is completed, creating the DB bean is straightforward: right-click and select **Create Java bean**. The user is given a choice as to how database credentials are supplied: as hardcoded strings in the implementation classes (a bad idea, in our view), as parameters during class instantiation, or from an existing web or EJB tier JNDI data source:

**Create a Java Bean that executes an SQL Select statement**

**Specify Runtime Database Connection Information**

Enter information for establishing a database connection at runtime

Use Data Source Connection  
 Data source/JNDI name:

Use Driver Manager Connection  
 Driver name:   
 URL:

How will user authentication be provided?  
 By the execute() method's caller  
 Inside the execute() method

User ID:   
 Password:   
 Re-enter Password:

[< Back](#) [Next >](#) [Finish](#) [Cancel](#)

Either of the latter two options are preferred, since they do not bury connection strings in class files — where they could be easily retrieved with a decompiler.

Once the user completes the wizard, it generates two implementation classes for the statement: a wrapper bean class for instantiating the query, and a class that wraps the resulting row set. Here is an the main wrapper

class:

```

package com.atstake.bestpractices;
import java.sql.*;
import com.ibm.db.beans.*;

public class SelectCustomerRow {
    private int rowNumber;
    private DBSelect select;

    /**
     * Constructs an object that represents a row from a DBSelect.
     */
    public SelectCustomerRow(DBSelect aRef, int aRowNumber) {
        select = aRef;
        rowNumber = aRowNumber;
    }

    /**
     * Returns the value of column CUSTOMER_ID in the row represented by this object.
     */
    public Object getCUSTOMER_ID() throws SQLException {
        return select.getCacheValueAt(rowNumber, 1);
    }

    /**
     * Returns the value of column CUSTOMER_NAME in the row represented by this object.
     */
    public Object getCUSTOMER_NAME() throws SQLException {
        return select.getCacheValueAt(rowNumber, 2);
    }

    /**
     * Returns a String that contains all of the values in the row represented by this
     * object.
     */
    public String toString() {
        String string = "";
        try {
            for (int i = 1; i <= select.getColumnCount(); i++) {
                string += select.getCacheValueAt(rowNumber, i);
                string += " ";
            }
        } catch (SQLException ex) {
            return null;
        }
        return string;
    }
}

```

As shown in the code sample, the result columns (CUSTOMER\_ID and CUSTOMER\_NAME) are wrapped in native Java types. This is excellent.

The rowset wrapper class prepares the SQL statement and executes the query. Here is an excerpt of the

relevant methods from the class:

```
package com.atstake.bestpractices;
import java.sql.*;
import com.ibm.db.beans.*;

public class SelectCustomer {
    private DBSelect select;
    ...
    /**
     * Creates a DBSelect instance and initializes its properties.
     */
    protected void initializer() {
        select = new DBSelect();
        try {
            select.setDriverName("COM.ibm.db2.jdbc.net.DB2Driver");
            select.setUrl("jdbc:db2://172.120.32.12:6789/ernie");
            select.setCommand(
                "SELECT BESTPRACTICESCHEMA.CUSTOMER.ID, BESTPRACTICESCHEMA.CUSTOMER.NAME,
                 FROM BESTPRACTICESCHEMA.CUSTOMER WHERE BESTPRACTICESCHEMA.CUSTOMER.NAME
                 = :name ORDER BY ID ASC, NAME ASC, PASSWORD ASC");
            DBParameterMetaData parmMetaData = select.getParameterMetaData();
            parmMetaData.setParameter(
                1,
                "name",
                java.sql.DatabaseMetaData.procedureColumnIn,
                java.sql.Types.VARCHAR,
                String.class);
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}

/**
 * Executes the SQL statement.
 */
public void execute(
    String userid,
    String password,
    int maxRows,
    String name)
throws SQLException {
    try {
        select.setUsername(userid);
        select.setPassword(password);
        select.setMaxRows(maxRows);
        select.setParameter("name", name);
        select.execute();
    }

    // Free resources of select object.
    finally {
        select.close();
    }
}
...
}
```

```
}
```

For this example, we used a SQL query that was, in essence, `SELECT ID, USERNAME FROM CUSTOMER WHERE NAME=:name`. As shown in the generated rowset implementation class above, the parameter used for the SQL statement (`:name`) is clearly identified and instantiated as such by the implementation class.

Here is sample code that uses the DB bean, as might be found in a servlet or JSP:

```
SelectCustomer customer = new SelectCustomer();
customer.initializer();
try {
    customer.execute(username, password, 255, "ajaquith");
    SelectCustomerRow[] rows = customer.getRows();
    for (int i=0; i > rows.length; i++) {
        SelectCustomerRow row = rows[i];
        System.out.println("ID: " + row.getCUSTOMER_ID()
            + ", NAME=" + row.getCUSTOMER_NAME());
    }
}
catch (Exception e) {
    System.out.println(e.getMessage());
}
```

Note the supplied username and password, which (we admit) is a potential point of vulnerability. These could be stored, ideally, in encrypted form as environment variables in the web application's `web.xml` file; a helper class would decrypt these at startup. IBM does not suggest any methods for doing this.

### Test Case 62.2

See the notes for the previous test case.

## 4.2.6 Data Validation

---

### Data Validation

Data validation ensures the integrity of information used by an application. Generally data validation is used for data input and output to ensure proper information is being passed to and from the application to prevent attack surface exposer and the disclosure of sensitive information. For web services it is very important that data be validated properly since their interface specification is published as WSDL which can aid attackers in generating hostile input. Web services are weighted high. Web application and intranet scenarios are weighted medium.

The Data Validation area of analysis includes these topics:

- Common Validators
- Data Sanitization
- Negative Data Validation
- Output Filtering
- Positive Data Validation
- Type Checking

The table below describes the topics at a high level, and lists prevailing best practices. The right-hand columns show the weights @stake assigned for each of the scenarios.

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Common Validators</b> . Common validators help a developer build proper input filtering for an application. It is rated low for all scenarios.	Custom validators should be extensible to enable centralized management, consistency, and re-use. Validators should be re-used by the application's web tier, application tier, and web service tier.	1	1	1
<b>Data Sanitization</b> . It is important that an application be able to easily sanitize its input. This feature is weighted medium for all scenarios.	If user input accepts HTML or XML markup, meta characters and the characters used for HTML should be escaped.	2	2	2

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Negative Data Validation</b> . Negative data validation is a second layer of defense against malicious input if positive data validation fails. For web services it is very important that data be validated properly since their interface specification is published as WSDL which can aid attackers in generating hostile input. Web services are weighted high. Web application and intranet scenarios are weighted medium.	Known bad input such as malformed parameters, strings of excessive length, and dynamic query strings should be rejected. It is best to accept only known good data, but this is not always possible.	2	3	2
<b>Output Filtering</b> . Output filtering helps a developer build an application which is not susceptible to cross site scripting attacks. It is rated low for all scenarios.	When information is displayed to users, it should be escaped. HTML should be rendered inactive to prevent cross site scripting attacks.	1	1	1
<b>Positive Data Validation</b> .	User-entered input is verified before being accepted as valid. Unconstrained and unchecked data can lead to serious application compromises. Ideally, the application should only accept meaningful input.	2	3	2
<b>Type Checking</b> . Type checking helps a developer build proper input validation. It is not a requirement for proper input validation. It is rated low for all scenarios.	User-entered character data should only allow input valid for the underlying object's field type. Validation of input fields includes checking for known data types such as integer or date, disallowing input of other types.	1	1	1

Scenario weights range from 1 (relatively unimportant) to 3 (extremely important). A scenario weight of zero means that the topic does not apply.

## Findings

The table below summarizes @stake's quantitative findings. For each analysis topic, the two high-level metrics (Best Practice Compliance and Ease of Securing) are shown. These are the arithmetic means of the underlying best practice and test case scores. The Ease of Securing metric is further broken down into four sub-metrics.

Topic	Platform	Ease of Securing					
		Best Practice Compliance	Overall	Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Common Validators <i>1 best practice, 1 test case</i>	.NET Windows	3	3.25	3	3	4	3
	WebSphere Linux	5	3.5	3	2	5	4
	WebSphere Unix	5	3.5	3	2	5	4

Topic	Platform	Best Practice Compliance	Ease of Securing				
			Overall	Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Data Sanitization <i>1 best practice, 1 test case</i>	.NET Windows	<b>3</b>	<b>4</b>	3	5	4	4
	WebSphere Linux	<b>4</b>	<b>3.25</b>	4	2	5	2
	WebSphere Unix	<b>4</b>	<b>3.25</b>	4	2	5	2
Negative Data Validation <i>1 best practice, 1 test case</i>	.NET Windows	<b>3</b>	<b>2.75</b>	3	2	4	2
	WebSphere Linux	<b>4</b>	<b>3.75</b>	4	2	5	4
	WebSphere Unix	<b>4</b>	<b>3.75</b>	4	2	5	4
Output Filtering <i>1 best practice, 1 test case</i>	.NET Windows	<b>4</b>	<b>3.75</b>	3	3	4	5
	WebSphere Linux	<b>5</b>	<b>4</b>	5	4	2	5
	WebSphere Unix	<b>5</b>	<b>4</b>	5	4	2	5
Positive Data Validation <i>1 best practice, 1 test case</i>	.NET Windows	<b>4</b>	<b>4.5</b>	4	5	4	5
	WebSphere Linux	<b>3</b>	<b>2.5</b>	2	2	5	1
	WebSphere Unix	<b>3</b>	<b>2.5</b>	2	2	5	1
Type Checking <i>1 best practice, 1 test case</i>	.NET Windows	<b>5</b>	<b>4.25</b>	4	5	4	4
	WebSphere Linux	<b>4</b>	<b>3.25</b>	3	2	5	3
	WebSphere Unix	<b>4</b>	<b>3.25</b>	3	2	5	3

The Data validation portion of the analysis was comprised six (6) test cases covering a wide variety of input validation tasks. Overall Microsoft scored slightly higher in areas of Data Sanitization, Positive Data Validation, and Type Checking. IBM was found to excel at areas where the use of the third-party Struts Validator Framework was employed. These included: Common Validators, Data Sanitization and Negative Data Validation. Once again, Microsoft documentation and sample code was found to be of superior quality. In certain areas, IBM documentation for the same best practices was found to be lacking.

The individual sections for each analysis topic describe @stake's findings in further detail.

## 4.2.6.1 Common Validators

---

### Common Validators

Common validators help a developer build proper input filtering for an application. It is rated low for all scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 1
- Web Service: 1
- Intranet: 1

The sections that follow contain the results of @stake's analysis of the Common Validators topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=39]. Custom validators should be extensible to enable centralized management, consistency, and re-use. Validators should be re-used by the application's web tier, application tier, and web service tier.	The .NET Framework provides for extensible common validators for web forms, incorporating both a client side and server side implementation.	<b>Developer extends (3)</b> . Though pre-built validators are available, they cannot be used for web services forcing developers to code.	

**Best Practice Compliance score**, based on mean of 1 best practice: 3

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=39.1]. Create classes that implement re-usable input validators.	Implementation Complexity	<b>Small amount of code (3)</b> . Creating a new validator class is relatively simple and can inherit from an validator object. Implementing a validator for web services needs to be designed from scratch.
	Documentation and Examples	<b>Adequate (3)</b> . No documentation available on extending validator controls nor on a validation framework for web services.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive web application development experience but very little with ASP and ASP.NET.
	Time to Implement	<b>Medium (3)</b> . Writing the validators is a relatively quick task.

**Ease of Securing score**, based on mean of 1 test case: 3.25

### Notes

#### Test Case 39.1

Creating a new validator component involves the following code:

```

Imports System
Imports System.Web.UI

Namespace atStakeCustomControls
    Public Class Class1
        Inherits System.Web.UI.WebControls.BaseValidator

        Protected Overrides Function EvaluateIsValid() As Boolean
            'do evaluation
            If GetControlValidationValue(ControlToValidate) = "@stake" Then
                Return True
            Else
                Return False
            End If
        End Function
    End Class
End Namespace

```

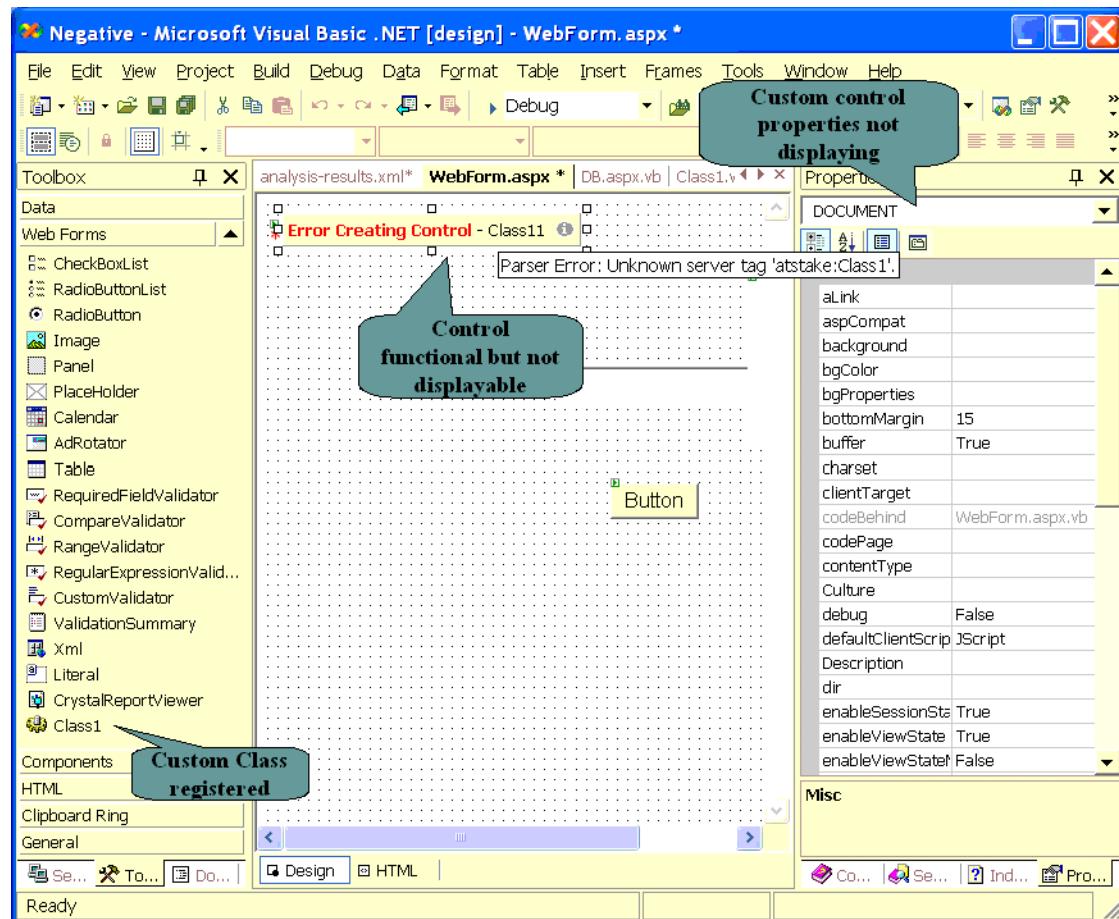
Using it in a web form can be done:

```

[...]
<%@ Register TagPrefix="atstake" Namespace="Negative.atStakeCustomControls" Assembly =
"negative" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
[...]
        <atstake:Class1 id="CustValidator1" runat="server"
ErrorMessage="CustFieldValidator Error Message"
ControlToValidate="TextBox1">*</atstake:Class1>

```

Unfortunately, adding this new validator to the toolbox components although sounding as easy as adding the assembly to the toolbox did not yielded:



The control is nevertheless functional as we verified EvaluateIsValid was correctly executed. We are confident that increased documentation and time would solve these issues.

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=39]. Custom validators should be extensible to enable centralized management, consistency, and re-use. Validators should be re-used by the application's web tier, application tier, and web service tier.	The Struts Validator framework is by design an extensible and re-useable.	<b>Transparent (5)</b> . The Struts Validator, shipped as part of the WebSphere, natively supports the concept of an extendable plug-in architecture for input validation.	

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
<b>Best Practice Compliance scores</b> , based on mean of 1 best practice: Linux 5, Unix 5			

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=39.1]. Create classes that implement re-usable input validators.	Implementation Complexity	<b>Small amount of code (3)</b> . After overcoming the steep learning curve of the Struts package, extending the <code>validator.xml</code> file created for a previous test cases was fairly straight forward.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Vendor supplied documentation on Struts Validators was scant and had a enormous learning curve. The implementer was expected to possess a profound knowledge of the Apache Struts environment and comprehensive understanding of the WebSphere deployment environment. The documentation did not provide information that could lead to insecure designs and deployment environments; its overall uselessness warranted this rating.
	Implementor Competence	<b>Novice (5)</b> . The tester had no prior Struts implementation experience. After six hours with the documentation and the creation of several examples, it was possible to use the functionality to create fairly sophisticated user input filtering routines.
	Time to Implement	<b>Low to Medium (4)</b> . Extending the existing <code>validator.xml</code> file took less than fifteen minutes

**Ease of Securing scores**, based on mean of 1 test case: Linux 3.5, Unix 3.5

## Notes

### Test Case 39.1

A previously created `validator.xml` configuration file is included below. To illustrate the extendable nature of the validator framework.

```

<constant>
  <constant-name>ssn</constant-name>
  <constant-value>^\(?(\d{3})\)?[- ]??(\d{2})\)?[- ] ?(\d{4})$</constant-value>
</constant>
...
<formset
  <form name="register">
    <field property="ssn"
      depends="mask">
      <arg0 key="register.ssn.displayname"/>
      <var>
        <var-name>mask</var-name>
        <var-value>${ssn}</var-value>
      </var>
    </field>
  </form>

```

```
</formset>
```

## 4.2.6.2 Data Sanitization

---

### Data Sanitization

It is important that an application be able to easily sanitize its input. This feature is weighted medium for all scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Data Sanitization topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=36]. If user input accepts HTML or XML markup, meta characters and the characters used for HTML should be escaped.	By default ASP.NET throws a security exception when HTML characters are entered into an input field. If this security restriction is relaxed and HTML characters permitted as input, they can be escaped with the Server.HtmlEncode() method.	Developer extends (3). Escaping HTML input only requires one method call.	

**Best Practice Compliance score**, based on mean of 1 best practice: 3

---

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=36.1]. Implement an input filter that sanitizes HTML/XML markup.	Implementation Complexity	<b>Small amount of code (3)</b> . This filtering is very easy to perform.
	Documentation and Examples	<b>Best practice (5)</b> . Microsoft's documentation, in addition to explanation associated with the security exception, provide ample detail.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . The developer has extensive web application development experience but very little with ASP and ASP.NET.
	Time to Implement	<b>Low to Medium (4)</b> . Creating a sample application and calling the Server.HtmlEncode() was very simple.

**Ease of Securing score**, based on mean of 1 test case: 4

### Notes

#### Test Case 36.1

To test the behavior of Server.HtmlEncode() a simple ASP.NET application was created that consisted of an input field, a button, and a label. When text was entered in the field and the button pressed, the text was filtered for HTML characters and displayed in the label. The method associated with the button was:

```
Private Sub Filter_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Filter.Click
    OutputLabel.Text = Server.HtmlEncode(InputTextBox.Text)
End Sub
```

MSDN does have a very comprehensive article on choosing the most appropriate persistent storage mechanism: Nine Options for Managing Persistent User State in Your ASP.NET Application

Unfortunately, @stake identified a very basic XSS flaw in the sample code provided for application store usage:

```
private void btnSubmit_Click(object sender, System.EventArgs e)
{
    if(IsValid)
    {
        Application.Lock();
        Application[txtName.Text] = txtValue.Text;
        Application.UnLock();

        lblResult.Text = "The value of <b>" + txtName.Text +
                        "</b> in the Application object is <b>" +
                        Application[txtName.Text].ToString() + "</b>";
    }
}
```

This code does not validate input nor does it sanitize output. Therefore a malicious user could attack any user of the web server through XSS.

See also: Server.HtmlEncode() documentation.

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=36]. If user input accepts HTML or XML markup, meta characters and the characters used for HTML should be escaped.	The following test case assumes that any data input to the bean that implements the sanitization has been through a canonicalization process prior to submittal.	<b>Wizard (4)</b> . By taking advantage of an application written in a previous test case, the implementation of this best practice required the creation of one Java class and the modification of modification of two XML configuration files.	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 4, Unix 4

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=36.1]. Implement an input filter that sanitizes HTML/XML markup.	Implementation Complexity	<b>Wizard+ (4)</b> . Efficiencies and a reductions in the overall level of effort required to create a Struts-Validator based HTML sanitization routine were gained by extending code created for a previous test case.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Vendor supplied documentation on Struts Validators was scant and had a enormous learning curve. The implementer was expected to possess a profound knowledge of the Apache Struts environment and comprehensive understanding of the WebSphere deployment environment. The documentation did not provide information that could lead to insecure designs and deployments; its overall uselessness warranted this rating.
	Implementor Competence	<b>Novice (5)</b> . The tester had no prior struts implementation experience. After six hours with the documentation, it was possible to use the functionality to create fairly sophisticated user input filtering routines.
	Time to Implement	<b>Medium to High (2)</b> . Extending the existing code base took slightly more then one hour

**Ease of Securing scores**, based on mean of 1 test case: Linux 3.25, Unix 3.25

### Notes

#### Test Case 36.1

The RFC 1738 specification defining Uniform Resource Locators (URLs) and the RFC 2396 specification for Uniform Resource Identifiers (URIs) both restrict the characters allowed in a URL or URI to a subset of the US-ASCII character set. According to the RFC 1738 specification, "only alphanumerics, the special characters "\$\_.+!\*'()", and reserved characters used for their reserved purposes may be used unencoded within a URL."

These characters, taken as a somewhat arbitrary base, may be combined with the more obviously “dangerous” characters outlined in the table below to form an extremely draconian (and massively inefficient) HTML sanitization routine:

Name	HTML code	Display	Description
&lsquo;		#	left single quote
&rsquo;		#	right single quote
&quot;	&#34;	"	double quotation mark
	&#37;	%	percent sign
&amp;	&#38;	&	ampersand
	&#39;	'	apostrophe
	&#40;	(	left parenthesis
	&#41;	)	right parenthesis
&lt;	&#60;	<	less-than sign
&gt;	&#62;	>	greater-than sign
	&#96;	`	grave accent
	&#123;	{	left curly brace
	&#124;		vertical bar
	&#125;	}	right curly brace
&laquo;	&#171;	«	left angle quote
&raquo;	&#172;	»	right angle quote

The source file below ( `HTMLEscape.java` ) implements the HTML sanitization routines required by this test case.

```
package app;

/**
 * HTMLEscape.
 *
 * @author Frank Heidt *
 */
public class HTMLEscape
{
    public HTMLEscape()
    {
    }

    /**
     * Process the specified HTTP data, replacing and "dangerous"
     * characters
  
```

```
* @param  String  str  The string to escape.  
*/  
public static final String doEscape(String str) {  
    StringBuffer outBuf = new StringBuffer();  
    char[] block = str.toCharArray();  
    String outStr = null;  
    int i, last;  
  
    for (i=0, last=0; i < block.length; i++) {  
        switch(block[i]) {  
            case '#' :  
                outStr = "\\'";           // left single quote  
                break;  
            case '#' :  
                outStr = "\'";           // right single quote  
                break;  
            case '\"' :  
                outStr = "\\\"";         // double quotation mark  
                break;  
            case '%' :  
                outStr = "\\%";          // percent sign  
                break;  
            case '&' :  
                outStr = "\\&";          // ampersand  
                break;  
            case '\'' :  
                outStr = "\\'";          // apostrophe  
                break;  
            case '(' :  
                outStr = "\\(";          // left parenthesis  
                break;  
            case ')' :  
                outStr = "\\)";          // right parenthesis  
                break;  
            case '<' :  
                outStr = "\\<";          // less than  
                break;  
            case '>' :  
                outStr = "\\>";          // greater than  
                break;  
            default :  
                /* no-op */ ;  
        }  
        if (outStr != null){  
            outBuf.append(block, last, i - last);  
            outBuf.append(outStr);  
            outStr = null;  
            last = i + 1;  
        }  
    }  
    if(last < block.length) {  
        outBuf.append(block, last, i - last);  
    }  
    return outBuf.toString();  
}
```



### 4.2.6.3 Negative Data Validation

---

#### Negative Data Validation

Negative data validation is a second layer of defense against malicious input if positive data validation fails. For web services it is very important that data be validated properly since their interface specification is published as WSDL which can aid attackers in generating hostile input. Web services are weighted high. Web application and intranet scenarios are weighted medium. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 3
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Negative Data Validation topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

#### Microsoft .NET

##### **Best Practice Compliance Analysis**

Best Practice	Options	Rating	Recommendation
[id=35]. Known bad input such as malformed parameters, strings of excessive length, and dynamic query strings should be rejected. It is best to accept only known good data, but this is not always possible.	Several mechanisms are available to perform this test case, see below for an explanation of each.	<b>Developer extends (3)</b> .NET does not provide mechanisms targeted specifically to bad input handling.	

**Best Practice Compliance score**, based on mean of 1 best practice: 3

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=35.1]. Implement an input filter that rejects known bad data.	Implementation Complexity	<b>Small amount of code (3)</b> . The complexity does not lay in implementing a functional solution, but more in designing a framework that will address all requests.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . MSDN only contains a few generic documents covering this topic.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited .NET exposure.
	Time to Implement	<b>Medium to High (2)</b> . Designing and implementing a framework is fairly time-consuming activity.

**Ease of Securing score**, based on mean of 1 test case: 2.75

### Notes

#### Test Case 35.1

Several solutions can address this issue:

- Helper function replacing malicious characters for each request to user input. This solution does not prevent direct calls to the Request object. It also forces an additional function call for every variable usage.
- Implement an ISAPI filter to replace malicious characters with harmless content. This is a great solution to the extent that it requires a different technology and area of expertise. Our experience has been that implementing a secure ISAPI filter is hard to achieve
- Implement a framework through inheritance so that any page on the site inherits the base class. In turn, the base class will validate all arguments and deny access to a page if an argument is deemed to contain malicious content.

@stake decided to implement a variant of the last option more in line with asp.NET:

Implementing Global\_PreRequestHandlerExecute() in global.asax.vb. This method is called before every page request is processes. It will go through every request argument and if it detects malicious input, it will redirect to an error page. We would have preferred to render potentially harmful characters harmless through a search and replace; and for harmful patterns (such as "<script"), deny the request. Unfortunately, the Request object being read-only, only request denial is possible.

Following is a sample basic implementation:

```
Dim reg As Regex

Private Sub Global_PreRequestHandlerExecute(ByVal sender As Object, ByVal e As System.EventArgs)
    Handles MyBase.PreRequestHandlerExecute
```

```

Dim key As String
If (IsNothing(reg)) Then
    reg = New Regex("script", RegexOptions.Compiled
                    Or RegexOptions.Singleline
                    Or RegexOptions.IgnoreCase)
End If

For Each key In Request.Params.AllKeys
    ' Response.Write(key + "=" + Request(key) + "<br>")
    If reg.IsMatch(Request(key)) Then
        Response.StatusCode = 500
        Response.Close()
        Return
    End If
Next
End Sub

```

To be ready for production, his sample code needs to extend the regular expression pattern and redirect to a specific error page.

Not much info on Microsoft sites on handling XSS and SQL injection issues

#### Cross-site Scripting Overview

@stake accidentally discovered that the .NET Framework does attempt to detect XSS input of HTTP GET arguments:

The screenshot shows a Microsoft Internet Explorer browser window. The title bar reads "A potentially dangerous Request.QueryString value was detected from the client (VAR=<b>hello</b>)". The address bar shows the URL "http://127.0.0.1/Negative/WebForm.aspx?VAR=<b>hello</b>". The main content area displays a red error message: "Server Error in '/Negative' Application." Below it, a red warning message says: "A potentially dangerous Request.QueryString value was detected (VAR=<b>hello</b>)." A detailed description follows: "Description: Request Validation has detected a potentially dangerous client input value, and processing of the request attempt to compromise the security of your application, such as a cross-site scripting attack. You can disable request validation or in the configuration section. However, it is strongly recommended that your application explicitly check all input values for potentially dangerous content." An exception details section at the bottom states: "Exception Details: System.Web.HttpRequestValidationException: A potentially dangerous Request.QueryString value was detected from the client (VAR=<b>hello</b>)".

Given the lack of detailed information, in our experience this only detects the presence of < characters.

## IBM WebSphere J2EE

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=35]. Known bad input such as malformed parameters, strings of excessive length, and dynamic query strings should be rejected. It is best to accept only known good data, but this is not always possible.	The creation of filter that <i>rejects known bad data</i> may be implemented using the Struts Validation routines included with WebSphere 5.0. In order to simplify the otherwise widely divergent requirement of ascertaining what is <i>bad</i> , the test-case will be inverted to except only valid data. This is in keeping with industry best practices of "failing-closed" and "default-deny"	<b>Wizard (4)</b> . By taking advantage of an application written in a previous test case, the implementation of this best practice was a simple exercise of modifying one XML configuration file containing input validation rules.	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 4, Unix 4

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=35.1]. Implement an input filter that rejects known bad data.	Implementation Complexity	<b>Wizard+ (4)</b> . Massive efficiencies were gained in the reduction of implementation complexity by extending the <code>validator.xml</code> file created for a previous test case
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Vendor supplied documentation on Struts Validators was scant and had an enormous learning curve. The implementer was expected to possess a profound knowledge of the Apache Struts environment and comprehensive understanding of the WebSphere deployment environment. The documentation did not provide information that could lead to insecure designs and deployment environments; its overall uselessness warranted this rating.
	Implementor Competence	<b>Novice (5)</b> . The tester had no prior struts implementation experience. After six hours with the documentation, it was possible to use the functionality to create fairly sophisticated user input filtering routines.
	Time to Implement	<b>Low to Medium (4)</b> . Extending the existing <code>validator.xml</code> file took less than fifteen minutes

**Ease of Securing scores**, based on mean of 1 test case: Linux 3.75, Unix 3.75

## Notes

### Test Case 35.1

For the sake of brevity, only modifications to the previously created `validator.xml` configuration file is included below. the newly added functionality creates a United States Social Security Number and applies it to the `registerForm`

```
<constant>
<constant-name>ssn</constant-name>
<constant-value>^\(?(\d{3})\)?[- ] ?(\d{2})\)?[- ] ?(\d{4})$</constant-value>
</constant>
...
<formset
```

```
<form name="register">
  <field property="ssn"
    depends="mask">
    <arg0 key="register.ssn.displayname"/>
    <var>
      <var-name>mask</var-name>
      <var-value>${ssn}</var-value>
    </var>
  </field>
</form>
</formset>
```

## 4.2.6.4 Output Filtering

---

### Output Filtering

Output filtering helps a developer build an application which is not susceptible to cross site scripting attacks. It is rated low for all scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 1
- Web Service: 1
- Intranet: 1

The sections that follow contain the results of @stake's analysis of the Output Filtering topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=38]. When information is displayed to users, it should be escaped. HTML should be rendered inactive to prevent cross site scripting attacks.	.NET provides functions for rendering user controlled output inactive from an HTML perspective.	<b>Wizard (4)</b> . This functionality is available and easy to invoke.	

**Best Practice Compliance score**, based on mean of 1 best practice: 4

#### Level of Effort Analysis

Test Case	Criteria	Rating
[id=38.1]. Construct an output filter that renders HTML inactive.	Implementation Complexity	<b>Small amount of code (3)</b> . Only one function call is needed to escape HTML
	Documentation and Examples	<b>Adequate (3)</b> . The documentation is adequate but does not fully describe the security issues surrounding this test case.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . The developer has extensive web application development experience but very little with ASP and ASP.NET.
	Time to Implement	<b>Low (5)</b> . This is very quick to implement.

## Level of Effort Analysis

Test Case	Criteria	Rating
<b>Ease of Securing score</b> , based on mean of 1 test case: 3.75		

## Notes

### Test Case 38.1

.NET provides 2 functions useful to filter output:

- `HttpServerUtility::HTMLEncode()`
- `HttpServerUtility::URLEncode()`

Unfortunately, only limited information could be found regarding the functions' usefulness.

@stake discovered that, by default, the .NET Framework does detect XSS input on HTTP GET arguments:

The screenshot shows a Microsoft Internet Explorer browser window. The title bar reads "A potentially dangerous Request.QueryString value was detected from the client (VAR=<b>hello</b>)". The address bar shows the URL "http://127.0.0.1/Negative/WebForm.aspx?VAR=<b>hello</b>". The main content area displays a red error message: "Server Error in '/Negative' Application.  
  
*A potentially dangerous Request.QueryString value was detected (VAR=<b>hello</b>).*" Below this, there is a "Description:" section and an "Exception Details:" section, both of which are mostly cut off by the bottom of the screenshot. The status bar at the bottom of the browser window shows "Exception Details: System.Web.HttpRequestValidationException: A potentially dangerous Request.QueryString val..."

Although this feature did not appear to be documented, in our experience it only detected < characters.

Using the `HTMLEncode()` function:

```
<%=Server.HTMLEncode(Request("VAR"))%>
```

Provides an added layer of security in case input validation failed to detect malicious input.

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=38]. When information is displayed to users, it should be escaped. HTML should be rendered inactive to prevent cross site scripting attacks.	The model used by WebSphere will automatically filter <b>any</b> HTML output written by the <bean:write> JSP tag. This is the default behavior of the platform.	<b>Transparent (5)</b> . The platform natively supports this best practice. No code modifications or platform configuration changes are required.	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 5, Unix 5

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=38.1]. Construct an output filter that renders HTML inactive.	Implementation Complexity	<b>Wizard (5)</b> . Anytime a Java Bean -specifically the <bean:write> JSP tag is used to present contents all sensitive HTML characters are escaped. No particular modifications are required to gain this functionality
	Documentation and Examples	<b>Suitable (4)</b> . The Apache documentation was completely clear, spelling out the default nature of HTML character escaping.
	Implementor Competence	<b>Expert/intermediate (2)</b> . The tester has wide experience in configuring and deploying web servers.
	Time to Implement	<b>Low (5)</b> . No effort was expended in order to meet the requirements of the test case.

**Ease of Securing scores**, based on mean of 1 test case: Linux 4, Unix 4

### Notes

#### Test Case 38.1

The following JSP code snippet illustrates HTML output filtering functionality. By virtue of using the <bean:write> JSP tag on line 10 any data received from the primitivesForm Java bean will be *escaped*

```

1  <%@ page contentType="text/html;charset=UTF-8" language="Java" %>
2  <%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
3  <%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>

4  <html:html locale="true">
5  <head>
6  <title><bean:message key="index.title"/></title>
7  <html:base/>
8  </head>
9  <body bgcolor="white">
10 <p>You Wrote: <bean:write name="primitivesForm" property="keyin"/>
11 </body>
```

```
12 </html:html>
```

Test output from an attempt to include a snippet of JavaScript POST'ed in an HTML TEXTAREA widget is shown below:

```
<html lang="en">
<head>
<title>Struts Validator</title>
<base href="http://localhost:9080/struts-validator/result.jsp">
</head>
<body bgcolor="white">
<p>You Wrote: &lt;html&gt;&lt;body&gt;&lt;script
language="javaScript"&gt;window.alert("Hello")
; &lt;/script&gt;&lt;/body&gt;&lt;/html&gt;
</body>
</html>
```

## Documentation Sources

The Apache web site provided API documentation describing Apache Struts API Documentation: Class WriteTag

IBM supplies a extremely detailed DeveloperWorks regarding the Use a custom tag library to encode dynamic content.

## 4.2.6.5 Positive Data Validation

---

### Positive Data Validation

@stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 3
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Positive Data Validation topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=34]. User-entered input is verified before being accepted as valid. Unconstrained and unchecked data can lead to serious application compromises. Ideally, the application should only accept meaningful input.	The .NET Framework provides several validator classes that perform both client side and server side validation of user input.	<b>Wizard (4)</b> .	

**Best Practice Compliance score**, based on mean of 1 best practice: 4

#### Level of Effort Analysis

Test Case	Criteria	Rating
[id=34.1]. Implement an input filter that accepts only valid, named data.	Implementation Complexity	<b>Wizard+ (4)</b> . Most validators will not require code to implement, requiring only a drag and drop operation.
	Documentation and Examples	<b>Best practice (5)</b> . Extensive documentation available.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited .NET exposure.
	Time to Implement	<b>Low (5)</b> . Validators are very fast to create.

### **Level of Effort Analysis**

---

Test Case	Criteria	Rating
<b>Ease of Securing score</b> , based on mean of 1 test case: 4.5		

---

### **Notes**

#### **Test Case 34.1**

.NET provides several validators:

- *RequiredFieldValidator* - Checks that the user has entered or selected anything.
- *RegularExpressionValidator* - Checks user input against a regular expression. This allows a wide variety of checks to be made and can be used for things like ZIP codes and phone numbers.
- *CompareValidator* - Compares an input control to a fixed value or another input control. It can be used for password verification fields, for example. It is also possible to do typed date and number comparisons.
- *RangeValidator* - Much like CompareValidator, but can check that the input is between two fixed values.
- *CustomValidator* - This allows you to write your own code to take part in the validation framework.

These validators are to be used in conjunction with a ValidationSummary control to automatically display error messages.

The framework provides client and server validation or optionally, server only.

Using them requires little work:

1. Drag and Drop validator on HTML page
2. Point "ControlToValidate" property to input field
3. Customize validator properties such as error message, validation rules or client/server execution
4. Drag and Drop SummaryValidator
5. Implement appropriately your form submittal.

Here is how your submit handler might look:

```
public sub OnSubmit(source as Object, e as EventArgs)
    if Page.IsValid then
        ' Now we can perform our transaction.
    end if
end sub
```

**Do not forget to call Page.IsValid in your event handlers. Otherwise, server side validation will be ignored.**

See also: User Input Validation in ASP.NET

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=34]. User-entered input is verified before being accepted as valid. Unconstrained and unchecked data can lead to serious application compromises. Ideally, the application should only accept meaningful input.	By default, Java Servlet Pages (JSP) when tied to an Enterprise Java Bean (EJB) will only accept parameters defined in the jsp and will ignore all other data items. In keeping with the objective nature of the test, a validation test case will be applied similar in scope to the one created previously for the .NET Framework. This test case will be implemented using the Struts Validation routines included with WebSphere 5.0.	<b>Developer extends (3)</b> . Implementing a Struts based Validator framework in an application is a non-trivial undertaking. Creating even the most trivial input filter requiring the direct modification of no less than five files, including: XML configuration files, application resource bundles, and Java source files. When finally accomplished, the Validator framework proved to be a powerful and flexible method for validating user input.	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 3, Unix 3

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=34.1]. Implement an input filter that accepts only valid, named data.	Implementation Complexity	<b>Medium amount of code (2)</b> . To develop a useful set of custom input filters, or to implement a filtering scheme based on the supplied basic validator mapping would require a significant amount of code.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Vendor supplied documentation on validators was scant and had a enormous learning curve. The implementer was expected to possess a profound knowledge of the Apache Struts environment and comprehensive understanding of the WebSphere deployment environment. The documentation did not provide information that could lead to insecure designs and deployment environments; its overall uselessness warranted this rating.
	Implementor Competence	<b>Novice (5)</b> . The tester had no prior Struts implementation experience. After six hours with the documentation, it was possible to use the functionality to create fairly sophisticated user input filtering routines.
	Time to Implement	<b>High (1)</b> . the complexity of the deployment environments and the lack of sound documentation caused the implementation of the test case to exceed four hours

**Ease of Securing scores**, based on mean of 1 test case: Linux 2.5, Unix 2.5

### Notes

#### Test Case 34.1

The Struts distribution includes a Validator package containing classes that enforce basic input validation rules for data sent to ActionForm beans.

The table below lists the 14 basic validation mappings shipped with the Struts Validator classes. These base validators are appropriate for the majority of most web applications. When necessary, these base validators can be expanded with a plugin interface to accomplish any input/message validation an application could require.

Validator	Purpose
required	Succeeds if the field contains any characters other than white space
mask	Succeeds if the value matches the regular expression given by the mask attribute.
range	Succeeds if the value is within the values given by the min and max attributes.
maxLength	Succeeds if the field's length is less than or equal to the max attribute.
minLength	Succeeds if the field's length is greater than or equal to the min attribute.
byte	Succeeds if the value can be converted to the corresponding primitive
short	Succeeds if the value can be converted to the corresponding primitive
integer	Succeeds if the value can be converted to the corresponding primitive
long	Succeeds if the value can be converted to the corresponding primitive
float	Succeeds if the value can be converted to the corresponding primitive
double	Succeeds if the value can be converted to the corresponding primitive
date	Succeeds if the value represents a valid date. A date pattern may be provided.
creditCard	Succeeds if the value could be a valid credit card number.
email	Succeeds if the value could be a valid e-mail address.

There are several components that make up the Validator framework.

- Validators
- Configuration Files
- Resource Bundle
- JSP Custom Tags
- Validator Form Classes

For the sake of brevity, only the configuration file used to implement the four basic user-input filtering rules is included below.

Notes on the `validator.xml` file:

- The constructs within the `<global></global>` tags are regular expression *constants* used by the `match` validation operator

```
<?xml version="1.0" encoding="UTF-8"?>
<form-validation>
  <formset>

    <global>
      <constant>
        <constant-name>phone</constant-name>
        <constant-value>^(\d{3})\)?[- ]?(\d{3})[- ]?(\d{4})$</constant-value>
      </constant>

      <constant>
        <constant-name>zip</constant-name>
        <constant-value>^\d{5}\d*&$</constant-value>
      </constant>
    </global>

    <form name="logonForm">
      <field property="userName"
        depends="required">
        <arg0 key="loginForm.username" />
      </field>

      <field property="password"
        depends="required">
        <arg0 key="loginForm.password" />
      </field>
    </form>

    <form name="registerForm">
      <field property="cityStateZip.zipPostal[1]"
        depends="required,mask">
        <arg0 key="registerForm.zip.displayname" />
      <var>
        <var-name>mask</var-name>
        <var-value>${zip}</var-value>
      </var>

      <field property="phone"
        depends="mask">
        <arg0 key="registerForm.phone.displayname" />
      <var>
        <var-name>mask</var-name>
        <var-value>${phone}</var-value>
      </var>
    </field>
  </form>
</formset>
</form-validation>
```

## 4.2.6.6 Type Checking

---

### Type Checking

Type checking helps a developer build proper input validation. It is not a requirement for proper input validation. It is rated low for all scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 1
- Web Service: 1
- Intranet: 1

The sections that follow contain the results of @stake's analysis of the Type Checking topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=37]. User-entered character data should only allow input valid for the underlying object's field type. Validation of input fields includes checking for known data types such as integer or date, disallowing input of other types.	.NET provides a series of validators automating user input validation in web forms.	<b>Transparent (5)</b> . The validators are very easy to incorporate into programs.	No recommendation.

**Best Practice Compliance score**, based on mean of 1 best practice: 5

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=37.1]. Implement an input filter that performs date, integer, and string type checking of user-supplied data.	Implementation Complexity	<b>Wizard+ (4)</b> . Little or no code is necessary to perform web form validation.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation is well-written and mentions a number of security best practices.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited .NET exposure.
	Time to Implement	<b>Low to Medium (4)</b> . Configuring the validators is a very quick operation.

**Ease of Securing score**, based on mean of 1 test case: 4.25

### Notes

#### Test Case 37.1

Type checking of user input can be done through a RangeValidator:

```
<asp:TextBox id="IntBox" runat="server"></asp:TextBox>

<asp:RangeValidator id="RangeValidator1" runat="server" ControlToValidate="IntBox"
    ErrorMessage="Need to be an integer between -1 and 10" MaximumValue="10"
    MinimumValue="-1" Type="Integer">*</asp:RangeValidator>
```

This will verify that the Text field called "IntBox" only contains integers between -1 and 10.

Checking for a date range can be done as follows:

```
<asp:TextBox id="DateBox" runat="server"></asp:TextBox>

<asp:RangeValidator id="RangeValidator1" runat="server" ControlToValidate="DateBox"
    ErrorMessage="Date between 1/1/2003 and 4/10/2003"
    MaximumValue="4/10/2003" MinimumValue="1/1/2003"
    Type="Date">*</asp:RangeValidator>
```

This will verify that DateBox is between 1/1/2003 and 4/10/2003.

Similarly, one can use the RangeValidator to ensure a field is of type string, double, or currency.

CompareValidator can similarly used to validate data type without validating a range. In that case, the "Operator" argument needs to be set to "DataTypeCheck"

## IBM WebSphere J2EE

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=37]. User-entered character data should only allow input valid for the underlying object's field type. Validation of input fields includes checking for known data types such as integer or date, disallowing input of other types.	This test case will be implemented using the Struts Validation routines included with WebSphere 5.0. and by extending the <code>registerForm</code> bean created in a previous test case.	<b>Wizard (4)</b> . By taking advantage of an application written in a previous test case, the implementation of this best practice was accomplished in short order with a moderate amount of coding or configuration changes	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 4, Unix 4

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=37.1]. Implement an input filter that performs date, integer, and string type checking of user-supplied data.	Implementation Complexity	<b>Small amount of code (3)</b> . By taking advantage of an application written in a previous test case, the implementation of this best practice required the modification of two Java class files, one JSP file, two XML configuration files, and one resource bundle
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Vendor supplied documentation on Struts Validators was scant and had an enormous learning curve. The implementer was expected to possess a profound knowledge of the Apache Struts environment and comprehensive understanding of the WebSphere deployment environment. The documentation did not provide information that could lead to insecure designs and deployment environments; its overall uselessness warranted this rating.
	Implementor Competence	<b>Novice (5)</b> . The tester had no prior Struts implementation experience.
	Time to Implement	<b>Medium (3)</b> . Extending the existing code base took slightly less than thirty minutes

**Ease of Securing scores**, based on mean of 1 test case: Linux 3.25, Unix 3.25

## Notes

### Test Case 37.1

For the sake of brevity, only the salient portions of the `validator.xml` configuration and the `struts_config` files are included below. The new functionality will Validate an `Integer`, a `Float` and a `String:Date`

Source of `validator.xml`

```
<form name="primitivesForm">

<!-- integer input field -->

<field property="integer" depends="required,mask,maxlength">
  <arg0 keys="primitivesForm.integer.displayname"/>
  <arg1 name="maxlength" key="${var:maxlength}" resource="false"/>
  <var>
```

```

<var-name>mask</var-name>
<var-value>^\d+$</var-value>
</var>
<var>
    <var-name>maxlength</var-name>
    <var-value>15</var-value>
    </var>
</field>

<<!-- float input field -->
<field      property="float" depends="required,float">
    <arg0 key="typeForm.float.displayname"/>
</field>

<<!-- date input field -->
<field      property="date" depends="required,date">
    <arg0 key="primitivesForm.date.displayname"/>
    <var>
        <var-name>datePatternStrict</var-name>
        <var-value>MM/dd/yyyy</var-value>
    </var>
</field>
</form>

```

#### Source of struts-config.xml

```

<!DOCTYPE struts-config PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
"http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
<form-beans>
    <form-bean name="primitivesForm"
        type="org.apache.struts.webapp.validator.PrimitivesForm"/>
</form-beans>

<global-forwards>
    <forward    name="home"           path="/index.jsp"/>
</global-forwards>

<action-mapping>
    <action   path="/primitives"
        type="org.apache.struts.webapp.validator.PrimitivesAction"
        input="/primitives.jsp" name="primitivesForm" scope="request" validate="true">
        <forward
            name="success" path="/index.jsp">
        </forward>
    </action>
</action-mappings>

<message-resources
    parameter="org.apache.struts.webapp.validator.ApplicationResources"/>

<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
    <set-property property="pathnames" value="/WEB-INF/validator-rules.xml",

```

```
    /WEB-INF/validation.xml"/>
  </plug-in>
</struts-config>
```

## 4.2.7 Information Disclosure

---

### Information Disclosure

Information disclosure is the unauthorized presentation of sensitive or confidential information. Proper application design should safeguard against information disclosure to protect both application end users and intellectual property kept by the application. All application scenarios are weighted medium.

The Information Disclosure area of analysis includes these topics:

- Error Handling
- Stack Traces and Debugging

The table below describes the topics at a high level, and lists prevailing best practices. The right-hand columns show the weights @stake assigned for each of the scenarios.

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Error Handling</b> . Web and application servers provide facilities for handling error conditions that may be triggered by anomalous user activities or system malfunctions. Detailed output helps system administrators diagnose potential problems. When configured improperly, error handlers can also be a prime source of information for malicious attackers. We weighted this topic as medium for all three scenarios: web, web service, and intranet applications.	When errors are encountered by the web application, the server returns concise error messages to the service client; specific details are logged securely on the server and not disclosed.	2	2	2
<b>Stack Traces and Debugging</b> . It is important for all applications to properly handle debugging information. All application scenarios are weighed medium.	If a server-based web application fails, it does so in a manner that does not provide detailed failure information to the user.	2	2	2

Scenario weights range from 1 (relatively unimportant) to 3 (extremely important). A scenario weight of zero means that the topic does not apply.

### Findings

The table below summarizes @stake's quantitative findings. For each analysis topic, the two high-level metrics (Best Practice Compliance and Ease of Securing) are shown. These are the arithmetic means of the underlying best practice and test case scores. The Ease of Securing metric is further broken down into four sub-metrics.

Topic	Platform	Best Practice Compliance	Overall	Ease of Securing			
				Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Error Handling <i>1 best practice, 1 test case</i>	.NET Windows	<b>5</b>	<b>4.25</b>	5	3	4	5
	WebSphere Linux	<b>4</b>	<b>3.75</b>	4	4	3	4
	WebSphere Unix	<b>4</b>	<b>3.75</b>	4	4	3	4
Stack Traces and Debugging <i>1 best practice, 1 test case</i>	.NET Windows	<b>5</b>	<b>4.25</b>	5	3	4	5
	WebSphere Linux	<b>5</b>	<b>4</b>	5	3	3	5
	WebSphere Unix	<b>5</b>	<b>4</b>	5	3	3	5

This area of analysis covered two (2) test cases pertaining to the platform's ability to limit unauthorized presentation of sensitive or confidential information. The topics under evaluation included Error Handling and Stack Traces and Debugging. Under both test cases, Microsoft scored slightly higher. The default configuration of the .NET Framework limits information disclosure.

The individual sections for each analysis topic describe @stake's findings in further detail.

## 4.2.7.1 Error Handling

---

### Error Handling

Web and application servers provide facilities for handling error conditions that may be triggered by anomalous user activities or system malfunctions. Detailed output helps system administrators diagnose potential problems. When configured improperly, error handlers can also be a prime source of information for malicious attackers. We weighted this topic as medium for all three scenarios: web, web service, and intranet applications. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Error Handling topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=52]. When errors are encountered by the web application, the server returns concise error messages to the service client; specific details are logged securely on the server and not disclosed.	The .NET Framework enables developers to display verbose error messages to all users, to local ones only (i.e. users on the same machine), or not at all.	<b>Transparent (5)</b> . By default web applications built with Visual Studio .NET will only display verbose error messages when the browser is running on the same machine as the web server (i.e. for development/debugging purposes).	

**Best Practice Compliance score**, based on mean of 1 best practice: 5

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=52.1]. Configure server to provide terse errors only.	Implementation Complexity	<b>Wizard (5)</b> . A single configuration setting defines whether specific details are returned with error messages.
	Documentation and Examples	<b>Adequate (3)</b> . Documentation is clear but does not mention the information disclosure implications of enabling remote viewing of verbose error messages.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive web application development experience but very little with ASP and ASP.NET.
	Time to Implement	<b>Low (5)</b> . No configuration changes need to be made to implement this best practice.

**Ease of Securing score**, based on mean of 1 test case: 4.25

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=52]. When errors are encountered by the web application, the server returns concise error messages to the service client; specific details are logged securely on the server and not disclosed.	The WebSphere Application Server provides mappings between error codes or exception types and paths to resources in the Web application. These mappings define the content to display to the client in the event of a specific error or exception. The granularity of exception and error reporting can be controlled with great specificity using functionality exposed by the Web Application Deployment Descriptor wizard.	<b>Wizard (4)</b> . WebSphere displays verbose error messages by default. In order to bring the platform into best practice compliance, several steps must be completed using two or more property sheets.	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 4, Unix 4

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=52.1]. Configure server to provide terse errors only.	Implementation Complexity	<b>Wizard+ (4)</b> . Manipulation of the Web Application Deployment Descriptor is accomplished via a simple property sheet
	Documentation and Examples	<b>Suitable (4)</b> . Documentation in online help files supplied with the WebSphere development environment were clear, accurate and to the point.
	Implementor Competence	<b>Intermediate (3)</b> . The tester has wide experience in configuring and deploying web based applications.
	Time to Implement	<b>Low to Medium (4)</b> . The modifications to the Web Application Deployment Descriptor were accomplished in short order

**Ease of Securing scores**, based on mean of 1 test case: Linux 3.75, Unix 3.75

## Notes

### Test Case 52.1

Error page locations allow a Servlet to find and serve a URI to a client based on a specified error status code or exception type. These properties are used if the error handler is another Servlet or JSP file. The properties specify a mapping between an error code or exception type and the path of a resource in the Web application. The container examines the list in the order that it is defined, and attempts to match the error condition by status code or by exception class. On the first successful match of the error condition, the container serves back the resource defined in the Location property. Overall, this functionality was somewhat more flexible than that exposed by the .NET Framework: it was also possible to specify Java exceptions as well as HTTP error coded in the exception field of the property sheet wizard as illustrated below:

The screenshot shows the 'WebSphere v5.0 Test Enviro...' configuration interface. The 'welcome Pages' section lists files: index.html, index.htm, index.jsp, default.html, default.htm, and default.jsp. It includes buttons for Add, Remove, Up, and Down. The 'Login' section shows configuration for Authentication method (Unspecified), Realm name, Login page, and Error page. The 'Error Pages' section lists resources for HTTP error codes: 404 and java.io.FileNotFoundException, both mapped to /error\_404.htm. A toolbar at the bottom includes Overview, Servlets, Filters, Listeners, Security, Environment, References, Pages, Parameters, MIME, Extensions, and Source.

Error Code	Location
404	/error_404.htm
java.io.FileNotFoundException	/error_404.htm

## 4.2.7.2 Stack Traces and Debugging

---

### Stack Traces and Debugging

It is important for all applications to properly handle debugging information. All application scenarios are weighed medium. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Stack Traces and Debugging topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=53]. If a server-based web application fails, it does so in a manner that does not provide detailed failure information to the user.	The .NET Framework enables developers to display stack traces to all users, to local ones only (i.e. users on the same machine), or not at all.	<b>Transparent (5)</b> . By default web applications built with Visual Studio .NET will only display stack traces when the browser is running on the same machine as the web server.	

**Best Practice Compliance score**, based on mean of 1 best practice: 5

---

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=53.1]. Configure server to disable stack traces.	Implementation Complexity	<b>Wizard (5)</b> . A single configuration setting defines whether stack traces are returned with error messages.
	Documentation and Examples	<b>Adequate (3)</b> . Documentation is clear but does not mention the information disclosure implications of enabling remote viewing of stack traces.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive web application development experience but very little with ASP and ASP.NET.
	Time to Implement	<b>Low (5)</b> . No configuration changes need to be made to implement this best practice.

**Ease of Securing score**, based on mean of 1 test case: 4.25

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=53]. If a server-based web application fails, it does so in a manner that does not provide detailed failure information to the user.	By default, Servlet errors are not reported to the client via printStackTrace() method invocation. Code would have to be deliberately constructed to create a stack trace visible to the end user.	<b>Transparent (5)</b> . By default, the platform does not display stack traces to the end user.	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 5, Unix 5

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=53.1]. Configure server to disable stack traces.	Implementation Complexity	<b>Wizard (5)</b> . No modifications to the platform were needed
	Documentation and Examples	<b>Adequate (3)</b> . There was little specific documentation available on the vendors web sites regarding the security implications of display trace information.
	Implementor Competence	<b>Intermediate (3)</b> . The tester has wide experience in configuring and deploying web based applications.
	Time to Implement	<b>Low (5)</b> . No modifications were required

**Ease of Securing scores**, based on mean of 1 test case: Linux 4, Unix 4

## 4.2.8 Runtime Container Security

---

### Runtime Container Security

Runtime container security features user, group, and code permissions for an application server runtime environment. Account privileges limit the privileges granted to the runtime environment as a whole, while code access security controls limit the privileges granted to individual applications.

The Runtime Container Security area of analysis includes these topics:

- Code Security
- Runtime Account Privileges

The table below describes the topics at a high level, and lists prevailing best practices. The right-hand columns show the weights @stake assigned for each of the scenarios.

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Code Security</b> . Code security features of the application runtime environment allow administrators to run application modules from internal or external sources in a granular manner. Generally, declarative policies specify that some modules may run with restricted permissions, while others run with more enhanced privileges. Code security features can help prevent attackers from running rogue code, or by replacing existing code. We weighted this topic as being of low importance for most customers, primarily because most server code is generally vetted and well protected. For a shared service situation, such as an ISP or third party service, the weights would be higher.	Code running in the application runtime environment is digitally signed by the originating party. When the classloader attempts to run a code module, it verifies that it was signed by a recognized party. Policies limit code from accessing particular runtime resources such as network sockets and filesystem resources. Code runs with a subset of privileges by default. When additional privileges are required, mechanisms exist that allow the code to acquire enhanced privileges and call privileged code. The application runtime environment, by default, prohibits code modules from calling native code that is not managed by the application runtime environment.	1	1	1
<b>Runtime Account Privileges</b> . Runtime account privileges are privileges available to the user ID used during normal operation of the service. This is weighted medium for all application scenarios.	Services should run as non-privileged users, with only the minimum privileges required for proper functionality	2	2	2

Scenario weights range from 1 (relatively unimportant) to 3 (extremely important). A scenario weight of zero means that the topic does not apply.

### Findings

The table below summarizes @stake's quantitative findings. For each analysis topic, the two high-level

metrics (Best Practice Compliance and Ease of Securing) are shown. These are the arithmetic means of the underlying best practice and test case scores. The Ease of Securing metric is further broken down into four sub-metrics.

Topic	Platform	Best Practice Compliance	Ease of Securing				
			Overall	Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Code Security <i>4 best practices, 5 test cases</i>	.NET Windows	<b>3.75</b>	<b>3.95</b>	3.8	4	3.8	4.2
	WebSphere Linux	<b>2.75</b>	<b>2.55</b>	2.4	1.6	4.2	2
	WebSphere Unix	<b>2.75</b>	<b>2.55</b>	2.4	1.6	4.2	2
Runtime Account Privileges <i>1 best practice, 1 test case</i>	.NET Windows	<b>5</b>	<b>4.5</b>	4	4	5	5
	WebSphere Linux	<b>4</b>	<b>3</b>	3	3	3	3
	WebSphere Unix	<b>4</b>	<b>3</b>	3	3	3	3

The security of the runtime environment ensures that rogue processes or code cannot damage other applications or host servers. @stake evaluated the degree to which application privileges could be locked down through operating system restrictions on server processes. We found that Windows Server 2003 installed a non-privileged user by default, and that the default privileges should be adequate for most customers. The equivalent process steps for WebSphere were more involved and operating-system specific. IBM's documentation was also somewhat sparse.

In addition, @stake evaluated the degree to which application privileges could be controlled through runtime container security policies. We were specifically looking to see support for *declarative* policies, code signing, native code protection, restrictions on socket and file permissions, and mechanisms for allowing code to programmatically acquire more privileges when needed. @stake found that .NET Framework provided a generally coherent and logically implemented set of code security controls through assembly runtime permissions, privilege assertions and support for so-called "strong naming" of modules. In the case of WebSphere, while the Java 2 security framework is more complex (some might say richer), IBM missed an opportunity to help developers leverage it. The documentation we reviewed was generally poor or non-existent, and there were few examples to help us understand IBM's implementation. We also found that support for a key feature of the Java 2 security framework — code signing — appeared to be missing. For these reasons, we would have difficulty recommending WebSphere for implementations where multiple customers needed to share an application server. Microsoft's approach was much better organized, documented and implemented.

The individual sections for each analysis topic describe @stake's findings in further detail.

## 4.2.8.1 Code Security

---

### Code Security

Code security features of the application runtime environment allow administrators to run application modules from internal or external sources in a granular manner. Generally, declarative policies specify that some modules may run with restricted permissions, while others run with more enhanced privileges. Code security features can help prevent attackers from running rogue code, or by replacing existing code. We weighted this topic as being of low importance for most customers, primarily because most server code is generally vetted and well protected. For a shared service situation, such as an ISP or third party service, the weights would be higher. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 1
- Web Service: 1
- Intranet: 1

The sections that follow contain the results of @stake's analysis of the Code Security topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=70]. Code running in the application runtime environment is digitally signed by the originating party. When the classloader attempts to run a code module, it verifies that it was signed by a recognized party.	.NET applications can be signed with Authenticode which can validate whether a publisher is trusted or they can be <i>strongly named</i> , which validates whether or not an assembly is a particular assembly. Since only one assembly can have a particular strong name, strong naming is a stricter test than Authenticode but does not give any indication of trust. Strong naming verifies that an assembly has not been tampered with since it was built and protects against spoofing. Delayed signing is available when the code developer does not have access to the private key.	<b>Wizard (4)</b> . A developer can sign a .NET assembly with a private key and bind a particular assembly to a strong name. Another assembly which consumes the signed assembly can require that it have a particular strong name and optionally that it have a particular version number. The common language runtime enforces this at runtime. A tool is provided to generate the keys and sign assemblies. The consumer of a strongly named assembly can specify the strong name at compile time or run time.	Developers should use strong names for all assemblies that they publish. They should also require all assemblies that they consume be strongly named.

---

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=71]. Policies limit code from accessing particular runtime resources such as network sockets and filesystem resources.	Policies can be implemented at various levels: enterprise, machine, and user. The machine level is where most policy settings will be performed. Policies can be applied to different code groups within a level. Examples are by Strong Name, Publisher, Directory, URL, Hash, Site, and Zone.	<b>Wizard (4)</b> . All policy settings can be accomplished using a wizard. Command line tools are also available. Custom policies can be created that will only allow access to particular file system resources or network sockets.	Developers should ship their product with a permissions set file that allows administrators to enforce the minimal set of privileges that their application needs. Administrators should follow the developers guidance to lock down an assembly to just the permissions it requires.
[id=72]. Code runs with a subset of privileges by default. When additional privileges are required, mechanisms exist that allow the code to acquire enhanced privileges and call privileged code.		<b>Developer extends (3)</b> . The .NET Framework allows an assembly to be configured to specifically deny permissions such as File IO. If a restricted resource needs to be accessed, a wrapper assembly can be built to access the resource. This smaller wrapper assembly can be given more trust by the administrator without giving this trust to a larger assembly. Security auditors need only verify the correctness of privileged operations within the privileged wrapper assembly and not throughout the entire application.	Privileged operations such as File IO should be wrapped in assemblies so that the practice of least privilege can be enforced within an application. The wrapper assembly should perform a demand on the calling assembly before it asserts the permissions needed to access a restricted resource so that it can vouch for its legitimacy. After performing the restricted operation the assembly should call RevertAssert to revert the permissions.
[id=73]. The application runtime environment, by default, prohibits code modules from calling native code that is not managed by the application runtime environment.	The .NET Framework configuration tool allows an administrator to set fine grained code access security permissions for assemblies running on the machine or being run by a particular user. ASP.NET allows a further level of configuration based on a particular location path. This is useful when you are hosting multiple web sites on one machine.	<b>Wizard (4)</b> . The default for managed code originating on the machine is to allow FullTrust which grants the permission to call native code. The default for ASP.NET code is FullTrust. An administrator can edit the web.config file for a particular site and change the default from Full to a lower level such as High. This will disallow native code execution.	ASP.NET administrators should default to High or lower trust so that native code is not executed. Native code access should be allowed on a case by case basis.

Best Practice Compliance score, based on mean of 4 best practices: 3.75

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=70.1]. Create a security policy that specifies that code from a particular source must be signed, and associate a recognized signer with the code source. Attempt to run code with the same package or class name, but without a signature; verify that it fails.	Implementation Complexity	<b>Wizard+ (4)</b> . A simple descriptor needs to be placed in the code to reference the strong name.
	Documentation and Examples	<b>Best practice (5)</b> . The SDK documentation had clear descriptions of the process. There were sample files that showed how to use strong names.
	Implementor Competence	<b>Novice/intermediate (4)</b> . Novice developer can implement.
	Time to Implement	<b>Low to Medium (4)</b> . Easy to implement once the concepts were understood.

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=70.2]. Perform the same test as above, but with a fraudulent signature (signed by party with the same distinguished name, but a different private key).	Implementation Complexity	<b>Wizard+ (4)</b> . A simple descriptor needs to be placed in the code to reference the strong name.
	Documentation and Examples	<b>Best practice (5)</b> . The SDK documentation had clear descriptions of the process. There were sample files that showed how to use strong names.
	Implementor Competence	<b>Novice/intermediate (4)</b> . Novice developer can implement.
	Time to Implement	<b>Low to Medium (4)</b> . Easy to implement once the concepts were understood.
[id=71.1]. Create a security policy file that limits file access by a given class to a single named local file location.	Implementation Complexity	<b>Wizard+ (4)</b> . A wizard can be used to accomplish all of the policy setting. It is a multistep process that requires understanding a few different concepts.
	Documentation and Examples	<b>Adequate (3)</b> . Documentation was good. It was not grouped in one place. Several pieces of documentation needed to be discovered and understood to understand the policy concepts.
	Implementor Competence	<b>Novice/intermediate (4)</b> . Intermediate developers should be able to uses policies.
	Time to Implement	<b>Low (5)</b> . No implementation required.
[id=72.1]. Create a less-privileged class that, by default, cannot access an object, file or method unless it possesses a specific privilege. Add supplemental code that allows the module to acquire the credentials needed to perform a privileged action and access the file.	Implementation Complexity	<b>Small amount of code (3)</b> . Code access security is a fairly complex concept. Several lines of code had to be written to perform the security checks.
	Documentation and Examples	<b>Adequate (3)</b> . Documentation was clear. No examples for building a wrapper assembly that was trusted that wrapped a restricted resource.
	Implementor Competence	<b>Intermediate (3)</b> . The concepts involved for building a wrapper assembly require a medium level developer familiar with security and object oriented programming.
	Time to Implement	<b>Medium (3)</b> . It took a less than an hour and several lines of code to implement.
[id=73.1]. Create a code module that attempts to access a native library or DLL present in the parent operating system, but is prohibited from doing so by the security policy. Create a second module that, by policy, is allowed to do so.	Implementation Complexity	<b>Wizard+ (4)</b> . Configuration is done through a wizard for the entire machine or through a text configuration file, web.config, for ASP.NET.
	Documentation and Examples	<b>Suitable (4)</b> . Documentation was good and sample configuration files were available.
	Implementor Competence	<b>Novice/intermediate (4)</b> . A intermediate level administrator can configure the system.
	Time to Implement	<b>Low (5)</b> . There was no implementation time

**Ease of Securing score**, based on mean of 5 test cases: 3.95

## Notes

### Test Case 70.1

The calling assembly could not be built because the referenced assembly did not have a strong name. If a

calling assembly has a strong name the referenced assembly must also.

CompanyAShared.cs:

```
using System;
using System.Reflection;
using System.Security.Permissions;

namespace SecuritySamples {

    public class SharedComponent {

        public static string GetInfo() {
            return "I am the correct component.";
        }
    }
}
```

CompanyACaller.cs:

```
using System;
using System.Reflection;

[assembly:AssemblyKeyFile("../keypair.dat")]

namespace SecuritySamples {

    public class GoodCaller {

        public static void Main() {

            Assembly.Load("CompanyAShared.dll,Version=1.0.0.0," +
                "Culture=neutral,PublicKeyToken=5460c711dce23c5d");

            try {
                Console.WriteLine(SharedComponent.GetInfo());
            }
            catch(Exception e) {
                Console.WriteLine("\nException occurred: {0}\n", e);
            }

            Console.Write("\nPress Enter to exit...");
            Console.Read();

        }
    }
}
```

Test Case 70.2

CompanyAShared.cs:

```
using System;
using System.Reflection;
using System.Security.Permissions;

[assembly:AssemblyKeyFile("../keypairB.dat")]

namespace SecuritySamples {

    public class SharedComponent {

        public static string GetInfo() {
            return "I shouldn't be called. I am not signed by the right key.";
        }
    }
}
```

CompanyACaller.cs:

```
using System;
using System.Reflection;

[assembly:AssemblyKeyFile("../keypair.dat")]

namespace SecuritySamples {

    public class GoodCaller {

        public static void Main() {

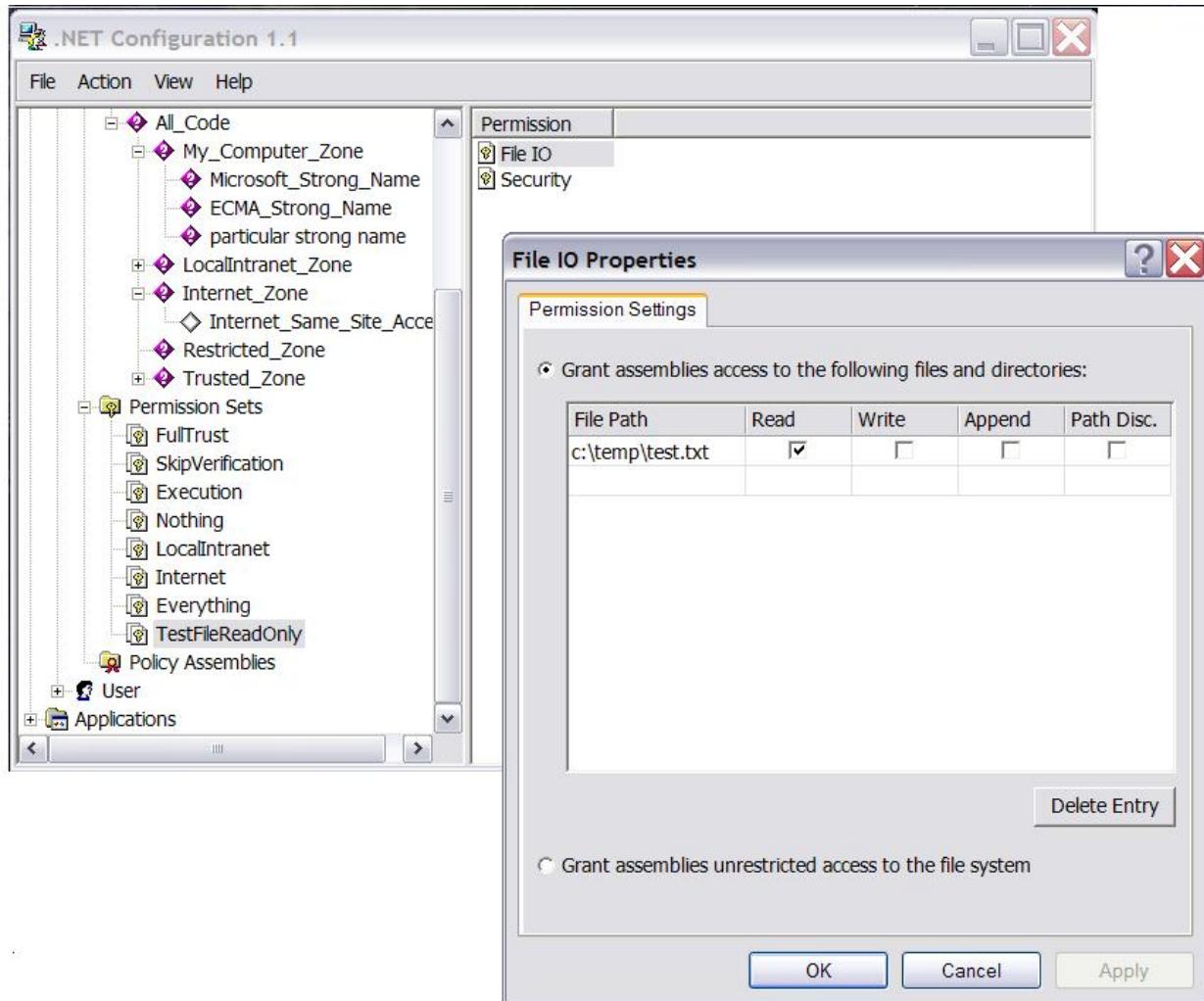
            try {
                Assembly.Load("CompanyAShared, PublicKeyToken=a5d015c7d5a0b012");
                Console.WriteLine(SharedComponent.GetInfo());
            }
            catch(Exception e) {
                Console.WriteLine("\nException occurred: {0}\n", e);
            }

            Console.Write("\nPress Enter to exit...");
            Console.Read();
        }
    }
}
```

```
Fusion log follows:
    === Pre-bind state information ===
    LOG: DisplayName = CompanyAShared, PublicKeyToken=a5d015c7d5a0b012
          (Partial)
    LOG: Appbase = C:\Program Files\Microsoft Visual Studio .NET\FrameworkSDK\Sample
```

```
s\QuickStart\howto\samples\security\codeidentitydemand\cs\
LOG: Initial PrivatePath = NULL
Calling assembly : CompanyACaller, Version=0.0.0.0, Culture=neutral, PublicKeyTo
ken=null.
===
LOG: Policy not being applied to reference at this time (private, custom, partia
l, or location-based assembly bind).
LOG: Post-policy reference: CompanyAShared, PublicKeyToken=a5d015c7d5a0b012
LOG: Attempting download of new URL file:///C:/Program Files/Microsoft Visual St
udio .NET/FrameworkSDK/Samples/QuickStart/howto/samples/security/codeidentitydem
and/cs/CompanyAShared.DLL.
WRN: Comparing the assembly name resulted in the mismatch: PUBLIC KEY TOKEN
```

### Test Case 71.1



Sample programs were created that read from the file specified in the policy and from a file not specified in the policy. When an attempt was made to access a file outside of that specified in the policy the following error was generated:

```
The file could not be read:
Request for the permission of type System.Security.Permissions.FileIOPermission,
mscorlib, Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b77a5c561934e089
failed.
```

### Test Case 72.1

```
// Add the classes in the following namespaces to our namespace
using System;
using System.IO;
using System.Security.Permissions;
using System.Security;

[assembly:FileIOPermission(SecurityAction.RequestMinimum, Unrestricted = true)]

///////////////////////////////



// This class represents the application itself
class App {
    public static void Main() {

        // Create a permission set that allows read, write, and append access
        // to SomeFile
        PermissionSet ps = new PermissionSet(PermissionState.None);
        ps.AddPermission(
            new FileIOPermission(FileIOPermissionAccess.Read |
                FileIOPermissionAccess.Write | FileIOPermissionAccess.Append,
                Path.GetFullPath("SomeFile")));

        // Deny access to the resources we specify
        ps.Deny();

        // Try to access resources using the permissions we've just denied.
        AttemptAccess("Deny permissions");

        // perform some test to check if a user has privileges to access
        // the file and if it passes continue.

        // Stop security checks at this point in the stack walk
        // for the specified permissions
        CodeAccessPermission.RevertDeny();

        // Try to access resources using the permissions we've just asserted.
```

```

        AttemptAccess("Assert permissions");

        // Deny access to the resources we specify
        ps.Deny();

        // Try to access resources using the permissions currently available.
        AttemptAccess("Deny permissions");

    }

    static public void AttemptAccess(String s) {
        FileStream fs = null;

        // Try to access a file
        try {
            fs = new FileStream("SomeFile", FileMode.OpenOrCreate);
        }
        catch (Exception) {
        }

        // Display what we sucessfully did.
        Console.WriteLine(s + " test: "
            + ((fs != null) ? "FileOpen" : "FileNOTOpened"));

        // If we opened the file, close it and delete it
        if (fs != null) {
            fs.Close();
            File.Delete("SomeFile");
        }
    }

}

```

### Test Case 73.1

Sample calling native code.

```

using System;
using System.Reflection;
using System.Runtime.InteropServices;
using System.Security.Permissions;

[assembly:AssemblyKeyFile("../keypair.dat")]

public class LibWrap {
    // C# doesn't support varargs so all arguments must be explicitly defined
}

```

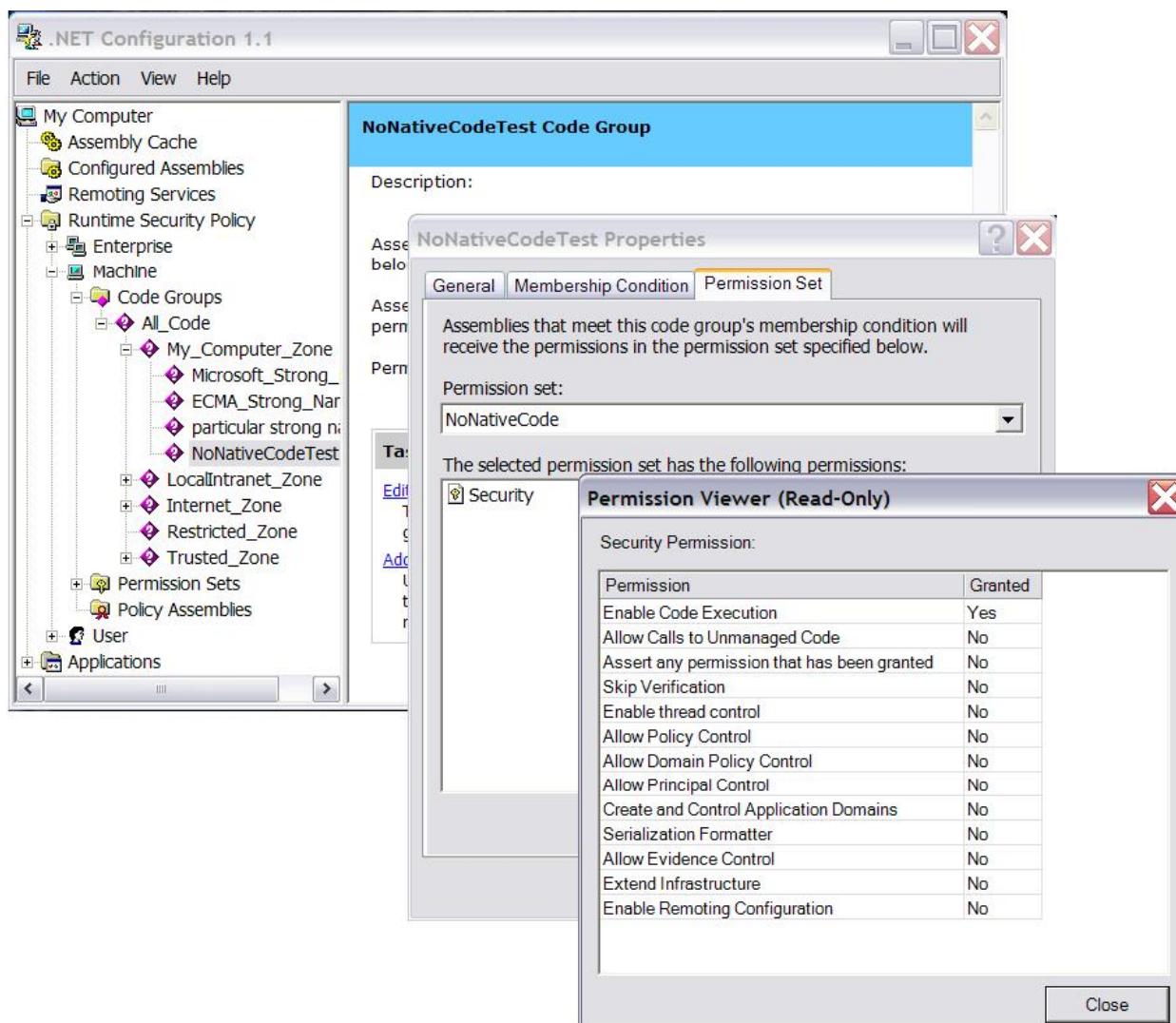
```
// CallingConvention.Cdecl must be used since the stack is cleaned up by the caller
// int printf( const char *format [, argument]... )

[ DllImport( "msvcrt.dll", CharSet=CharSet.Ansi,
CallingConvention=CallingConvention.Cdecl )]
public static extern int printf( String format, int i, double d );

[ DllImport( "msvcrt.dll", CharSet=CharSet.Ansi,
CallingConvention=CallingConvention.Cdecl )]
public static extern int printf( String format, int i, String s );

}

public class App {
    public static void Main() {
        LibWrap.printf( "\nPrint params: %i %f", 99, 99.99 );
        LibWrap.printf( "\nPrint params: %i %s", 99, "abcd" );
    }
}
```



```

Unhandled Exception: System.Security.SecurityException:
System.Security.Permissions.SecurityPermission at App.Main()

The state of the failed permission was:
<IPermission class="System.Security.Permissions.SecurityPermission, mscorel,
Version=1.0.5000,
0, Culture=neutral, PublicKeyToken=b77a5c561934e089" version="1" Flags="UnmanagedCode" />

```

## IBM WebSphere J2EE

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=70]. Code running in the application runtime environment is digitally signed by the originating party. When the classloader attempts to run a code module, it verifies that it was signed by a recognized party.	WebSphere Application Server does not appear to support code-signing of user JAR files, even though it is an integral part of the Java 2 specification.	<b>Not possible (1)</b> . WebSphere does not appear to honor security policy file declarations that require code to be signed. IBM provided no documentation on this functionality; several days of exploration proved fruitless.	IBM should immediately fix this problem. If our conclusion is incorrect, however, IBM should document correct procedures.
[id=71]. Policies limit code from accessing particular runtime resources such as network sockets and filesystem resources.	WebSphere uses policy files to control permissions for access by running Java code to potentially sensitive files and sockets I/O. Controls can be implemented at the server, application, and code module levels. The policy grammar is human-readable and provides a high degree of expressiveness.	<b>Wizard (4)</b> . WebSphere security policies are text files in specific locations. The files can be edited using a text editor or with policytool, a sparse-but-effective GUI tool.	IBM should provide more restrictive application security policies to enforce a "default-deny" posture with respect to file and socket I/O. In addition, more documentation and examples are needed.
[id=72]. Code runs with a subset of privileges by default. When additional privileges are required, mechanisms exist that allow the code to acquire enhanced privileges and call privileged code.	The Java 2 security model provides two methods by which code can acquire additional privileges: it can use JAAS to authenticate itself to a more-privileged loginContext, or call the AccessController class method doPrivileged() with a PrivilegedAction object passed as a parameter.	<b>Developer extends (3)</b> . Developers can implement either method using, in most cases, less than a dozen lines of code.	IBM should provide additional examples and documentation to more clearly explain how JAAS and privileged actions are used, and when they are appropriate.
[id=73]. The application runtime environment, by default, prohibits code modules from calling native code that is not managed by the application runtime environment.	Using a combination of Java 2 security policies and custom permission classes, developers can readily protect access to Java Native Interface (JNI) code.	<b>Developer extends (3)</b> . Implementation requires the developer to write a small amount of code and modify an existing server policy file.	Native code is rare in most Java applications. However, IBM should provide better documentation on how to secure JNI classes.

Best Practice Compliance scores, based on mean of 4 best practices: Linux 2.75, Unix 2.75

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=70.1]. Create a security policy that specifies that code from a particular source must be signed, and associate a recognized signer with the code source. Attempt to run code with the same package or class name, but without a signature; verify that it fails.	Implementation Complexity	<b>Large amount of code (1)</b> . Because WebSphere does not appear to support application code signing, a custom dynamic policy class would need to be written to substitute for IBM's own WSDynamicPolicy. This would have required far more time (and black magic) than any single developer would ever want to invest.
	Documentation and Examples	<b>Incorrect or insecure (1)</b> . IBM's documentation was non-existent.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . The evaluator has approximately two years of Java and J2EE experience.
	Time to Implement	<b>High (1)</b> . The developer spent nearly three days attempting to reverse-engineer this functionality, without success.

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=70.2]. Perform the same test as above, but with a fraudulent signature (signed by party with the same distinguished name, but a different private key).	Implementation Complexity	<b>Large amount of code (1)</b> . See test case 70.1
	Documentation and Examples	<b>Incorrect or insecure (1)</b> . See test case 70.1
	Implementor Competence	<b>Novice/intermediate (4)</b> . See test case 70.1
	Time to Implement	<b>High (1)</b> . See test case 70.1
[id=71.1]. Create a security policy file that limits file access by a given class to a single named local file location.	Implementation Complexity	<b>Wizard+ (4)</b> . Modifying security policies is done by changing a few lines in the server's <code>app.policy</code> file. To prevent a user-supplied application (EAR) from arbitrarily specifying additional privileges for itself, the <code>filter.policy</code> file must also be modified.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . The documentation was limited to one page of online content on IBM's website. While it clearly indicated what files to change and provided examples, it did not treat sufficiently seriously the notion that a user-supplied EAR can over-ride the default policies for certain types of permissions.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The evaluator has approximately two years of Java and J2EE experience.
	Time to Implement	<b>Medium (3)</b> . Modifying the default <code>app.policy</code> was trivial and took only a few minutes. However, because the documentation was unclear about the dependencies on <code>filter.policy</code> , it took about an hour to implement and test a work-around.
[id=72.1]. Create a less-privileged class that, by default, cannot access an object, file or method unless it possesses a specific privilege. Add supplemental code that allows the module to acquire the credentials needed to perform a privileged action and access the file.	Implementation Complexity	<b>Small amount of code (3)</b> . The <code>doPrivileged()</code> technique required augmenting an existing example with less than a dozen lines of code.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . IBM provides one example of using JAAS and <code>doPrivileged()</code> correctly. However, the inter-relationship with Java policy files could be better explained.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has approximately two years of Java and J2EE experience.
	Time to Implement	<b>Medium (3)</b> . The actual implementation took less than an hour, although thorough verification of the results took a bit longer.
[id=73.1]. Create a code module that attempts to access a native library or DLL present in the parent operating system, but is prohibited from doing so by the security policy. Create a second module that, by policy, is allowed to do so.	Implementation Complexity	<b>Small amount of code (3)</b> . The developer created a new customer Permission class, which the native code object checks for during instantiation. A Grant statement in the server's <code>app.policy</code> file grants the code the ability to possess the permission.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . IBM provides information on security policies generally, but none on creating custom permissions — and no documentation on how policies might apply to native code.
	Implementor Competence	<b>Novice (5)</b> . The developer has extensive Java experience, but has never written a custom permission class before.
	Time to Implement	<b>Medium to High (2)</b> . Although the finished implementation was quite simple, it took several hours to prototype the customer permission class and test the effectiveness of the policy change.

**Ease of Securing scores**, based on mean of 5 test cases: Linux 2.55, Unix 2.55

## Notes

### Test Case 70.1

The Java 2 security specification supports code-signing as a criterion for granting privileges. For example, the following lines in the Java runtime environment's `JRE_HOME/lib/security/java.policy` file grant classes in the `foo.jar` privileges to read the `java.home` system property, but only if the JAR was signed by a key with alias `mykey`:

```
keystore "file:${java.home}/lib/security/trustedsigningkeys.jks";
grant codeBase "file:/home/ajaquith/projects/jarsigner/foo.jar", signedBy "mykey" {
    permission java.util.PropertyPermission "java.home", "read";
};
```

The `keystore` entry points to the location where certificates of trusted signers are kept.

To test this functionality, we created a class that accessed external files and granted it the privilege to do so in the server's application security policy file (`app.policy`). We included a `keystore` and `signedBy` entries to so that the jar's signature would be checked prior to granting file access.

Curiously, when we implemented the policy, WebSphere's classloader did not appear to respect the requirement to inspect the signatures. Here is an informational message from the log file:

```
[5/9/03 17:05:54:070 EDT] 5dd6b402 WSDynamicPoli < getPermissions()
[5/9/03 17:05:54:070 EDT] 5dd6b402 SecurityManag < SecurityManager()
[5/9/03 17:05:54:070 EDT] 5dd6b402 SecurityCompo I SECJ0132I: Java 2 Security is enabled
[5/9/03 17:05:54:080 EDT] 5dd6b402 ApplicationPa W SECJ0168W: Keystore
file:${was.install.root}
${{/}config${{/}cells${{/}codekeys.jks of type jks is being ignored}
```

IBM provides zero documentation on how to use signed code in WebSphere, but that didn't stop us from trying to figure it out anyway. After nearly three full days of trying every imaginable combination of syntaxes, policy files, and debug tracing, we still couldn't make it work. We searched IBM's documentation (several thousand pages), Googled and UseNetted — still nothing. We even decompiled the "dynamic policy" classes (in package `com.ibm.ws.security.policy`) and found the line that produced the error message. But short of attaching a debugger to the application server (or obtaining clear documentation, whichever), we concluded that code-signing was either an incredibly well-kept secret — or just not supported.

If true, this is rather shocking stuff. A platform that claims to implement Java 2 platform security should *just do it*; that WebSphere appears to be selective about which aspects it supports is unreasonable, and borderline deceptive.

### Test Case 71.1

WebSphere documentation states that it implements the Java 2 platform security specification, with some product-specific customizations. The base permissions used by the application server itself are specified by the files `WAS_HOME/java/jre/lib/security/java.policy` and

`WAS_HOME/properties/server.policy`. These policy files are necessarily fairly permissive, to enable the application server to run.

Global security policies for applications are set in

`WAS_HOME/config/cells/cell_name/nodes/node_name/app.policy`. The base policy is fairly restrictive. For example, here is the default policy applied to web modules (*e.g.*, servlets, JSPs and) as defined in `app.policy`:

```
grant codeBase "file:${webComponent}" {
    permission java.io.FilePermission "${was.module.path}${/}-", "read, write";
    permission java.lang.RuntimePermission "loadLibrary.*";
    permission java.lang.RuntimePermission "queuePrintJob";
    permission java.net.SocketPermission "*", "connect";
    permission java.util.PropertyPermission "*", "read";
};
```

Among other things, this policy declaration allows all web modules to create socket connections to any hosts, a policy that in our view is overly permissive. The policy also permits web modules to read or write files, but only if those file are inside the applications' web root. This is good. Commenting out the `java.io.FilePermission` line has the effect of prohibiting *all* file I/O by all web applications, without impacting the application server runtime's abilities to perform its own file I/O operations.

Here is a revised `app.policy` file that explicitly denies all file I/O, except for the `BestPracticesWeb.war` module, which is allowed to access only the `userpreferences.properties` file in its own `/WEB-INF` directory:

```
grant codeBase "file:${webComponent}" {
    permission java.lang.RuntimePermission "loadLibrary.*";
    permission java.lang.RuntimePermission "queuePrintJob";
    permission java.net.SocketPermission "*", "connect";
    permission java.util.PropertyPermission "*", "read";
};

grant codeBase "file:BestPracticesWeb.war" {
    permission java.io.FilePermission
    "${was.module.path}${/}WEB-INF${/}userpreferences.properties",
    "read, write";
};
```

An attempt by a web application to access the `secretPasswords.properties` file produced a server console error. Here is the trace log:

```
[5/1/03 18:37:13:397 EDT] 26a6f8bf SecurityManag W SECJ0314W: Current Java 2
Security policy reported a potential violation of Java 2 Security Permission. Please
refer to
Problem Determination Guide for further information.

Permission:
```

```

D:\ibm-websphere-5.0\installedApps\ARJ-HOOVER\BestPractices.ear\BestPracticesWeb.war\
WEB-INF\secretpasswords.properties : access denied (java.io.FilePermission
D:\ibm-web
sphere-5.0\installedApps\ARJ-HOOVER\BestPractices.ear\BestPracticesWeb.war\WEB-INF\se
cretPasswords.properties read)

Code:
com.atstake.bestpractices.FilePermissionTester in
{file:/D:/ibm-websphere-5.0/installedApps/ARJ-HOOVER/BestPractices.ear/BestPracticesWe
b.war/WEB-INF/classes/}

Stack Trace:
java.security.AccessControlException: access denied (java.io.FilePermission
D:\ibm-websphere-5.0\installedApps\ARJ-HOOVER\BestPractices.ear\BestPracticesWeb.war\WEB-IN
F\secretpasswords.properties read)
at
java.security.AccessControlContext.checkPermission(AccessControlContext.java(Compiled
Code))
at java.security.AccessController.checkPermission(AccessController.java(Compiled
Code))
...

```

Prohibiting all file I/O in the manner described would, at first glance, appear to represent a form of "default-deny" policy. However, that is not the end of the story. As mentioned in the documentation, developers and application assemblers can optionally embed a `was.policy` file in the application EAR that causes WebSphere to add other permissions. For example, the following `was.policy` file permits the application to access the `secretpasswords.properties` file we forbade earlier:

```

grant codeBase "file:${webComponent}" {
    permission java.io.FilePermission
    "${was.module.path}${/}WEB-INF${/}secretpasswords.properties",
    "read, write";
}

```

While this is mentioned in the documentation, it is treated as a feature. We believe it is a potential weakness. For example, it would be trivial to change the target to a wildcard ("\*") and therefore allow the application to access any file in the server's filesystem that the application server process was entitled to. This would include, at a minimum, all of the J2EE applications installed on the WebSphere server, and all of its configuration files.

Fortunately, to protect against potentially dangerous permission grants in application `was.policy` files, WebSphere provides a `filter.policy` file which will cause particular permissions to always be refused for an application. The default settings are basic but effective:

```

//
// Dynamic Policy - Filter Templates
//
// Permissions defined in this file is going to be filtered out in app.policy and
was.policy.

```

```

// The security runtime will attempt to remove permissions defined in "filterMask" from
// application Java 2 Security policy during application start. However, this has some
// limitations, for example, if the application is grant permission to read and write
// to all Java system properties, if the following is defined in "filterMask":
//
//     java.util.PropertyPermission "someproperty", "write";
//
// The security runtime would not remove the permission to read and write to all Java
// system properties, instead it will log warning during application deployment and
// application start.
//
// The security runtime will deny application the permissions defined in
// "runtimeFilterMask",
// no matter what permissions are granted to the application. Basically, all application
// threads
// (non-system threads) will be denied permissions defined in "runtimeFilterMask". Be
// extreme
// careful of what permissions that defined in "runtimeFilterMask", any other
// permissions could
// cause applications fail to run.
//
// NOTE: Please do not specify permission java.lang.RuntimePermission "getClassLoader"
// in
//       the "runtimeFilterMask", it is not supported at this time.
//

filterMask {
    permission java.lang.RuntimePermission "exitVM";
    permission java.lang.RuntimePermission "setSecurityManager";
    permission java.security.SecurityPermission "setPolicy";
    permission java.io.FilePermission "*", "read, write";
};

runtimeFilterMask {
    permission java.lang.RuntimePermission "exitVM";
    permission java.lang.RuntimePermission "setSecurityManager";
    permission java.security.SecurityPermission "setPolicy";
    permission javax.security.auth.AuthPermission "setLoginConfiguration";
};

```

We added the explicit (if rather blunt-instrumented) line `permission java.io.FilePermission "*", "read, write";` to eliminate access to `secretPasswords.properties`.

See also: WebSphere InfoCenter

[http://publib7b.boulder.ibm.com/wasinfo1/en/info/aes/ae/tsec\\_dynamic.html](http://publib7b.boulder.ibm.com/wasinfo1/en/info/aes/ae/tsec_dynamic.html)

### Test Case 72.1

The goal of this test case was to show how code permissions can change depending on the runtime context. In this case, we re-used the native code and custom permission example from test case 73.1 (below). Specifically, we wanted to show that a servlet running in the web tier could access native code, but only by

calling another code module that was allowed to run with elevated privileges.

The Java 2 security normally performs access control checks based on the privilege of the calling code. So if one module calls another, the privileges that the second one can exercise depend entirely on the first. If the first module isn't allowed to access the file system, than the second one won't be, either. The Java 2 `AccessController` class — a core part of the runtime environment — enforces this through a process known as *stack-walking*.

However, in many cases this behavior is not desirable because the developer would like to create more privileged modules that can be called without needing to loosen all of the calling applications' security policies. To deal with this scenario, Java provides an `AccessController` method called `doPrivileged()` method which, in essence, interrupts the normal security checks. The `doPrivileged()` accepts an object which implements the `PrivilegedAction` interface. The object's `run()` method executes with the privileges granted to that code, not those of the caller.

To implement the solution, we modified the JNI object `HelloWorld` created in test case 73.1. The un-modified version of the `HelloWorld` class used the `AccessController.checkPermission()` method to determine whether the caller has sufficient privileges to access our sample native code. For this test case, we modified the class so that it calls `checkPrivilege()` inside a `doPrivileged()` call instead. This means that the check is made using the privileges granted to the `HelloWorld` base, not those of the caller. Thus, if the `HelloWorld` is deployed to a codebase location that is granted the required privilege in the policy file, it will always allow access, regardless of what class calls it or what privileges callers might possess themselves. This is the behaviour we want.

Here is the modified `HelloWorld` class, complete with a few `System.out` calls for debugging:

```
import com.atstake.bestpractices.jni.NativeCodePermission;
import java.security.AccessController;
import java.security.PrivilegedAction;

public class HelloWorld {

    public static final String PERMISSION = "HelloWorld";

    public static final HelloWorld INSTANCE = new HelloWorld();

    private HelloWorld() {
        System.out.println("-- Entered HelloWorld private constructor --");
        AccessController.doPrivileged( new PrivilegedAction() {
            public Object run() {
                System.out.println("-- Entered HelloWorld doPrivileged block--");
                NativeCodePermission permission = new NativeCodePermission(PERMISSION);
                System.out.println("-- Created NativeCodePermission; trying checkPermission() --");
                try {
                    AccessController.checkPermission(permission);
                    System.out.println("-- Success; returning null --");
                }
                catch (SecurityException e) {
                    System.out.println("The caller has not been granted permission NativeCodePermission ");
                }
            }
        });
    }
}
```

```

        + PERMISSION + ".");
    }
    return null;
}
});
System.out.println("-- Exiting HelloWorld private constructor --");
}

public native void displayHelloWorld();

static {
    System.loadLibrary("hello");
}
}

```

To test the sample solution, we first copied the `hello-jni.jar` file to `WAS_HOME/lib`, which by default has a fairly permissive security policy. This codebase is granted `NativeCodePermission`.

Next, we opened the server's application policy file (`WAS_HOME/config/cells/ARJ-HOOVER/nodes/ARJ-HOOVER/app.policy`), and removed all privilege grants of `NativeCodePermission` to user application code. This meant no web application code possessed the permission.

Attempting to access an unprivileged servlet that instantiates `HelloWorld` produced this output in the server log, as expected:

```

[5/15/03 15:15:22:661 EDT] 2141db85 WebGroup      I SRVE0180I: [bestpractices]
  [/bestpractices] [Servlet.LOG]: NativeCodeServlet: init
[5/15/03 15:15:22:711 EDT] 2141db85 SystemOut      O -- Entered HelloWorld private
constructor --
[5/15/03 15:15:22:721 EDT] 2141db85 SystemOut      O -- Entered HelloWorld doPrivileged
block--
[5/15/03 15:15:22:721 EDT] 2141db85 SystemOut      O -- Created NativeCodePermission;
trying chec
  kPermission() --
[5/15/03 15:15:22:721 EDT] 2141db85 SystemOut      O -- Success; returning null --
[5/15/03 15:15:22:721 EDT] 2141db85 SystemOut      O -- Exiting HelloWorld private
constructor --

```

IBM's documentation correctly points out that `doPrivileged()` can be a dangerous technique when used incorrectly or abused. The solution example above provides a case in point: *any* web application code could have successfully instantiated `HelloWorld`. This could have been easily fixed with an additional `checkPermission()` call in the "non-privileged" portion of the `HelloWorld()` constructor that checks for a different custom Permission.

IBM provides only extremely limited documentation on use of the `doPrivileged()` method. We found just one example, the `ITSOBank` application.

### Test Case 73.1

The Java Native Interface (JNI) is the method by which Java applications execute code running in the underlying operating system. A full treatment of JNI is beyond the scope of this test analysis, but the key points are these. In order to create a Java class that calls native code, the developer:

- Opens an existing native source file (C, C++ *et al*) and adds `JNIEXPORT` and include references (for example, `#include <jni.h>`)
- Creates a Java class that contains stub methods for the native module
- Generates a header file based on the Java class using the `javah` tool. These are then referenced in the source
- Compiles the native source, using the `include` references. The resulting shared library or DLL can be called directly by the Java class

For this test case, we implemented a simple "Hello World" example from the GNUWin32 website (<http://www.xraylith.wisc.edu/~khan/software/gnu-win32>) and implemented it on a local WebSphere instance. The `HelloWorld` class contains a single public method `native void displayHelloWorld()`; that echoes the words `Java JNI` to the system console and prints the value of the local OS' `HOME` environment variable. We executed this test case entirely on Windows, although the solution works similarly on Linux and Solaris.

IBM does not provide documentation describing how Java platform security might be used to protect access to native code. To implement the solution, we consulted two books for patterns and ideas:

- Patrick Chan, *The Java Developer's Almanac 1.4, Volume 1 (4th edition)*
- Li Gong, *Inside Java 2 Platform Security*

As mentioned in the summary table for this test case, a small amount of development was required to implement native code protection. First, we created a custom `Permission` class that extends `java.security.BasicPermission`. The permission, when instantiated, takes a single parameter denoting the native code class we wish to allow access to (in this case, `HelloWorld`). The implementation is simple:

```
package com.atstake.bestpractices.jni;
import java.security.*;

public class NativeCodePermission extends BasicPermission {

    String className = null;

    public NativeCodePermission(String className) {
        super(className);
        this.className= className;
    }

    public NativeCodePermission(String className, String action) {
        super(className);
        this.className= className;
    }
}
```

```

    }

    public boolean implies(Permission permission) {
        if (!(permission instanceof NativeCodePermission)) {
            return false;
        }
        NativeCodePermission nativePerm = (NativeCodePermission) permission;
        return (this.className.equals(nativePerm.className));
    }

    public String getActions() {
        return "";
    }

    public boolean equals(Object obj) {
        if (!(obj instanceof NativeCodePermission)) {
            return false;
        }
        NativeCodePermission nativePerm = (NativeCodePermission) obj;
        return (this.className.equals(nativePerm.className));
    }

}

```

In calling classes, the permission is created by instantiating `NativeCodePermission` and passing the name of the protected native class as a parameter:

```
NativeCodePermission permission = new NativeCodePermission("com.atstake.Foo")
```

Next, we modified the native code class `HelloWorld.java` to check for the permission `NativeCodePermission HelloWorld`. The code was added so that the check is performed when the object is instantiated:

```

import com.atstake.bestpractices.jni.NativeCodePermission;
import java.security.AccessController;

public class HelloWorld {

    public static final String PERMISSION = "HelloWorld";

    public static final HelloWorld INSTANCE = new HelloWorld();

    private HelloWorld() {
        NativeCodePermission permission = new NativeCodePermission(PERMISSION);
        try {
            AccessController.checkPermission(permission);
        }
        catch (SecurityException e) {
            System.out.println("The caller has not been granted permission
NativeCodePermission "
                + PERMISSION + ".");
        }
    }
}

```

```

    }

    public native void displayHelloWorld();

    static {
        System.loadLibrary( "hello" );
    }
}

```

Note that, to keep things simple, there is no enclosing package name for the `HelloWorld` class; otherwise the permission would have been `com.foo.HelloWorld` or similar. Note also the use of a *private constructor* to ensure that the native class will always be a singleton (see Joshua Bloch's excellent *Effective Java*, Item 2). This isn't strictly necessary, but we did it in this example to facilitate a load-once type factory pattern similar to the way a database pool or JCA resource operates.

Finally, to allow a caller to instantiate `HelloWorld`, the server's application policy file needs to grant `NativeCodePermission` to `HelloWorld`. In the majority of cases, the file one should modify is the global application security policy file

`WAS_HOME/config/cells/cell_name/nodes/node_name/app.policy`. Using a text editor, we granted all web applications `NativeCodePermission` as follows:

```

grant codeBase "file:${webComponent}" {
    permission java.lang.RuntimePermission "loadLibrary.*";
    permission java.lang.RuntimePermission "queuePrintJob";
    permission java.net.SocketPermission "*", "connect";
    permission java.util.PropertyPermission "*", "read";
    permission com.atstake.bestpractices.jni.NativeCodePermission "HelloWorld";
};

```

We also created a small JAR that contained the compiled `NativeCodePermission` class, and added it to the server's `WAS_HOME/lib` directory so that the permission could be resolved by the server runtime environment.

To test the (now-protected) native code, we created a servlet `NativeCodeServlet` that instantiates the `HelloWorld` class and attempts to execute the `displayHelloWorld` method. Here is the relevant try-catch block from the servlet:

```

try {
    resp.setContentType( "text/html" );
    out = resp.getWriter();
    out.println("Calling HelloWorld...");

    );

    HelloWorld hello = HelloWorld.INSTANCE;
    hello.displayHelloWorld();
    out.println("
```

We successfully called JNI class HelloWorld.

```
" );
} catch (Exception e) {
    System.out.println(e.getMessage());
}
```

When the `app.policy` file contains the `NativeCodePermission` grant, the servlet functions normally. When it doesn't, the following stack trace echoes to the console, as expected:

```
[5/14/03 1:36:05:056 EDT] 6714d2d8 WSDynamicPoli < getPermissions()
[5/14/03 1:36:05:546 EDT] 6714d2d8 WebGroup      E SRVE0026E:
[Servlet Error]-[]: java.lang.ExceptionInInitializerError:
java.security.AccessControlException:
    access denied (com.atstake.bestpractices.jni.NativeCodePermission HelloWorld)
    at
java.security.AccessControlContext.checkPermission(AccessControlContext.java(Compiled
Code))
    at java.security.AccessController.checkPermission(AccessController.java(Compiled
Code))
    at HelloWorld.<init>(HelloWorld.java:10)
    at HelloWorld.<clinit>(HelloWorld.java:6)
    at
com.atstake.bestpractices.NativeCodeServlet.sayHelloNatively(NativeCodeServlet.java:39)
    at com.atstake.bestpractices.NativeCodeServlet doGet(NativeCodeServlet.java:22)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:740)
```

See also:

IBM DeveloperWorks <https://www6.software.ibm.com/reg/devworks/dw-javajni-i>

## 4.2.8.2 Runtime Account Privileges

---

### Runtime Account Privileges

Runtime account privileges are privileges available to the user ID used during normal operation of the service. This is weighted medium for all application scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Runtime Account Privileges topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=61]. Services should run as non-privileged users, with only the minimum privileges required for proper functionality	.NET provides the option of running application servers under the context of non-administrative users. It also provides the option of configuring particular applications to run as different users.	<b>Transparent (5)</b> . The default behavior for the web server was to run as a non privileged process. The web server was running as the NETWORK_SERVICE user. If separate user privileges are desired for different applications they can be configured with by using a set of dialog boxes.	The default of NETWORK_SERVICE is appropriate for most users. Users should consider running individual applications under their own user context.

**Best Practice Compliance score**, based on mean of 1 best practice: 5

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=61.2]. Configure application server processes to use non-privileged accounts (not the OS administrator account).	Implementation Complexity	<b>Wizard+ (4)</b> . The default of NETWORK_SERVICE is appropriate for most users. In order to run an application under a different user a new application pool must be created. Next a new user account must be created using the platform account tool which is wizard driven. Next a new application pool needs to be created in the web server. This is done with a wizard. Finally the application pool needs to be configured to use the user account previously created. This is done through a dialog box.
	Documentation and Examples	<b>Suitable (4)</b> . Documentation was found quickly after consulting the web server help system. There were step by step instructions of how to accomplish the task.
	Implementor Competence	<b>Novice (5)</b> . No developer skills were required.
	Time to Implement	<b>Low (5)</b> . It took approximately 10 minutes to read the documentation and make the appropriate configuration changes.

Ease of Securing score, based on mean of 1 test case: 4.5

### Notes

#### Test Case 61.2

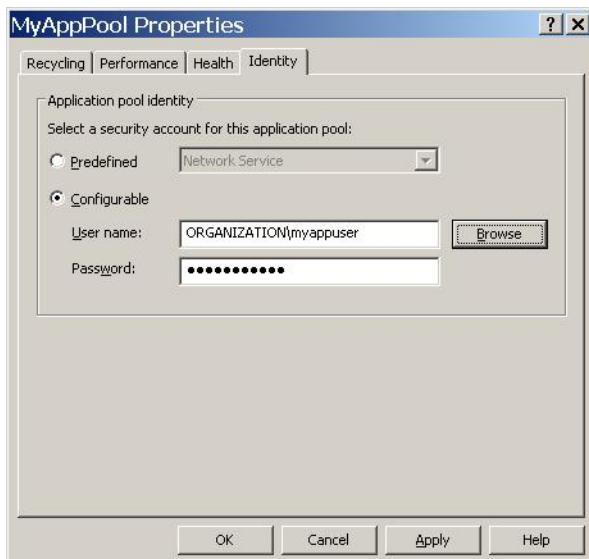
Application identity configuration:



Create new user:



Assign user identity to application pool:



Configure application to use application pool with new identity:



## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
<i>Linux. [id=61]. Services should run as non-privileged users, with only the minimum privileges required for proper functionality</i>	WebSphere provides the option of running application servers under the context of non-administrative users.	<b>Wizard (4)</b> . WebSphere application servers can be easily configured to run as non-privileged users.	Configure WebSphere application servers to execute under the context of individual, non-administrative users.
<i>Unix. [id=61]. Services should run as non-privileged users, with only the minimum privileges required for proper functionality</i>	WebSphere provides the option of running application servers under the context of non-administrative users.	<b>Wizard (4)</b> . WebSphere application servers can be easily configured to run as non-privileged users.	Configure WebSphere application servers to execute under the context of individual, non-administrative users.

Best Practice Compliance scores, based on mean of 1 best practice: Linux 4, Unix 4

## Level of Effort Analysis

Test Case	Criteria	Rating
<i>Linux. [id=61.2]. Configure application server processes to use non-privileged accounts (not the OS administrator account).</i>	Implementation Complexity	<b>Small amount of code (3)</b> . Several steps are required to configured WebSphere application servers to run as non-privileged users. Operating system users and groups must be created, and file permissions must be modified. Additionally, the specific server must be configured within the Administrative Console to execute under non-root privileges.
	Documentation and Examples	<b>Adequate (3)</b> . Vendor documentation exists for this configuration, but it is not verbose or detailed with configuration examples.
	Implementor Competence	<b>Intermediate (3)</b> . The administrator is very familiar with Unix operating systems and security and web administration but is unfamiliar with WebSphere administration.
	Time to Implement	<b>Medium (3)</b> . The operating and file system changes take a few minutes to manage, and the Administrative Console changes are straightforward.
<i>Unix. [id=61.2]. Configure application server processes to use non-privileged accounts (not the OS administrator account).</i>	Implementation Complexity	<b>Small amount of code (3)</b> . Several steps are required to configured WebSphere application servers to run as non-privileged users. Operating system users and groups must be created, and file permissions must be modified. Additionally, the specific server must be configured within the Administrative Console to execute under non-root privileges.
	Documentation and Examples	<b>Adequate (3)</b> . Vendor documentation exists for this configuration, but it is not verbose or detailed with configuration examples.
	Implementor Competence	<b>Intermediate (3)</b> . The administrator is very familiar with Unix operating systems and security and web administration but is unfamiliar with WebSphere administration.
	Time to Implement	<b>Medium (3)</b> . The operating and file system changes take a few minutes to manage, and the Administrative Console changes are straightforward.

**Ease of Securing scores**, based on mean of 1 test case: Linux 3, Unix 3

## Notes

### Linux Test Case 61.2

The online IBM document, *Running servers as non-root using the console*, provides details for running WebSphere servers under the context of non-administrative users.

### Unix Test Case 61.2

The online IBM document, *Running servers as non-root using the console*, provides details for running WebSphere servers under the context of non-administrative users.

## 4.2.9 Web Services

---

### Web Services

Web services provides an infrastructure for parties (computer-to-computer) to communicate with each other securely over an internal network for services in a platform-independent and language-neutral manner. Key components of the platform includes features such as security, distributed transaction management and connection pool management.

The Web Services area of analysis includes these topics:

- Credentials Mapping
- SOAP Router Data Validation

The table below describes the topics at a high level, and lists prevailing best practices. The right-hand columns show the weights @stake assigned for each of the scenarios.

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Credentials Mapping</b> . Credentials mapping maps the credentials presented in a SOAP header to identities used within a web service. It is ranked high for the web service scenario and not applicable for the other two scenarios.	User identities expressed in the SOAP header should be bound to identity store roles that exist in the application.	0	3	0
<b>SOAP Router Data Validation</b> . SOAP router data validation functionality can perform input validation for a web services. It is ranked high for the web service scenario and not applicable for the other two scenarios.	The SOAP router should provide type checking for requests submitted by web services clients. Unencrypted field contents need to be validated in a manner similar to web form submitted data, i.e. range and boundary checking.	0	3	0

Scenario weights range from 1 (relatively unimportant) to 3 (extremely important). A scenario weight of zero means that the topic does not apply.

### Findings

The table below summarizes @stake's quantitative findings. For each analysis topic, the two high-level metrics (Best Practice Compliance and Ease of Securing) are shown. These are the arithmetic means of the underlying best practice and test case scores. The Ease of Securing metric is further broken down into four sub-metrics.

Topic	Platform	Best Practice Compliance	Ease of Securing				
			Overall	Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Credentials Mapping <i>1 best practice, 1 test case</i>	.NET Windows	<b>3</b>	<b>4</b>	3	4	5	4
	WebSphere Linux	<b>3</b>	<b>1.75</b>	2	1	3	1
	WebSphere Unix	<b>3</b>	<b>1.75</b>	2	1	3	1
SOAP Router Data Validation <i>2 best practices, 2 test cases</i>	.NET Windows	<b>4</b>	<b>4.25</b>	4.5	3.5	4	5
	WebSphere Linux	<b>4</b>	<b>4.13</b>	4	4.5	4.5	3.5
	WebSphere Unix	<b>4</b>	<b>4.13</b>	4	4.5	4.5	3.5

This area of analysis covered security areas critical to implementing secure web services. These included the platform's ability to map the credentials presented in a SOAP header to identities used within a web service and the use of input filtering in the SOAP router. Both vendors score similarly for the test cases under this area of analysis.

The individual sections for each analysis topic describe @stake's findings in further detail.

## 4.2.9.1 Credentials Mapping

---

### Credentials Mapping

Credentials mapping maps the credentials presented in a SOAP header to identities used within a web service. It is ranked high for the web service scenario and not applicable for the other two scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 0
- Web Service: 3
- Intranet: 0

The sections that follow contain the results of @stake's analysis of the Credentials Mapping topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=42]. User identities expressed in the SOAP header should be bound to identity store roles that exist in the application.		<b>Developer extends (3)</b> . There is no server configuration setting or wizard to enable SOAP header mapping to a particular authentication store. The developer must extend a SOAP header class to add authentication information. There was a good example in the web services sample code that provides this functionality.	Following the sample code makes it easy to add SOAP header authentication to any web service.

**Best Practice Compliance score**, based on mean of 1 best practice: 3

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=42.1]. Configure the container to bind web service user identities to the authentication store.	Implementation Complexity	<b>Small amount of code (3)</b> . A small amount of code is required to extend the SOAPHeader class to add the authentication information. Methods need to be added to actually perform the authentication against the identity store.
	Documentation and Examples	<b>Suitable (4)</b> . The SOAP header documentation is clear about how to extend the header. The example used is extending the SOAP header to add authentication information. It was easy to understand how to perform this developer task.
	Implementor Competence	<b>Novice (5)</b> . The amount of code is small and there is a clear example. A novice developer will be able to implement the code.
	Time to Implement	<b>Low to Medium (4)</b> . The amount of code is small. Code needs to be implemented on both the client and server side to support the SOAP header authentication function.

**Ease of Securing score**, based on mean of 1 test case: 4

### Notes

#### Test Case 42.1

Derive a class that inherits from SOAPHeader class.

```
using System.Web.Services;
using System.Web.Services.Protocols;

// AuthHeader class extends from SoapHeader
public class AuthHeader : SoapHeader {
    public string Username;
    public string Password;
}

public class HeaderService : WebService {
    public AuthHeader sHeader;
    ...
}
```

Use the class to extract username and password from header and authenticate against a database.

```
using System;
using System.Web.Services;
using System.Web.Services.Protocols;

// Note the namespace has to be different from the one used
// on the proxy dll or we get errors about AuthHeader being
// defined in multiple places.
namespace SoapHeadersCS {
```

```
// AuthHeader class extends from SoapHeader
public class AuthHeaderCS : SoapHeader {
    public string Username;
    public string Password;
}

[WebService(Description="Simple sample to demonstrate use of SOAP Headers")]
public class HeaderService {

    public AuthHeaderCS sHeader;

    [WebMethod(Description="This method requires a custom soap header set by
the caller")]
    [SoapHeader("sHeader")]
    public string SecureMethod() {

        if (sHeader == null)
            return "ERROR: Please supply credentials";

        string usr = sHeader.Username;
        string pwd = sHeader.Password;

        if (AuthenticateUser(usr, pwd)) {
            return "SUCCESS: " + usr + "," + pwd;
        }
        else {
            return "ERROR: Could not authenticate";
        }
    }

    private bool AuthenticateUser(string usr, string pwd) {

        if ((usr != null)&&(pwd != null)) {
            // could query a database here for credentials...
            return true;
        }
        return false;
    }
}
```

## IBM WebSphere J2EE

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=42]. User identities expressed in the SOAP header should be bound to identity store roles that exist in the application.	WebSphere's SOAP router does not associate SOAP message header credentials with WebSphere container (JAAS) Principals. The Network Deployment Edition of WebSphere AS, which we did not test, apparently has a "Web Services Gateway" that does. For the Standard Edition of WAS, the developer must implement custom code.	<b>Developer extends (3)</b> . The Apache SOAP router used by WebSphere Studio-generated web services can be extended to use SOAP header credentials to perform a server-side login to WebSphere.	The upcoming J2EE 1.4 specification requires compliant containers to implement a <web-service> deployment descriptor. IBM will likely release an upgrade to WebSphere Studio in the next two quarters that includes support for J2EE 1.4. If users need solutions now, custom code is the answer.

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 3, Unix 3

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=42.1]. Configure the container to bind web service user identities to the authentication store.	Implementation Complexity	<b>Medium amount of code (2)</b> . Two classes must be extended or created, not including client-side code that places credentials in the SOAP header.
	Documentation and Examples	<b>Incorrect or insecure (1)</b> . Sufficient documentation exists in several places to give this reviewer enough ideas of how to implement a solution, but there is no consolidated set of documentation.
	Implementor Competence	<b>Intermediate (3)</b> . This reviewer has less than two years of Java and J2EE experience, but is familiar with enterprise security architectures.
	Time to Implement	<b>High (1)</b> . In the hands of an experienced J2EE developer, implementation would require about a day of effort. A less experienced developer would require more.

**Ease of Securing scores**, based on mean of 1 test case: Linux 1.75, Unix 1.75

## Notes

### Test Case 42.1

The evaluator expended the maximum time allowed attempting to put together a solution, then stopped. That said, the steps to implement are straightforward.

First, at the time of interface generation, the SOAP wizard accepts a parameter that specifies the provider to be used when calling a particular web service. This value is stored in the web context root in the file `dds.xml`. For example, for an EJB session bean called "Hello" that implements a web service interface, the entry looks like this:

```
<isd:provider type="com.ibm.soap.providers.WASStatelessEJBProvider"
    scope="Application" methods="getTestInput getTestInput getGoodbyeMessage
    getTestInput getTestInput getTestInput getTestInput getTestInput getTestInput
    getTestInput getTestInput getHelloMessage getTestInput">
```

```

<isd:option key="JNDIName" value="ejb/com/atstake/bestpractices/HelloHome" />
<isd:option key="ContextProviderURL" value="iiop://localhost:2809" />
<isd:option key="FullHomeInterfaceName" value="com.atstake.bestpractices.HelloHome" />
<isd:option key="FullContextFactoryName"
            value="com.ibm.websphere.naming.WsnInitialContextFactory" />
</isd:provider>
```

The `<type>` attribute would be changed to the name of a substitute soap provider similar to `WASStatelessEJBProvider`. The provider would implement Apache's `Provider` interface, and would call the SOAP message `Header` object's `getAttribute` accessors to retrieve the user's credentials.

After retrieving the credentials, the class would authenticate to the WebSphere Application Server using the credentials obtained from the SOAP message. Here is a sample (working) code block we implemented that does this:

```

import javax.security.auth.login.LoginContext;
import javax.security.auth.Subject;
import com.ibm.websphere.security.auth.callbackWSCallbackHandlerImpl;
import com.ibm.websphere.security.auth.WSSubject;
...
LoginContext lc = null;
Subject serverSubject;
String loginuid = soapPrincipal; //From the SOAP header
String loginrealm = "websphere";
String loginpwd = soapCredential; //From the SOAP header
try {
    System.out.println("Retrieving Subject on behalf of SOAP header
principal...");
    // creating the login context using WebSphere's callback handler
    lc = new LoginContext("WSLogin",
        new WSCallbackHandlerImpl(loginuid,loginrealm,loginpwd));
    lc.login();
    serverSubject = lc.getSubject();
    System.out.print("JAAS Subject: " +
serverSubject.getPrincipals().toString());
    PrivilegedEjbCall action = new PrivilegedEjbCall ();
    WSSubject.doAs(serverSubject, action);
}
catch (javax.security.auth.login.LoginException le) {
    // insert error processing code
}
catch (SecurityException se) {
    // Insert error processing
}
```

The class `PrivilegedEjbCall` implements the `PrivilegedAction` Interface and calls the EJB with the SOAP Principal's privileges.

See also:

[WebSphere 5 Security RedBook \(ItsoBank PrivilegedTransfer servlet example\)](#)

<http://www-3.ibm.com/software/webservers/appserv/doc/v40/ae/infocenter/was/040804.html>

<http://ws.apache.org/soap/docs/apiDocs/org/apache/soap/Header.html>

## 4.2.9.2 SOAP Router Data Validation

---

### SOAP Router Data Validation

SOAP router data validation functionality can perform input validation for a web services. It is ranked high for the web service scenario and not applicable for the other two scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 0
- Web Service: 3
- Intranet: 0

The sections that follow contain the results of @stake's analysis of the SOAP Router Data Validation topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=40]. The SOAP router should provide type checking for requests submitted by web services clients.		<b>Transparent (5)</b> . Argument types are validated by the web services framework automatically before any custom code is executed. Positive Input validation can then be easily done through regular expression patterns.	No recommendation.
[id=41]. Unencrypted field contents need to be validated in a manner similar to web form submitted data, i.e. range and boundary checking.	The .NET platform provides an input data validation framework. Application developers are required to implement validation calls manually.	<b>Developer extends (3)</b> . Using the functionality exposed by the platform, implementing the test cases within this best practice was straightforward.	

**Best Practice Compliance score**, based on mean of 2 best practices: 4

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=40.1]. Ensure that non-conforming input data is rejected.	Implementation Complexity	<b>Wizard (5)</b> . No customization needed, this functionality is included out of the box.
	Documentation and Examples	<b>Suitable (4)</b> . Documentation is solid but could highlight security best practices more effectively.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive web application development experience but very little with web services.
	Time to Implement	<b>Low (5)</b> . No changes needed to be made.
[id=41.1]. Implement data validation routines that reject invalid data in web service input messages.	Implementation Complexity	<b>Wizard+ (4)</b> . Moderate amounts of code are required to implement input filtering.
	Documentation and Examples	<b>Adequate (3)</b> . No web service specific documentation available.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The test case implementer has wide ranging experience in developing Windows based web applications.
	Time to Implement	<b>Low (5)</b> . Implementing simple input filtering was accomplished in under five minutes.

**Ease of Securing score**, based on mean of 2 test cases: 4.25

### Notes

#### Test Case 40.1

This is part of the Web Services framework. It validates that user input is of the expected type for the method signature.

If a web service has the following signature:

```
<WebMethod()> Public Function OneIntArg(ByVal arg1 As Integer) As String
```

Calling it with:

<http://localhost/Negative/Positive.asmx/OneIntArg?arg1=Test>

will return the following:

```
System.ArgumentException: Cannot convert Test to System.Int32.
Parameter name: type ---> System.FormatException: Input string was not in a correct
format.
   at System.Number.ParseInt32(String s, NumberStyles style, NumberFormatInfo info)
   at System.String.System.IConvertible.ToInt32(IFormatProvider provider)
   at System.Convert.ChangeType(Object value, Type conversionType, IFormatProvider
provider)
   at System.Web.Services.Protocols.ScalarFormatter.FromString(String value, Type type)
```

```

--- End of inner exception stack trace ---
at System.Web.Services.Protocols.ScalarFormatter.FromString(String value, Type type)
at
System.Web.Services.Protocols.ValueCollectionParameterReader.Read(NameValueCollection
collection)
at System.Web.Services.Protocols.UrlParameterReader.Read(HttpContext request)
at System.Web.Services.Protocols.HttpServerProtocol.ReadParameters()
at System.Web.Services.Protocols.WebServiceHandler.Invoke()
at System.Web.Services.Protocols.WebServiceHandler.CoreProcessRequest()

```

It was not clear how to disable this extensive error logging as it represents potential information leakage.

#### Test Case 41.1

Web service methods can easily validate input in addition to the .NET Framework validating the argument types. This can be done with simple regular expressions in string arguments:

```

<WebMethod()> Public Function OneStringArg(ByVal arg1 As String) As String
    Dim reg As Regex
    reg = New Regex("((\\d{3})\\ ?)|(\\d{3}-))?\d{3}-\d{4}", RegexOptions.
        Compiled Or RegexOptions.Singleline Or RegexOptions.IgnoreCase)

    If Not reg.IsMatch(arg1) Then
        Return "Arg1 Invalid"
    End If

    OneStringArg = "Hello World" + arg1
End Function

```

The framework could be improved to make use of the Validator controls web forms use.

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=40]. The SOAP router should provide type checking for requests submitted by web services clients.	The web services router provided by WebSphere automatically performs basic type checking of user-submitted data so that the correct Java classes are cast.	<b>Transparent (5)</b> . Type checking for XML simple types are built into the Apache SOAP router used by the generated web services interfaces. The default mappings between the web service types and native Java classes can be tweaked, if needed. Complex types are also supported, although they must be composed of simple types.	The default mappings should work well for most developers

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=41]. Unencrypted field contents need to be validated in a manner similar to web form submitted data, i.e. range and boundary checking.	WebSphere Studio provides support for custom SOAP parameter validation through the use of pluggable 'deserializer' classes. These classes transform the SOAP parameters into Java-equivalent classes. To perform validation, developers can implement the deserializer interface as a custom class, and specify that SOAP bindings for a particular type (e.g., String) use this class.	<b>Developer extends (3)</b> . Custom deserializer classes can be used to implement any kind of validation that is desired. When generating web service interfaces, the developer only needs to specify the custom deserializer.	Because of rigorous type checking routines already present in the Apache SOAP router for most primitive classes, the only deserializer that would ever need to be written is for type String.

**Best Practice Compliance scores**, based on mean of 2 best practices: Linux 4, Unix 4

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=40.1]. Ensure that non-conforming input data is rejected.	Implementation Complexity	<b>Wizard (5)</b> . Type checking is built-in.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation clearly explains the default mappings between XSD type names and Java classes.
	Implementor Competence	<b>Novice (5)</b> . This evaluator has limited experience with web services. No special skills are needed to implement type checking.
	Time to Implement	<b>Low (5)</b> . Unless the developer wishes to use custom mapping classes, the defaults take precedence. There is no net additional time required on the part of the developer.
[id=41.1]. Implement data validation routines that reject invalid data in web service input messages.	Implementation Complexity	<b>Small amount of code (3)</b> . To implement a simple validator that would only allow a subset of "safe" ASCII characters, a small class file needs to be created.
	Documentation and Examples	<b>Suitable (4)</b> . Documentation for this functionality is discussed in the principal documentation we used (the online help), but only briefly. Very late in the process, the evaluator discovered a slightly dated (but detailed and lucid) example in a V4 Redbook.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The evaluator has less than two years of Java experience, with limited knowledge of web services. No special skills were needed other than basic knowledge on how to implement a RegEx filter and an Interface.
	Time to Implement	<b>Medium to High (2)</b> . Because the documentation was scant, the initial learning curve was steeper than it needed to be: about three hours. However, the marginal time required to implement additional, similar serializers would likely be less than 15 minutes.

**Ease of Securing scores**, based on mean of 2 test cases: Linux 4.13, Unix 4.13

## Notes

### Test Case 40.1

To test the SOAP router's type mapping capabilities, we modified our Hello EJB session bean by adding method signatures for each type we wanted to check, *e.g.*:

```
// Tests the type checking for the SOAP router
public java.lang.String getTestInput(java.lang.Boolean test) {
    return "You sent me " + test.getClass().toString() + " '" + test.toString() + "'";
}

public java.lang.String getTestInput(java.lang.Byte test) {
    return "You sent me " + test.getClass().toString() + " '" + test.toString() + "'";
}

public java.lang.String getTestInput(java.lang.Short test) {
    return "You sent me " + test.getClass().toString() + " '" + test.toString() + "'";
}

public java.lang.String getTestInput(java.lang.Integer test) {
    return "You sent me " + test.getClass().toString() + " '" + test.toString() + "'";
}

public java.lang.String getTestInput(java.lang.Long test) {
    return "You sent me " + test.getClass().toString() + " '" + test.toString() + "'";
}

public java.lang.String getTestInput(java.lang.Float test) {
    return "You sent me " + test.getClass().toString() + " '" + test.toString() + "'";
}

public java.lang.String getTestInput(java.lang.Double test) {
    return "You sent me " + test.getClass().toString() + " '" + test.toString() + "'";
}

public java.lang.String getTestInput(java.lang.String test) {
    return "You sent me " + test.getClass().toString() + " '" + test.toString() + "'";
}

public java.lang.String getTestInput(java.math.BigDecimal test) {
    return "You sent me " + test.getClass().toString() + " '" + test.toString() + "'";
}

public java.lang.String getTestInput(java.util.GregorianCalendar test) {
    return "You sent me " + test.getClass().toString() + " '" + test.toString() + "'";
}

public java.lang.String getTestInput(java.util.Date test) {
    return "You sent me " + test.getClass().toString() + " '" + test.toString() + "'";
}
```

The Java and SOAP mappings used were these:

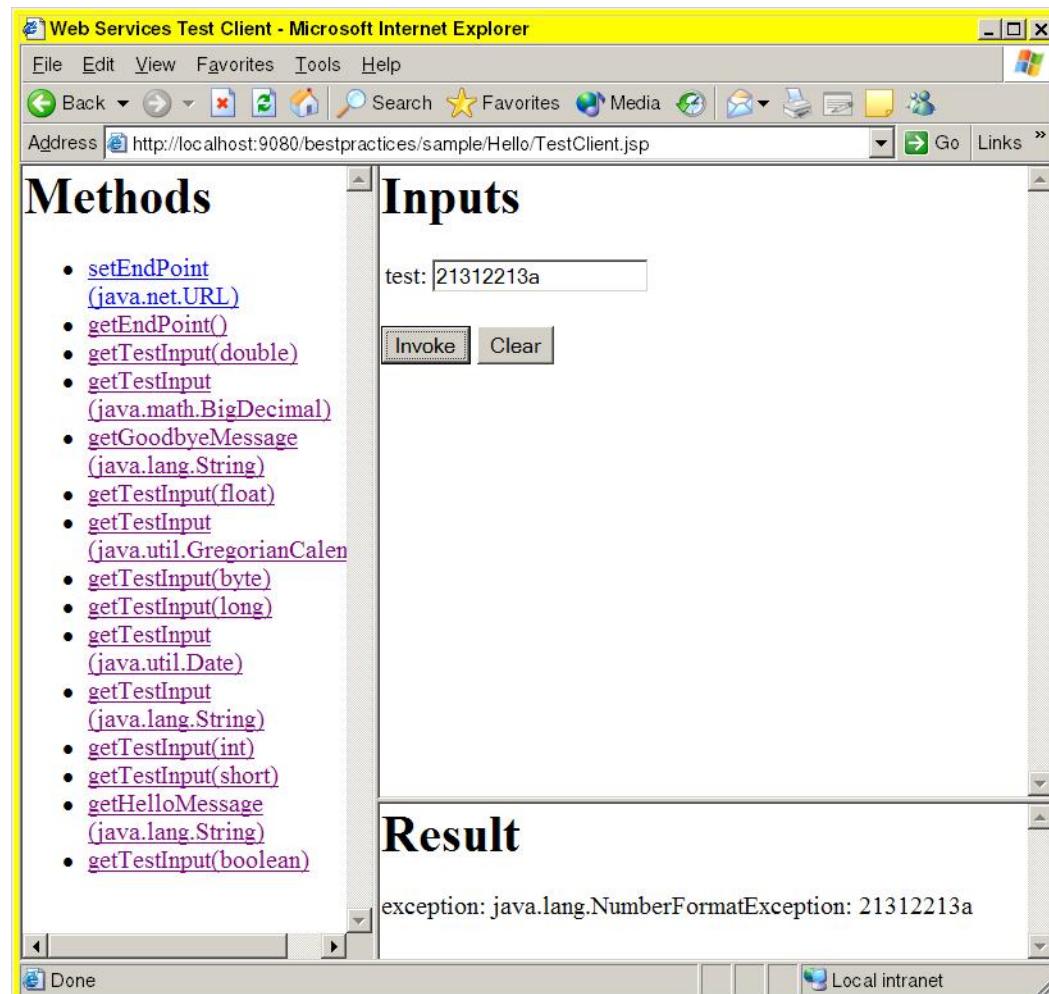
```
<format:typeMapping encoding="EJB" style="Java">
<format:typeMap formatType="java.lang.Boolean" typeName="xsd:boolean"/>
<format:typeMap formatType="java.lang.Float" typeName="xsd:float"/>
<format:typeMap formatType="java.lang.Double" typeName="xsd:double"/>
```

```

<format:typeMap formatType="java.util.Date" typeName="xsd:dateTime"/>
<format:typeMap formatType="java.util.GregorianCalendar" typeName="xsd:date"/>
<format:typeMap formatType="java.lang.Long" typeName="xsd:long"/>
<format:typeMap formatType="java.math.BigDecimal" typeName="xsd:decimal"/>
<format:typeMap formatType="java.lang.String" typeName="xsd:string"/>
<format:typeMap formatType="java.lang.Short" typeName="xsd:short"/>
<format:typeMap formatType="java.lang.Integer" typeName="xsd:int"/>
<format:typeMap formatType="java.lang.Byte" typeName="xsd:byte"/>
</format:typeMapping>

```

The JSP-based SOAP test client was used to run tests for each type. While a full "fuzzing" exercise was beyond the scope of this evaluation, the tests included negative testing (typing in string values in numeric fields, exceeding the bounds of allowed types) as well as positive testing. This screen shot shows an unsuccessful attempt to force a string value into a Long input parameter:



All negative tests passed without exception. With respect to positive testing, we could not seem to find a

value that would successfully map to `java.util.GregorianCalendar` (on the SOAP side, this was `xsd:date`), but that was probably an error on the part of this evaluator.

#### Test Case 41.1

by default, SOAP inputs are type-checked at runtime by the Apache SOAP router to ensure that they can be mapped to equivalent Java types. Invalid input for nearly all of the primitive types (*e.g.*, date, int, long, boolean) is caught automatically, and a SOAP fault is sent back to the caller. The one exception to this is String. Therefore, J2EE developers need only worry about validating Strings, but not others.

As discussed in the table above, Apache SOAP allows developers to specify custom deserializer (and serializer) classes for specified types. To ensure that an application detects 'illegal' strings (for example, one that contains JavaScript), the developer needs to:

- Create a deserializer class that conforms to Interface `org.apache.soap.util.xml.Deserializer`
- When web service interfaces are generated, specify that the custom deserializer should be used instead of the default
- Add the compiled class to a JAR file, and ensure that this JAR is on the server's runtime classpath (`WEBSHHERE_HOME/lib`)

In our example, we wanted to strictly limit String inputs so that they could only contain letters, numbers, periods, commas, and dashes. The deserializer class to implement the behavior was simple:

```
package com.atstake.bestpractices;

import org.apache.soap.rpc.SOAPContext;
import org.apache.soap.util.Bean;
import org.apache.soap.util.xml.*;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.apache.oro.text.perl.*;

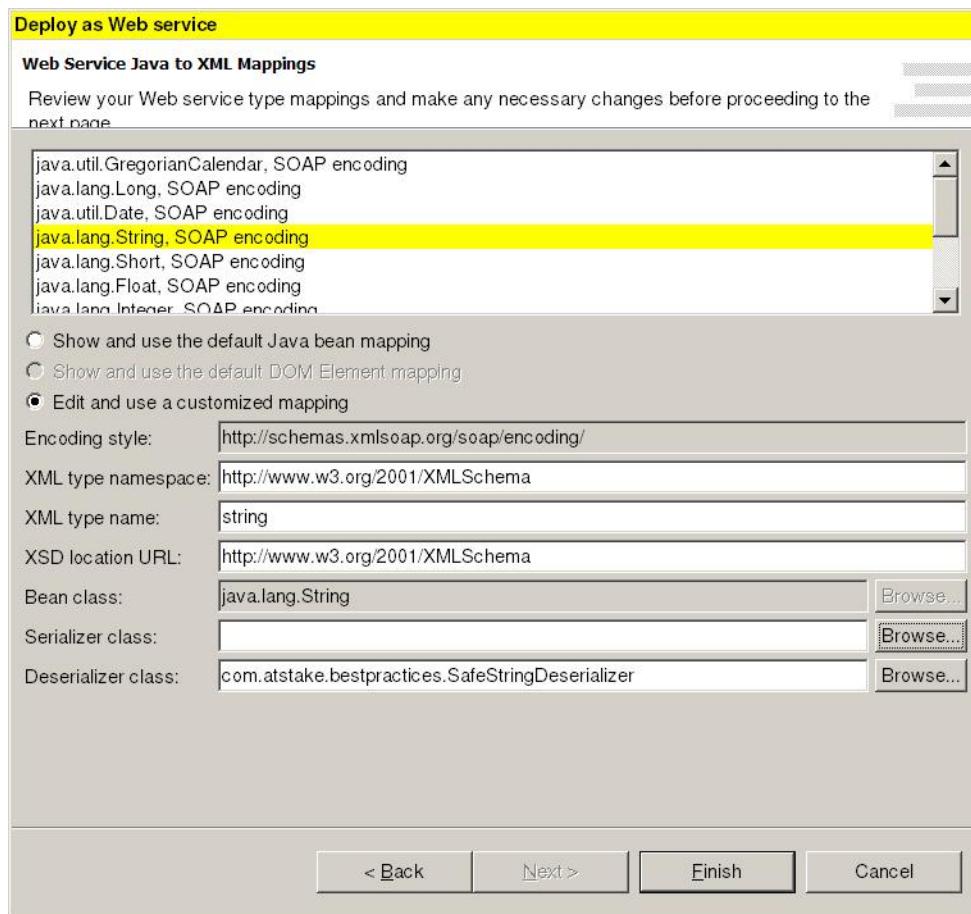
public class SafeStringDeserializer
    implements Deserializer {

    public SafeStringDeserializer()
    {
    }

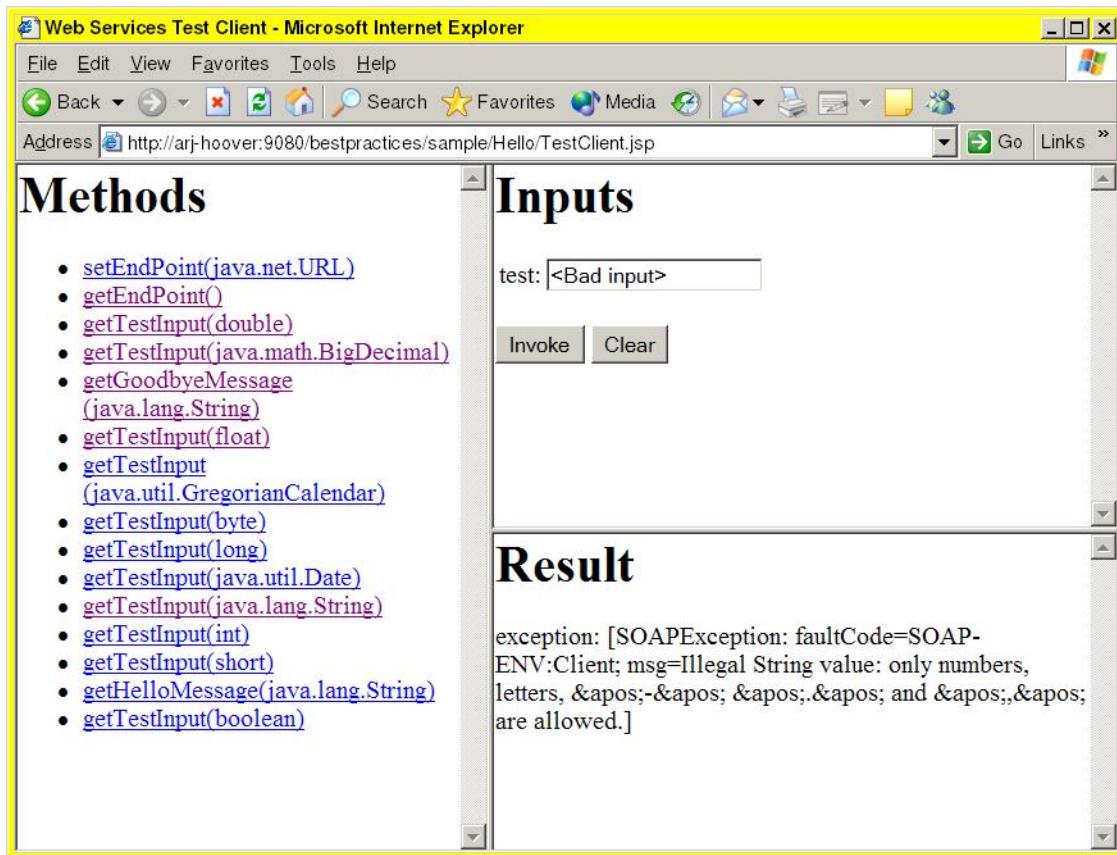
    static Class _mthclass$(String s)
    {
        try
        {
            return Class.forName(s);
        }
        catch(ClassNotFoundException classnotfoundexception)
        {
            throw new NoClassDefFoundError(classnotfoundexception.getMessage());
        }
    }
}
```

```
public Bean unmarshall(String s, QName qname, Node node, XMLJavaMappingRegistry  
xmljavamappingregistry,SOAPContext soapcontext)  
throws IllegalArgumentException  
{  
    Element element = (Element)node;  
    String safeAscii = "^[A-Z|a-z|1-9|\\ \\ |\\. |\\ , |\\ -]*$";  
    String s1 = DOMUtils.getChildCharacterData(element);  
    if (!matchRegexp(s1, safeAscii)) {  
        throw new IllegalArgumentException("Illegal String value: only numbers,  
letters, '-' '.' and ','  
        are allowed.");  
    }  
    return new Bean(java.lang.String.class, s1);  
}  
  
public static boolean matchRegexp(String value, String regexp) {  
    boolean match = false;  
    if (regexp != null && regexp.length() > 0) {  
        org.apache.oro.text.perl.Perl5Util r = new Perl5Util();  
        match = r.match="/" + regexp + "/", value);  
    }  
    return match;  
}  
}
```

When the web service was generated, we specified that SOAP type String should use the custom deserializer:



To test the validation routine, we used the JSP-based test client generated by WebSphere Studio:



The invalid string input for the web service `getTestInput(java.lang.String)` produced the SOAP fault, as expected.

See also:

<http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246292.html?Open>, page 448-464

<http://www-106.ibm.com/developerworks/webservices/library/ws-soapmap2>

<http://ws.apache.org/soap/docs/apiDocs/org/apache/soap/util/xml/class-use/Deserializer.html>

## 4.3 Host and Operating System

---

### Host and Operating System

The Host and Operating System are the foundation upon which an application executes. They include all system files, settings, network services, and other aspects of the server. IP Stack Hardening: A comparative analysis of the relative level of effort required to modify the network protocol stack. The analysis covers the changes or modifications required to the base IP stack in order to eliminate or to minimize available attack surfaces.

Specific Host and Operating System areas of analysis include:

- **IP Stack Hardening** . Changes or modifications required to the base IP stack in order to eliminate or to minimize available attack surfaces.
- **Service Minimization** . Reducing the number of network services listening on a network interface to the absolute minimum required for proper operation of the system.

The next sections provide further information on each area of analysis, and describe @stake's findings in detail.

## 4.3.1 IP Stack Hardening

---

### IP Stack Hardening

Changes or modifications required to the base IP stack in order to eliminate or to minimize available attack surfaces.

The IP Stack Hardening area of analysis includes this topic:

- Protocol Settings

The table below describes the topic at a high level, and lists prevailing best practices. The right-hand columns show the weights @stake assigned for each of the scenarios.

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Protocol Settings</b> . Production systems can be configured to reduce exposure to denials of service and other security issues. Protocol settings include refusing ICMP echo broadcasts and disabling IP forwarding systems. Hardening is most important in external scenarios, which are weighted high. The intranet scenario is rated medium.	ICMP Information disclosure attacks and other security issues are addressed and mitigated. Host IP stacks are configured to resist denial of service (DOS) attacks.	3	3	2

Scenario weights range from 1 (relatively unimportant) to 3 (extremely important). A scenario weight of zero means that the topic does not apply.

### Findings

The table below summarizes @stake's quantitative findings. For each analysis topic, the two high-level metrics (Best Practice Compliance and Ease of Securing) are shown. These are the arithmetic means of the underlying best practice and test case scores. The Ease of Securing metric is further broken down into four sub-metrics.

Topic	Platform	Best Practice Compliance	Overall	Ease of Securing			
				Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Protocol Settings <i>2 best practices, 4 test cases</i>	.NET Windows	<b>4</b>	<b>3.88</b>	4.5	4.5	2.25	4.25
	WebSphere Linux	<b>4</b>	<b>2.56</b>	3.5	2	1	3.75
	WebSphere Unix	<b>4</b>	<b>3.06</b>	3.75	4	1	3.5

This area of analysis was comprised of four (4) test cases designed to evaluate the ease and level of effort required to implement IP stack modifications in order to eliminate or to minimize available attack surfaces. Microsoft score generally higher than IBM in these test case due in large part to superior documentation and configuration examples.

The individual sections for each analysis topic describe @stake's findings in further detail.

## 4.3.1.1 Protocol Settings

---

### Protocol Settings

Production systems can be configured to reduce exposure to denials of service and other security issues. Protocol settings include refusing ICMP echo broadcasts and disabling IP forwarding systems. Hardening is most important in external scenarios, which are weighted high. The intranet scenario is rated medium. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 3
- Web Service: 3
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Protocol Settings topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=64]. ICMP Information disclosure attacks and other security issues are addressed and mitigated.	Windows 2003 includes the Internet Connection Firewall (ICF) and the Routing and Remote Access Basic Firewall. The ICF provides a way to restrict inbound TCP/IP traffic for a single host or server running Internet Connection Sharing (ICS). The Routing and Remote Access service Basic Firewall is a stateful firewall, which combines dynamic packet filtering of network traffic with a set of static packet filters.	<b>Wizard (4)</b> . The Internet Connection Firewall only requires a single checkbox to be enabled. Once enabled, all ICMP traffic is disabled by default.	Administrators should consider the Internet Connections Firewall (ICF) if they only need to restrict ICMP traffic. The administrator can then enable specific ICMP types individually from the ICF interface if required.
[id=65]. Host IP stacks are configured to resist denial of service (DOS) attacks.	The TCP/IP stack in Windows 2003 has various options that can be modified through the registry or Group Policy to configure it to be more resilient to network based attacks.	<b>Wizard (4)</b> . These changes necessary to implement this were clearly documented with appropriate explanations of the ramifications for each setting.	Test and implement the recommended changes in Microsoft Knowledge Base Article 324270 - HOW TO: Harden the TCP/IP Stack Against Denial of Service Attacks in Windows Server 2003.

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
<b>Best Practice Compliance score</b> , based on mean of 2 best practices: 4			

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=64.1]. Modify host to reject ICMP ECHO ICMP TIMESTAMP and ICMP NETMASK Requests.	Implementation Complexity	<b>Wizard (5)</b> . ICMP is blocked by default when enabling the Internet Connection Firewall.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation available locally in the Help and Support Center is clear and easy to follow for disabling ICMP with the Internet Connection Firewall (ICF).
	Implementor Competence	<b>Expert/intermediate (2)</b> . The administrator has extensive experience with the Windows OS family. However, the Internet Connection Firewall (ICF) and Basic Firewall are new to Windows XP and Windows 2003; therefore expectations relative to pre-existing knowledge of the products are low.
	Time to Implement	<b>Low (5)</b> . It took less than 2 minutes to search the Help and Support Center and get instructions on how to disable all forms of ICMP.
[id=64.2]. Modify the host to disable IP Source Routing..	Implementation Complexity	<b>Wizard+ (4)</b> . The configuration requires the addition of 1 registry value. This could be performed manually on one server or through Group Policy to apply to multiple servers.
	Documentation and Examples	<b>Suitable (4)</b> . Microsoft has documented OS hardening steps in chapter 6 of the <i>Securing Microsoft Windows 2000 Server Solution Guide</i> and includes clear and concise details on the additional registry value to disable IP Source Routing.
	Implementor Competence	<b>Intermediate (3)</b> . The administrator has extensive experience with the Windows OS family. However, this is not the kind of functionality an administrator would be exposed to daily; therefore expectations relative to pre-existing knowledge are low.
	Time to Implement	<b>Low to Medium (4)</b> . It took less than 10 minutes to search the Help and Support Center, get instructions on the recommended configurations, and apply them to the system.
[id=64.3]. Modify the host to disable ICMP redirects.	Implementation Complexity	<b>Wizard (5)</b> . ICMP redirects are blocked by default when enabling the Internet Connection Firewall.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation available locally in the Help and Support Center is clear and easy to follow for disabling ICMP with the Internet Connection Firewall (ICF).
	Implementor Competence	<b>Expert/intermediate (2)</b> . The administrator has extensive experience with the Windows OS family. However, the Internet Connection Firewall (ICF) and Basic Firewall are new to Windows XP and Windows 2003; therefore expectations relative to pre-existing knowledge of the products are low.
	Time to Implement	<b>Low (5)</b> . It took less than 2 minutes to search the Help and Support Center and get instructions on how to disable ICMP redirects.

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=65.1]. Modify the host to defend against Denial of Service attacks.	Implementation Complexity	<b>Wizard+ (4)</b> . The configuration requires the addition of 5 registry values. This could be performed manually on one server or through Group Policy for multiple servers.
	Documentation and Examples	<b>Suitable (4)</b> . Microsoft provides clear and concise details on protecting against DoS attacks in the Knowledge Base Article 324270 - HOW TO: Harden the TCP/IP Stack Against Denial of Service Attacks in Windows Server 2003. Additional information about the registry values can be found in chapter 6 of the Securing Microsoft Windows 2000 Server Solution Guide.
	Implementor Competence	<b>Expert/intermediate (2)</b> . The administrator has extensive experience with the Windows OS family. However, this is not the kind of functionality an administrator would be exposed to daily; therefore expectations relative to pre-existing knowledge are low.
	Time to Implement	<b>Medium (3)</b> . It took less than 35 minutes to search the Help and Support Center, get instructions on the recommended configurations, and apply them to the system.

**Ease of Securing score**, based on mean of 4 test cases: 3.88

### Notes

#### Test Case 64.1

We performed a search in the Help and Support Center for **disable ICMP** and got the specific article *Enable or disable Internet Control Message Protocol requests for ICF: Network Connections* as the fourth topic in Help Topics. We were pleased to see that the article cautioned, *All ICMP options are disabled by default. If you enable ICMP options on a computer running ICF, that computer may become visible to the Internet and vulnerable to attacks.*

#### Test Case 64.2

Microsoft has released a set of guides that include: the *Securing Microsoft Windows 2000 Server Solution Guide*, the *Securing Microsoft Windows 2000 Server Delivery Guide*, the *Securing Microsoft Windows 2000 Server Test Guide*, and the *Securing Microsoft Windows 2000 Support Plan*. These guides provide an extensive review of Microsoft Windows 2000 Server and the tools to assess security risks specific to Windows 2000 Servers. @stake suggests the guides be updated for Windows 2003 and referenced from the Security section of the Help and Support Center.

#### Test Case 64.3

I have to ask why the choice to enable Internet Connection Firewall is not part of the initial installation. Even if the default setting is to be disabled, I think it would be a positive addition. I would also like to see additional service definitions that correspond to the necessary setting for every type of server role and an option in the Server Role Wizard to enable the appropriate service definitions.

### Test Case 65.1

The level of effort for this entry was set to a 3 only because of the amount of time involved in reviewing each of the settings and applying them to the system. The individual application of any one of the settings would have been rated a 4.

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
<i>Linux.</i> [id=64]. ICMP Information disclosure attacks and other security issues are addressed and mitigated.	The Linux kernel can be modified to disable various ICMP information disclosure issues. Additionally, either of the two included firewall packages, ipchains and iptables, can be configured to disallow this traffic.	<b>Wizard (4)</b> . These configuration controls can be implemented in a fairly straightforward manner via kernel configuration and firewall configuration	Configure the Linux kernel to ignore the various means of information gathering via ICMP. Additionally, configure the chosen firewall package to disallow all or specific ICMP types.
<i>Unix.</i> [id=64]. ICMP Information disclosure attacks and other security issues are addressed and mitigated.	Network driver settings can be configured to limit information disclosure and security issues.	<b>Wizard (4)</b> . The network driver settings can be configured securely in a straightforward fashion. The vendor provides appropriate tools to implement the changes.	Use the vendor supplied nddconfig script to implement network driver security settings.
<i>Linux.</i> [id=65]. Host IP stacks are configured to resist denial of service (DOS) attacks.	Linux network stacks can be modified to protect against certain network-based Denial of Service (DoS) attacks through the sysctl kernel configuration utility. These kernel changes can be made to running systems or can be automated to occur with every reboot.	<b>Wizard (4)</b> . DoS mitigation can be implemented fairly easily in the Linux TCP/IP stack.	Use a sysctl configuration file or modified the rc.local file to modify the Linux kernel network settings to protect against various Denial of Service attacks.
<i>Unix.</i> [id=65]. Host IP stacks are configured to resist denial of service (DOS) attacks.	Unix network stacks can be modified to protect against certain network-based Denial of Service attacks through the ndd network driver configuration utility. These changes can be made to running systems or can be automated to occur with every reboot.	<b>Wizard (4)</b> . Best practice settings for TCP/IP stack resistance to DoS attacks can be easily implemented via vendor supplied tools.	Use the Unix vendor-provided nddconfig initialization script to implement DoS and other network stack hardening techniques.

**Best Practice Compliance scores**, based on mean of 2 best practices: Linux 4, Unix 4

### Level of Effort Analysis

Test Case	Criteria	Rating
<i>Linux. [id=64.1]. Modify host to reject ICMP ECHO ICMP TIMESTAMP and ICMP NETMASK Requests.</i>	Implementation Complexity	<b>Small amount of code (3)</b> . The configuration requires the modification of one kernel parameter, and the configuration of two firewall rules. If desired, all of these packets can be blocked by either of the included firewall packages.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Documentation for these kernel settings is not readily apparent and accessible on the vendor site. Further searching will discover the configuration required for this setting, but proper context and background is not supplied.
	Implementor Competence	<b>Expert (1)</b> . The implementer is very familiar with Linux, Unix operating systems, security, and host hardening practices.
	Time to Implement	<b>Low to Medium (4)</b> . Once documentation for the setting is found and reviewed, the modification is very simple.
<i>Linux. [id=64.2]. Modify the host to disable IP Source Routing..</i>	Implementation Complexity	<b>Wizard+ (4)</b> . The configuration requires the modification of one kernel parameter.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Documentation for this kernel setting is not readily apparent and accessible on the vendor site. Further searching will discover the configuration required for this setting, but proper context and background is not supplied.
	Implementor Competence	<b>Expert (1)</b> . The implementer is very familiar with Linux, Unix operating systems, security, and host hardening practices.
	Time to Implement	<b>Low to Medium (4)</b> . Once documentation for the setting is found and reviewed, the modification is very simple.
<i>Linux. [id=64.3]. Modify the host to disable ICMP redirects.</i>	Implementation Complexity	<b>Wizard+ (4)</b> . The configuration requires the modification of one kernel parameter.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Documentation for this kernel setting is not readily apparent and accessible on the vendor site. Further searching will discover the configuration required for this setting, but proper context and background is not supplied.
	Implementor Competence	<b>Expert (1)</b> . The implementer is very familiar with Linux, Unix operating systems, security, and host hardening practices.
	Time to Implement	<b>Low to Medium (4)</b> . Once documentation for the setting is found and reviewed, the modification is very simple.

### Level of Effort Analysis

Test Case	Criteria	Rating
<i>Unix. [id=64.1]. Modify host to reject ICMP ECHO ICMP TIMESTAMP and ICMP NETMASK Requests.</i>	Implementation Complexity	<b>Small amount of code (3)</b> . This configuration requires the modification of three network driver parameters. Broadcast and multicast ICMP echo requests can be blocked through driver configuration, but unicast ICMP echo requests cannot be disallowed on a network driver level; the use of a host-based firewall is required. A vendor-supplied nddconfig init script can implement the network driver modifications.
	Documentation and Examples	<b>Suitable (4)</b> . Excellent documentation exists describing the configuration of the application network drivers. Although the relevant Blueprint document is tailored for the previous version of the vendor's Unix distribution, the documentation is applicable to the current version as well.
	Implementor Competence	<b>Expert (1)</b> . The implementer is very familiar with the vendor's Unix distribution, Unix operating systems in general, security, and host hardening practices.
	Time to Implement	<b>Medium to High (2)</b> . The kernel configuration parameters require only a few minutes. Installing and configuring a host-based firewall would take considerably longer.
<i>Unix. [id=64.2]. Modify the host to disable IP Source Routing..</i>	Implementation Complexity	<b>Wizard+ (4)</b> . The configuration requires the modification of one kernel parameter. The required modification is included in the vendor supplied nddconfig script.
	Documentation and Examples	<b>Suitable (4)</b> . Clear documentation exists for the implementation of these stack modifications. The most comprehensive documentation, the Unix vendor's Blueprints, have not been updated for the current version of the OS, though the documentation for the previous version is appropriate.
	Implementor Competence	<b>Expert (1)</b> . The implementer is very familiar with the vendor's Unix distribution, Unix operating systems in general, security, and host hardening practices.
	Time to Implement	<b>Low to Medium (4)</b> . Once documentation for the setting is found and reviewed, the modification is very simple.
<i>Unix. [id=64.3]. Modify the host to disable ICMP redirects.</i>	Implementation Complexity	<b>Wizard+ (4)</b> . The configuration requires the modification of one kernel parameter. The required modification is included in the vendor supplied nddconfig script.
	Documentation and Examples	<b>Suitable (4)</b> . Clear documentation exists for the implementation of these stack modifications. The most comprehensive documentation, the Unix vendor's Blueprints, have not been updated for the current version of the OS, though the documentation for the previous version is appropriate.
	Implementor Competence	<b>Expert (1)</b> . The implementer is very familiar with the vendor's distribution of Unix, Unix operating systems in general, security, and host hardening practices.
	Time to Implement	<b>Low to Medium (4)</b> . Once documentation for the setting is found and reviewed, the modification is very simple.

### Level of Effort Analysis

Test Case	Criteria	Rating
<i>Linux.</i> [id=65.1]. Modify the host to defend against Denial of Service attacks.	Implementation Complexity	<b>Small amount of code (3)</b> . This configuration requires the modification of several kernel parameters.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Full documentation of these kernel settings is not easily found within the vendor documentation. The required settings are more fully documented in the kernel source code.
	Implementor Competence	<b>Expert (1)</b> . The implementer is very familiar with Linux, Unix operating systems, security, and host hardening practices.
	Time to Implement	<b>Medium (3)</b> . Some time is required to review the relevant documentation and edit the appropriate files to make the configuration persistent across reboots. These changes can be made to running systems; a reboot is not required.
<i>Unix.</i> [id=65.1]. Modify the host to defend against Denial of Service attacks.	Implementation Complexity	<b>Wizard+ (4)</b> . The Unix vendor-provided nddconfig script provides a straightforward means of implementing the required protections against network-based DoS attacks.
	Documentation and Examples	<b>Suitable (4)</b> . Clear documentation exists for the implementation of these stack modifications. The most comprehensive documentation, the Unix vendor's Blueprints, have not been updated for the current version of the OS, though the documentation for the previous version is appropriate.
	Implementor Competence	<b>Expert (1)</b> . The implementer is very familiar with the vendor's distribution of Unix, Unix operating systems in general, security, and host hardening practices.
	Time to Implement	<b>Low to Medium (4)</b> . The documentation supporting these changes is easily found on the vendor site. These changes can be effected on running systems; they do not require a reboot.

**Ease of Securing scores**, based on mean of 4 test cases: Linux 2.56, Unix 3.06

### Notes

#### Linux Test Case 64.1

The kernel parameter modified is /proc/sys/net/ipv4/icmp\_echo\_ignore\_all.

#### Linux Test Case 64.2

The kernel parameter modified is /proc/sys/net/ipv4/conf/\*/accept\_source\_route. (\* represents an interface name).

#### Linux Test Case 64.3

The kernel parameter modified is /proc/sys/net/ipv4/conf/\*/accept\_redirects. (\* represents an interface name).

#### Unix Test Case 64.1

See the the Unix vendor's Blueprint document *Unix Operating Environment Network Settings for Security* for further information on Unix network driver hardening. The ndd parameters configured are: /dev/ip ip\_respond\_to\_echo\_broadcast, /dev/ip ip\_respond\_to\_timestamp\_broadcast, ip\_respond\_to\_address\_mask\_broadcast.

#### Unix Test Case 64.2

See the the Unix vendor's Blueprint document *Unix Operating Environment Network Settings for Security* for further information on Unix network driver hardening. The ndd parameter configured is: /dev/tcp tcp\_rev\_src\_routes

#### Unix Test Case 64.3

See the the Unix vendor's Blueprint document *Unix Operating Environment Network Settings for Security* for further information on Unix network driver hardening. The ndd parameter configured is: /dev/ip ip\_ignore\_redirect

#### Linux Test Case 65.1

The kernel parameters modified are /proc/sys/net/ipv4/icmp\_ignore\_broadcasts and /proc/sys/net/ipv4/tcp\_max\_syn\_backlog.

#### Unix Test Case 65.1

See the the Unix vendor's Blueprint document *Unix Operating Environment Network Settings for Security* for further information on Unix network driver hardening. The ndd parameters configured are: /dev/tcp tcp\_conn\_req\_max\_q0, /dev/tcp tcp\_conn\_req\_max\_q, /dev/ip ip\_respond\_to\_echo\_broadcast

## 4.3.2 Service Minimization

---

### Service Minimization

Reducing the number of network services listening on a network interface to the absolute minimum required for proper operation of the system.

The Service Minimization area of analysis includes these topics:

- Installed Packages
- Network Services

The table below describes the topics at a high level, and lists prevailing best practices. The right-hand columns show the weights @stake assigned for each of the scenarios.

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Installed Packages</b> . Installed Packages: Installed software packages are all software applications, libraries, and support packages present on a system, regardless of whether or not they are actively used. All application scenarios are weighted medium.	Modify the host to remove any extraneous software packages and functionality.	2	2	2
<b>Network Services</b> . Network services handle requests made on a network port. Production systems should only run required network services. Production network services should be limited to running on production network interfaces. Service minimization is most important in external application scenarios which are weighted high. The intranet scenario is rated medium.	Production systems should only run required network services.	3	3	2

Scenario weights range from 1 (relatively unimportant) to 3 (extremely important). A scenario weight of zero means that the topic does not apply.

### Findings

The table below summarizes @stake's quantitative findings. For each analysis topic, the two high-level metrics (Best Practice Compliance and Ease of Securing) are shown. These are the arithmetic means of the underlying best practice and test case scores. The Ease of Securing metric is further broken down into four sub-metrics.

Topic	Platform	Best Practice Compliance	Ease of Securing				
			Overall	Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Installed Packages <i>1 best practice, 3 test cases</i>	.NET Windows	<b>5</b>	<b>4</b>	5	5	1	5
	WebSphere Linux	<b>4</b>	<b>2.75</b>	3	3	2.33	2.67
	WebSphere Unix	<b>4</b>	<b>2.83</b>	3	3.33	2.33	2.67
Network Services <i>1 best practice, 2 test cases</i>	.NET Windows	<b>4</b>	<b>3.63</b>	4	4.5	1.5	4.5
	WebSphere Linux	<b>4</b>	<b>3.5</b>	4	4	1	5
	WebSphere Unix	<b>3</b>	<b>2.63</b>	3	3.5	1	3

This area of analysis was comprised of five (5) test cases designed to evaluate the ease and level of effort required to reducing the number of network services listening on a network interface to the absolute minimum required for secure operation of the environment. With the release of Windows Server 2003, Microsoft has dramatically changed its stance on enabling many services by default. This has resulted in a much smaller default attack surface for the platform. Red Hat and other Unix vendors have been slower to take this approach. The scores for this area of analysis reflect Microsoft's efforts to reduce the amount of services enabled by default.

The individual sections for each analysis topic describe @stake's findings in further detail.

### 4.3.2.1 Installed Packages

---

## Installed Packages

**Installed Packages:** Installed software packages are all software applications, libraries, and support packages present on a system, regardless of whether or not they are actively used. All application scenarios are weighted medium. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Installed Packages topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

## Microsoft .NET

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=67]. Modify the host to remove any extraneous software packages and functionality.	Windows 2003 Server includes numerous core components that are not installed in the base OS. These Windows Components can be added or removed through the Manage Your Server Wizard or individually through the Windows Components Wizard.	<b>Transparent (5)</b> . The base installation of Windows 2003 Server includes only a limited number of Windows Components. It is up to the administrator to run the Manage Your Server Wizard after installation and select the roles that will be necessary for the server to meet business requirements.	The default Windows 2003 Server has the minimum system software installed. Administrators should use the Manage Your Server Wizard to enable any server roles that are necessary and consider removing any Windows Components that are not necessary to meet business requirements.

**Best Practice Compliance score**, based on mean of 1 best practice: 5

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=67.1]. Configure base operating system to remove all extraneous software packages.	Implementation Complexity	<b>Wizard (5)</b> . By default, the base OS is installed with a limited number of Windows Components.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation available locally in the Help and Support Center is clear and easy to follow for security features of Windows 2003.
	Implementor Competence	<b>Expert (1)</b> . The administrator has extensive experience with the Windows OS family.
	Time to Implement	<b>Low (5)</b> . No time is required to implement this; it is enabled by default.
[id=67.2]. Configure system to remove any unnecessary functionality from the application server.	Implementation Complexity	<b>Wizard (5)</b> . By default, the application server is installed in a secure configuration with minimum functionality.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation available locally in the Help and Support Center is clear and easy to follow for security features of IIS 6.0 and ASP.NET.
	Implementor Competence	<b>Expert (1)</b> . The administrator has extensive experience with the Windows OS family.
	Time to Implement	<b>Low (5)</b> . No time is required to implement this; it is enabled by default.
[id=67.3]. Configure system to remove any unnecessary functionality from the web server.	Implementation Complexity	<b>Wizard (5)</b> . By default, the web server is installed in a secure configuration with minimum functionality.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation available locally in the Help and Support Center is clear and easy to follow for security features of IIS 6.0.
	Implementor Competence	<b>Expert (1)</b> . The administrator has extensive experience with the Windows OS family.
	Time to Implement	<b>Low (5)</b> . No time is required to implement this; it is enabled by default.

Ease of Securing score, based on mean of 3 test cases: 4

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
<i>Linux</i> . [id=67]. Modify the host to remove any extraneous software packages and functionality.	RedHat uses RPM (RedHat Package Manager) packages to distribute, add, and upgrade vendor-supplied software. Several operating system utilities are supplied to manage, install, and query system packages. Kickstart, a means of implementing automated installations, can simplify the process of software minimization during system installation.	<b>Wizard (4)</b> . The modular design and implementation of RPM packages, coupled with the supplied tools, makes the process of minimizing system software straightforward to implement.	Minimize system software installed, starting with system installation.

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
<i>Unix.</i> [id=67]. Modify the host to remove any extraneous software packages and functionality.	Unix uses a package format to distribute, add, and upgrade vendor-supplied software. Several operating system utilities are supplied to manage, install, and query system packages. Jumpstart, a means of implementing automated installations, can simplify the process of software minimization during system installation.	<b>Wizard (4)</b> . The modular design and implementation of Unix packages, coupled with the supplied tools, makes the process of minimizing system software straightforward to implement.	Minimize system software installed, starting with system installation. Use the vendor-supplied minimization methodology to guide testing and implementation of minimized systems.

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 4, Unix 4

## Level of Effort Analysis

Test Case	Criteria	Rating
<i>Linux.</i> [id=67.1]. Configure base operating system to remove all extraneous software packages.	Implementation Complexity	<b>Small amount of code (3)</b> . Effectively minimizing RedHat systems requires a systematic approach and methodology. The tools provided are adequate to minimize installed software.
	Documentation and Examples	<b>Adequate (3)</b> . Adequate documentation exists to support the effective use of RPM to manage system software. However, the documentation does not explicitly prescribe a methodology or approach to minimizing a system for security purposes.
	Implementor Competence	<b>Expert (1)</b> . The implementer is very familiar with Linux, Unix operating systems, security, and host hardening practices.
	Time to Implement	<b>Medium to High (2)</b> . The actual act of removing extraneous packages takes only minutes. However, research and testing is required to determine the appropriate amount of minimization that can be applied to each system (i.e. determining the minimum required services and support files and software).
<i>Linux.</i> [id=67.2]. Configure system to remove any unnecessary functionality from the application server.	Implementation Complexity	<b>Small amount of code (3)</b> . Unneeded RPM packages during the initial installation can be removed with the RPM tool. Additionally, the WebSphere Administrative Console provides a graphical means of uninstalling sample and default applications created during installation.
	Documentation and Examples	<b>Adequate (3)</b> . Removing packages and software is documented within the WebSphere Administrative Console and online. However, the documentation does not appear to directly address minimizing the application server for security purposes.
	Implementor Competence	<b>Intermediate (3)</b> . The administrator is very familiar with Unix operating systems and web administration but is unfamiliar with WebSphere administration.
	Time to Implement	<b>Medium (3)</b> . Removing the software packages and installed applications takes only a few minutes. It takes a few minutes to review the documentation and installed packages to determine which components can be removed.

### Level of Effort Analysis

Test Case	Criteria	Rating
<i>Linux. [id=67.3]. Configure system to remove any unnecessary functionality from the web server.</i>	Implementation Complexity	<b>Small amount of code (3)</b> . Removing extraneous files and material from the IBM HTTP server involves deleting unneeded files and directories.
	Documentation and Examples	<b>Adequate (3)</b> . Adequate documentation exists online and as product manuals to support the minimization of the IBM HTTP Server. However, the documentation does not appear to directly address minimizing the server for security purposes.
	Implementor Competence	<b>Intermediate (3)</b> . The administrator is very familiar with Unix operating systems and web administration but is not directly familiar with IBM HTTP server administration.
	Time to Implement	<b>Medium (3)</b> . Removing the unneeded files and directories takes only a few minutes. It takes some time to review the documentation and installed files to determine which components can be removed.
<i>Unix. [id=67.1]. Configure base operating system to remove all extraneous software packages.</i>	Implementation Complexity	<b>Small amount of code (3)</b> . Effectively minimizing Unix systems requires a systematic approach and methodology. The tools provided are adequate to minimize installed software.
	Documentation and Examples	<b>Suitable (4)</b> . A Unix Blueprint document directly addresses the issue of removing extraneous software from Unix systems to improve security. Additionally, the Unix vendor provides a great deal of documentation that addresses installation and maintenance automation that helps facilitate effective system minimization.
	Implementor Competence	<b>Expert (1)</b> . The implementer is very familiar with the vendor's distribution of Unix, Unix operating systems in general, security, and host hardening practices.
	Time to Implement	<b>Medium to High (2)</b> . The actual act of removing extraneous packages takes only minutes. However, research and testing is required to determine the appropriate amount of minimization that can be applied to each system (i.e. determining the minimum required services and support files and software).
<i>Unix. [id=67.2]. Configure system to remove any unnecessary functionality from the application server.</i>	Implementation Complexity	<b>Small amount of code (3)</b> . Sample packages installed during the initial installation can be removed with the Unix pkgadd tools. Additionally, the WebSphere Administrative Console provides a graphical means of uninstalling sample and default applications created during installation.
	Documentation and Examples	<b>Adequate (3)</b> . Removing packages and software is documented within the WebSphere Administrative Console and online. However, the document does not appear to directly address minimizing the application server for security purposes.
	Implementor Competence	<b>Intermediate (3)</b> . The administrator is familiar with Unix operating systems and web administration but is unfamiliar with WebSphere administration.
	Time to Implement	<b>Medium (3)</b> . Removing the software packages and installed applications takes only a few minutes. It takes a few minutes to review the documentation and installed packages to determine which components can be removed.

### Level of Effort Analysis

Test Case	Criteria	Rating
<i>Unix.</i> [id=67.3]. Configure system to remove any unnecessary functionality from the web server.	Implementation Complexity	<b>Small amount of code (3)</b> . Removing extraneous files and material from the IBM HTTP server involves deleting unneeded files and directories.
	Documentation and Examples	<b>Adequate (3)</b> . Adequate documentation exists online and as product manuals to support the minimization of the IBM HTTP Server. However, the documentation does not appear to directly address minimizing the server for security purposes.
	Implementor Competence	<b>Intermediate (3)</b> . The administrator is familiar with Unix operating systems and web administration but is not directly familiar with IBM HTTP server administration.
	Time to Implement	<b>Medium (3)</b> . Removing the unneeded files and directories takes only a few minutes. It takes some time to review the documentation and installed files to determine which components can be removed.

**Ease of Securing scores**, based on mean of 3 test cases: Linux 2.75, Unix 2.83

### Notes

#### Linux Test Case 67.2

In the WebSphere Administrative Console, Applications/Enterprise Applications provides a means to remove unneeded installed WebSphere applications. For the purposes of the testing, specific functionality requirements for the applications server were not specified, so components of the application server that may often be disabled or removed were not addressed. Instead, default files, applications, and documentation were the focus of minimization.

#### Linux Test Case 67.3

For the purposes of the testing, specific functionality requirements for the web servers were not specified, so components of the web server software that may often be disabled or removed were not addressed. Instead, default files and documentation were the focus of minimization.

#### Unix Test Case 67.1

See the the Unix vendor's Blueprint document *Minimizing the Unix Operating Environment for Security* for further information on Unix minimization methodology and procedures.

#### Unix Test Case 67.2

In the WebSphere Administrative Console, Applications/Enterprise Applications provides a means to remove unneeded installed WebSphere applications. The Unix packages WSBM1AA and WSBM1AA are both listed as WebSphere samples. For the purposes of the testing, specific functionality requirements for the applications server were not specified, so components of the application server that may often be disabled or

removed were not addressed. Instead, default files, applications, and documentation were the focus of minimization.

#### Unix Test Case 67.3

For the purposes of the testing, specific functionality requirements for the web servers were not specified, so components of the web server software that may often be disabled or removed were not addressed. Instead, default files and documentation were the focus of minimization.

### 4.3.2.2 Network Services

---

#### Network Services

Network services handle requests made on a network port. Production systems should only run required network services. Production network services should be limited to running on production network interfaces. Service minimization is most important in external application scenarios which are weighted high. The intranet scenario is rated medium. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 3
- Web Service: 3
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Network Services topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

#### Microsoft .NET

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=66]. Production systems should only run required network services.	Windows 2003 Server only installs a limited set of network services that includes NetBIOS, Direct Hosting, IPSec, and NTP. Additional network services can be installed via the Manage Your Server snap-in that will allow the user to add server roles such as; File Server, Print Server, Application Server, Terminal Server, DNS Server, DHCP Server, and/or Domain Controller (Active Directory). Windows 2003 Server includes the Security Templates snap-in for managing configuration templates that can be used to disable unnecessary services. The templates can be applied to an individual machine with the Security Configuration and Analysis snap-in or to all computers in the Active Directory through Group Policy. Windows 2003 Server also includes sc.exe (Service Controller) in the base build to manage Windows Services at the command line.	<b>Wizard (4)</b> . Windows 2003 Server only installs a limited set of network services and provides the Manage Your Server snap-in to install only the components necessary to meet business requirements.	Administrators should install Windows 2003 Server and then use the Manage Your Server snap-in to configure the appropriate role(s). If additional configuration is necessary, then the administrator should consider implementing a custom Security Template that can be applied with the Security Configuration and Analysis snap-in or through Group Policy.

**Best Practice Compliance score**, based on mean of 1 best practice: 4

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=66.1]. Modify host to disable services not required to support traditional web application and web services.	Implementation Complexity	<b>Wizard+ (4)</b> . By default, Windows 2003 Server only enabled the necessary services based on selected server roles. However, to get the least number of services, a Security Template was created through the Security Template snap-in and applied with the Security Configuration and Analysis snap-in. The Security Template could have also been applied through Group Policy to configure multiple servers.
	Documentation and Examples	<b>Suitable (4)</b> . Microsoft has documented Hardening steps for specific server roles in chapter 7 of the <i>Securing Microsoft Windows 2000 Server Solution Guide</i> and includes clear and concise details on security configuration for an IIS web server and the potential impact of the recommended hardening steps. The documentation available locally in the Help and Support Center is clear and easy to follow for using the Security Template snap-in and the Security Configuration and Analysis snap-in.
	Implementor Competence	<b>Expert (1)</b> . The administrator has extensive experience with the Windows OS family.
	Time to Implement	<b>Low to Medium (4)</b> . It took less than 10 minutes to disable services that were not required to support the web application and web services.

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=66.2]. Configure the host to log information regarding attempts to access restricted services.	Implementation Complexity	<b>Wizard+ (4)</b> . Internet Connection Firewall provides the functionality to log all dropped or successful packets. This can be enabled through the ICF interface.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation available locally in the Help and Support Center is clear and easy to follow for configuration logging with the Internet Connection Firewall (ICF).
	Implementor Competence	<b>Expert/intermediate (2)</b> . The administrator has extensive experience with the Windows OS family. However, the Internet Connection Firewall (ICF) and Basic Firewall are new to Windows XP and Windows 2003; therefore expectations relative to pre-existing knowledge of the products are low.
	Time to Implement	<b>Low (5)</b> . It took less than 2 minutes to search the Help and Support Center and get instructions on how to enable logging in the ICF.

**Ease of Securing score**, based on mean of 2 test cases: 3.63

### Notes

#### Test Case 66.1

Microsoft has released a set of guides that include: the *Securing Microsoft Windows 2000 Server Solution Guide*, the *Securing Microsoft Windows 2000 Server Delivery Guide*, the *Securing Microsoft Windows 2000 Server Test Guide*, and the *Securing Microsoft Windows 2000 Support Plan*. These guides provide an extensive review of Microsoft Windows 2000 Server and the tools to assess security risks specific to Windows 2000 Servers. @stake suggests the guides be updated for Windows 2003 and referenced from the Security section of the Help and Support Center.

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
<i>Linux</i> . [id=66]. Production systems should only run required network services.	By default, RedHat Advanced Server installs a very limited set of network services. Additional unneeded services can be installed through a number of methods, including ntsysv, a simple interface for modifying network services that start at system boot, and manual system modification.	<b>Wizard (4)</b> . RedHat Advanced Server can be easily configured to disable extraneous network services.	Use ntsysv and manual configuration to disable unneeded network services.

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
<i>Unix.</i> [id=66]. Production systems should only run required network services.	By default, the Unix vendor's distribution installs and enables a number of unneeded network services. These services can be disabled through a number of means, including inetd configuration and manual configuration.	<b>Developer extends (3)</b> . A number of services need to be disabled on the base OS installation, and several configuration means must be employed to disable these services.	Disable all unessential network services through manual configuration.

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 4, Unix 3

### Level of Effort Analysis

Test Case	Criteria	Rating
<i>Linux.</i> [id=66.1]. Modify host to disable services not required to support traditional web application and web services.	Implementation Complexity	<b>Wizard+ (4)</b> . Only one service needed to be disabled. The service was disabled through the ntsysv application.
	Documentation and Examples	<b>Suitable (4)</b> . Adequate documentation exists to facilitate the disabling of unneeded network services.
	Implementor Competence	<b>Expert (1)</b> . The implementer is very familiar with Linux, Unix operating systems, security, and host hardening practices.
	Time to Implement	<b>Low (5)</b> . It took only a minute to disable the one unneeded service. The system does not need to be restarted to implement the change.
<i>Linux.</i> [id=66.2]. Configure the host to log information regarding attempts to access restricted services.	Implementation Complexity	<b>Wizard+ (4)</b> . Logging of invalid or inappropriate access attempts to restricted network services is easily facilitated through configuration of either of the included firewall packages.
	Documentation and Examples	<b>Suitable (4)</b> . System manual (man) pages, hard copy, and online documentation exist to support firewall configuration.
	Implementor Competence	<b>Expert (1)</b> . The implementer is very familiar with Linux, Unix operating systems, security, host hardening, and firewall practices.
	Time to Implement	<b>Low (5)</b> . It takes only a few minutes to edit the firewall configuration to support logging of invalid or inappropriate connections.
<i>Unix.</i> [id=66.1]. Modify host to disable services not required to support traditional web application and web services.	Implementation Complexity	<b>Small amount of code (3)</b> . Several different configurations need to be manually altered to disable unneeded network services, including disabling inetd and manually disabling other network services.
	Documentation and Examples	<b>Suitable (4)</b> . Adequate documentation exists to assist in disabling extraneous network services.
	Implementor Competence	<b>Expert (1)</b> . The implementer is very familiar with the vendor's distribution of Unix, Unix operating systems in general, security, and host hardening practices.
	Time to Implement	<b>Low to Medium (4)</b> . The modifications required involve file editing and renaming. The system does not have to be restarted to implement the required changes.

### Level of Effort Analysis

Test Case	Criteria	Rating
<i>Unix.</i> [id=66.2]. Configure the host to log information regarding attempts to access restricted services.	Implementation Complexity	<b>Small amount of code (3)</b> . Logging of connection attempts to restricted services cannot be implemented on a global basis. The inetd daemon can log connection attempts. Additionally, libwrap-enabled services can log connection attempts through TCPWrappers. A host-based firewall, such as the one included with the OS, would need to be implemented to facilitate more global logging.
	Documentation and Examples	<b>Adequate (3)</b> . Adequate documentation exists to configure inetd logging and (HOST-BASED FIREWALL) firewalls.
	Implementor Competence	<b>Expert (1)</b> . The implementer is very familiar with the vendor's Unix distribution, Unix operating systems, security, and host hardening practices.
	Time to Implement	<b>Medium to High (2)</b> . Modifying inetd and libwrap-enabled services takes only a few minutes. Installing and configuring a host-based firewall would take considerably longer.

**Ease of Securing scores**, based on mean of 2 test cases: Linux 3.5, Unix 2.63

### Notes

#### Linux Test Case 66.1

The portmap was the only network service that needed to be disabled.

#### Linux Test Case 66.2

The relevant configuration files for ipchains and iptables are /etc/sysconfig/ipchains and /etc/sysconfig/iptables, respectively. The keyword LOG is used to enable logging for iptables. The l argument is used to enable logging for ipchains.

#### Unix Test Case 66.1

The Unix vendor's blueprint document, *Unix Operating Environment Security*, provides details on disabling unneeded network services.

## 4.4 Web Server

---

### Web Server

A web server is an integral component of a high performance and secure architecture. In addition to serving static content to browsers, web servers reduce the load on application servers by managing client connections, terminating SSL connections, and caching application server output. Web servers also allow for application partitioning to permit the placement of application servers behind firewalls or on separate networks.

Specific Web Server areas of analysis include:

- **Architecture**. Application architecture should be designed with security in mind to protect against un-privileged use and unauthorized access. Strong application security will limit externally facing services and protocols to prohibit information disclosure and compromise.
- **Authentication**. Authentication guards against un-privileged access to application functionality. Authentication mechanisms serve to manage access controls, establish the authenticity of a user. Key security concerns include whether credentials are passed in cleartext over a network, how authentication can be integrated into an existing environment (*e.g.*, Active Directory, LDAP).
- **Communication Security**. Communication Security covers the different security options for an external source to communicate with an application securely. Options included message integrity, authentication, encryption and non-repudiation.
- **Information Disclosure**. Information disclosure is the unauthorized presentation of sensitive or confidential information. Proper application design should safeguard against information disclosure to protect both application end users and intellectual property kept by the application. Attackers often utilize information disclosure vulnerabilities to stage more serious attacks.
- **Session Management**. A series of requests to a Web-based application originating from the same user at the same browser is identified by a unique Session Identifier and comprises a session. The Session Identifier can be tracked as cookies or appended to the URL request, and allows only authorized users access to the application.

The next sections provide further information on each area of analysis, and describe @stake's findings in detail.

## 4.4.1 Architecture

---

### Architecture

Application architecture should be designed with security in mind to protect against un-privileged use and unauthorized access. Strong application security will limit externally facing services and protocols to prohibit information disclosure and compromise.

The Architecture area of analysis includes this topic:

- Security Partitioning

The table below describes the topic at a high level, and lists prevailing best practices. The right-hand columns show the weights @stake assigned for each of the scenarios.

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Security Partitioning</b> . Security partitioning is a fundamental security tenant that all secure systems must follow. It is weighted as medium for all application scenarios.	Implementation of security services such as authentication should be cleanly separated from the access control declarations. Web server runs as non-privileged user. Sensitive configuration files are secured from perusal and are never returned to a client web browser.	2	2	2

Scenario weights range from 1 (relatively unimportant) to 3 (extremely important). A scenario weight of zero means that the topic does not apply.

### Findings

The table below summarizes @stake's quantitative findings. For each analysis topic, the two high-level metrics (Best Practice Compliance and Ease of Securing) are shown. These are the arithmetic means of the underlying best practice and test case scores. The Ease of Securing metric is further broken down into four sub-metrics.

Topic	Platform	Best Practice Compliance	Overall	Ease of Securing			
				Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Security Partitioning <i>3 best practices, 4 test cases</i>	.NET Windows	<b>5</b>	<b>4.38</b>	4.75	5	2.75	5
	WebSphere Linux	<b>5</b>	<b>4.19</b>	4.75	4.5	2.5	5
	WebSphere Unix	<b>5</b>	<b>4.19</b>	4.75	4.5	2.5	5

The Architecture area of analysis was comprised of four (4) test cases used to ascertain the ability and support of the platform to operate in partitioned security environments. Scores for both environments were nearly identical. No platform showed a clear superiority in the test cases.

The individual sections for each analysis topic describe @stake's findings in further detail.

## 4.4.1.1 Security Partitioning

---

### Security Partitioning

Security partitioning is a fundamental security tenant that all secure systems must follow. It is weighted as medium for all application scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Security Partitioning topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=21]. Implementation of security services such as authentication should be cleanly separated from the access control declarations.	For simple URL based authorization, the web.config configuration file can be used to store authorization information independent of the built in authentication types: Windows Integrated, Forms based, or Passport. If finer grained authorization is required, IPrincipal and IIdentity objects can be created using the identity from authentication method.	<b>Transparent (5)</b> . For URL based authorization, .NET stores access control declarations in an application specific configuration file, web.config. This allows the developer to change the authentication method employed and still retain independent access control declarations. This feature is limited to the built in authentication types such as Windows Integrated Authentication, Forms based authentication, and Passport. Imperative demand security checks using IPrincipal and IIdentity can be performed independent of the selected authentication store. No code needs to be written to separate authorization information from authentication.	The authorization needs of the application will dictate whether URL based authorization is adequate enough or if the more fine grained authorization of IPrincipal and IIdentity is required. In either case authentication can be easily isolated from authorization allowing a switch of authentication schemes simple.

---

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=22]. Web server runs as non-privileged user.	In the IIS 6.0 web server, the worker processes runs as Network Service, which is a new built-in account with very few privileges. IIS 6.0 also includes new restrictions that will not allow command line tools to be executed by the web server.	<b>Transparent (5)</b> . The web server runs as a non-privileged user by default.	Implement with the default configuration.
[id=23]. Sensitive configuration files are secured from perusal and are never returned to a client web browser.	File access can be controlled at the web server with web site permissions and/or at the OS level with NTFS permissions. Windows 2003 Server also includes the new Authorization Manager that will allow a developer to incorporate role based access controls into applications.	<b>Transparent (5)</b> . Windows 2003 Server provides Discretionary Access Control at the OS level with NTFS permissions and Role Based Access Control at the application level with the Authorization Manager.	Implement both web site permissions and NTFS permissions to protect sensitive configuration files. Use the Authorization Manager to implement role based access control when necessary.

**Best Practice Compliance score**, based on mean of 3 best practices: 5

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=21.1]. Configure security service to use two separate authentication stores, but using the same access control declarations.	Implementation Complexity	<b>Wizard+ (4)</b> . For URL based authentication a configuration file needs to be updated. If you are using IPrincipal role based security a few lines of code may need to be changed to set the Principal from the new authentication method.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation is clear and the samples do not introduce any security issues. There was no explanation of switching authentication methods but it was easy to understand how to do it.
	Implementor Competence	<b>Novice (5)</b> . No code needs to be written to separate authorization information from authentication.
	Time to Implement	<b>Low (5)</b> . No code needs to be written to separate authorization information from authentication.
[id=22.1]. Configure web server to run as non-privileged user.	Implementation Complexity	<b>Wizard (5)</b> . By default, the web server runs as a non-privileged user.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation available locally in the Help and Support Center is clear and easy to follow for security features of IIS 6.0.
	Implementor Competence	<b>Intermediate (3)</b> . The administrator has extensive experience with the Windows OS family. However, the components of the .NET Framework are new; therefore expectations relative to pre-existing knowledge are low.
	Time to Implement	<b>Low (5)</b> . No time is required to implement this feature; it is enabled by default.

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=23.1]. Configure web server to prevent serving configuration files.	Implementation Complexity	<b>Wizard (5)</b> . A simple and clear interface exists for modifying web site permission thought the IIS Manager snap-in.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation available locally in the Help and Support Center is clear and easy to follow for Securing Sites with Web Site Permissions.
	Implementor Competence	<b>Expert/intermediate (2)</b> . The administrator has extensive experience with the Windows OS family. However, the components of the .NET Framework are new; therefore expectations relative to pre-existing knowledge are low.
	Time to Implement	<b>Low (5)</b> . It took less than 5 minutes to search the Help and Support Center and get instructions on how to modify web site permissions.
[id=23.2]. Configure base operating system to protect configuration files.	Implementation Complexity	<b>Wizard (5)</b> . A simple and clear interface exists for modifying NTFS permissions via Windows Explorer. Directory and File permissions can also be implemented in a Security Template and applied with the Security Configuration and Analysis snap-in. The Security Template could also be applied through Group Policy to configure multiple servers.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation available locally in the Help and Support Center is clear and easy to follow for Securing Files with NTFS and Setting NTFS Permissions for Directories or Files.
	Implementor Competence	<b>Expert (1)</b> . The administrator has extensive experience with the Windows OS family.
	Time to Implement	<b>Low (5)</b> . It took less than 5 minutes to search the Help and Support Center and get instructions on how to modify NFTS permissions.

**Ease of Securing score**, based on mean of 4 test cases: 4.38

### Notes

#### Test Case 21.1

Sample web.config authorization section for URL based authorization.

```
<authorization>
  <allow users="Jim Flynn,Mary Tastings">
  <allow roles="Admins">
  <deny users="*">
</authorization>
```

#### Test Case 23.1

Additional information about role based access control and the Authentication Manager can be found in the Help and Support center and the Microsoft document *Building in Security for Applications*.

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=21]. Implementation of security services such as authentication should be cleanly separated from the access control declarations.	Identity stores can be swapped in and out without affecting access controls. Supported identity stores include LDAP, local operating system users/groups, a custom file registry. In addition, any identity store can be used provided it implements a defined Interface.	<b>Transparent (5)</b> . The access control system is completely decoupled from the back-end identity store. Developers do not need to modify applications in any way to suit the needs of a particular authentication system. Declarative roles are mapped to actual users and groups at deployment time.	No recommendations; platform implements best practice.
[id=22]. Web server runs as non-privileged user.	The IBM http Server runs as an unprivileged user by default. The user that the server runs as can be specified through the main web server configuration file.	<b>Transparent (5)</b> . The web server runs as a non-privileged user by default.	Implement the default behavior.
[id=23]. Sensitive configuration files are secured from perusal and are never returned to a client web browser.	Web server configuration files are by default contained in a directory that is outside of the web server root, which prevents the files from being viewed by site visitors.	<b>Transparent (5)</b> . The configuration files are protected by default.	Implement the default behavior.

**Best Practice Compliance scores**, based on mean of 3 best practices: Linux 5, Unix 5

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=21.1]. Configure security service to use two separate authentication stores, but using the same access control declarations.	Implementation Complexity	<b>Wizard (5)</b> . The application server identity store is chosen through a single pull-down menu on the Administration console.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation clearly and concisely explained how to change identity stores. The custom file registry (a "proof of concept") was poorly documented, which is a good thing considering how manifestly insecure this option is.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . The evaluator has significant experience in applications development, but less in systems administration. Configuration was simple and did not require special skills.
	Time to Implement	<b>Low (5)</b> . Changing identity stores required only a property change and a server restart.
[id=22.1]. Configure web server to run as non-privileged user.	Implementation Complexity	<b>Wizard (5)</b> . By default, the web server runs as a non-privileged user.
	Documentation and Examples	<b>Best practice (5)</b> . No documentation is required, but the configuration file is clearly commented to explain the process of running the web server under a specific user context.
	Implementor Competence	<b>Expert/Intermediate (2)</b> . The implementer is familiar with Unix operating systems and security and web administration but is not directly familiar with IBM HTTP server administration.
	Time to Implement	<b>Low (5)</b> . No time is required to implement this feature; it is enabled by default.

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=23.1]. Configure web server to prevent serving configuration files.	Implementation Complexity	<b>Wizard (5)</b> . The configuration file dictates the web document root; the configuration files are not contained in the document root, and are thus not accessible to site visitors.
	Documentation and Examples	<b>Suitable (4)</b> . Adequate comments are provided in the configuration file to describe the appropriate configuration parameter.
	Implementor Competence	<b>Expert/intermediate (2)</b> . The implementer is familiar with Unix operating systems and security and web administration but is not directly familiar with IBM HTTP server administration.
[id=23.2]. Configure base operating system to protect configuration files.	Time to Implement	<b>Low (5)</b> . This is the default configuration; no changes are required.
	Implementation Complexity	<b>Wizard+ (4)</b> . More fully protecting the configuration files for the web server requires changing the permissions of two files.
	Documentation and Examples	<b>Suitable (4)</b> . Online manual (man) pages fully describe the process of applying modifications to file permissions.
	Implementor Competence	<b>Expert/intermediate (2)</b> . The implementer is familiar with Unix operating systems and security and web administration but is not directly familiar with IBM HTTP server administration.
	Time to Implement	<b>Low (5)</b> . The commands can be researched and executed in a few minutes.

**Ease of Securing scores**, based on mean of 4 test cases: Linux 4.19, Unix 4.19

### Notes

#### Test Case 22.1

The configuration file is /opt/IBMHttpServer/conf/httpd.conf. The configuration parameters that control the user and group context under which the web server runs are User and Group, respectively.

#### Test Case 23.1

The configuration file is /opt/IBMHttpServer/conf/httpd.conf. The configuration parameter that controls the web document root is DocumentRoot.

#### Test Case 23.2

Permissions are modified on /opt/IBMHttpServer/conf/admin.conf and /opt/IBMHttpServer/conf/httpd.conf. The initial file permissions are 644; they are changed to 640 so that normal system users cannot read the configuration files.

## 4.4.2 Authentication

---

### Authentication

Authentication guards against un-privileged access to application functionality. Authentication mechanisms serve to manage access controls, establish the authenticity of a user. Key security concerns include whether credentials are passed in cleartext over a network, how authentication can be integrated into an existing environment (*e.g.*, Active Directory, LDAP).

The Authentication area of analysis includes these topics:

- Authentication Input Validation
- Authentication Methods
- Credential Handling
- Digital Certificates
- External Authentication
- Platform Integrated Authentication

The table below describes the topics at a high level, and lists prevailing best practices. The right-hand columns show the weights @stake assigned for each of the scenarios.

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Authentication Input Validation</b> . Credential handling is a requirement for all types of applications so it is weighted uniformly for the three scenarios. It is weighted low for all three scenarios due to its limited impact in the overall security of an application.	User-entered input is verified and cleansed before being accepted as valid. Unconstrained and unchecked data such as escape characters, excessive string lengths or dynamic query strings can lead to serious application compromises. Ideally, the application should only accept meaningful input.	1	1	1
<b>Authentication Methods</b> . Support for web standard authentication is a requirement for all types of applications so it is weighted uniformly for the three scenarios. It is weighted medium for all three scenarios.	Authentication methods are chosen based on the threat profile or security requirements of a given application. If Basic Authentication is used, it must be used in concert with SSL to be secure. Using Basic Auth without communication security is a severe security risk.	2	2	2

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Credential Handling</b> . Credential handling is an important requirement for all types of applications so it is weighted uniformly for the three scenarios.	Passwords are never displayed or stored in clear text, including when being entered by users. All passwords are one-way encrypted (hashed) before storage to prevent them from being compromised. SHA-1 is preferred over MD5. A random value, or salt, should be applied to each password prior to encryption to counter the threat of dictionary attacks. Authentication credentials are not transmitted in plain text.	2	2	2
<b>Digital Certificates</b> . Digital certificates are most important for authentication in unattended application scenarios and authentication models where trust is external or cross-organizational. These properties are both found in web services which are weighted high for digital certificate support. Web applications may also have these properties. This scenario is weighted medium. Digital certificate support is not as important for intranet applications, so it is weighted low.	A secure location stores certificates. Generally this is a protected location in the file system or Windows registry. Certificates are stored in an industry accepted format such as PKCS12 and/or can be accessed through standardized APIs (JCE, CAPI, PKCS11). Certificate authentication can be configured to recognize client certificates issued by named public and private certificate authorities (CAs). Certificates can be mapped to users stored in the application's identity store. Platform supplies certificate revocation checking and verification functionality via industry-standard methods such as certificate revocation lists (CRLs), OCSP, or XKMS.	2	3	1
<b>External Authentication</b> . External authentication is weighted highest for web applications as they need to scale to larger numbers of users and may be hosted in environments where external authentication is a requirement. This is also true of web services. Intranet applications will typically make less use of external authentication.	The application server should provide flexible support for external identity stores such as LDAP servers. Proprietary identity stores (e.g., Active Directory) may also be used, provided the access protocols are well-documented and supported by the industry. If an identity store is used for authentication, binding and lookup operations should be performed using a secure protocol.	3	3	2
<b>Platform Integrated Authentication</b> . Platform integrated authentication is typically easier to implement than external authentication and often provides additional account management features. It is most likely used in intranet applications.	Passwords expire every 30-180 days. Password history prevents re-use of the previous 6-12 passwords	1	1	2

Scenario weights range from 1 (relatively unimportant) to 3 (extremely important). A scenario weight of zero means that the topic does not apply.

## Findings

The table below summarizes @stake's quantitative findings. For each analysis topic, the two high-level metrics (Best Practice Compliance and Ease of Securing) are shown. These are the arithmetic means of the underlying best practice and test case scores. The Ease of Securing metric is further broken down into four sub-metrics.

Topic	Platform	Best Practice Compliance	Ease of Securing				
			Overall	Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Authentication Input Validation <i>1 best practice, 2 test cases</i>	.NET Windows	<b>3</b>	<b>4.13</b>	5	2.5	4	5
	WebSphere Linux	<b>4</b>	<b>3.75</b>	4	2	5	4
	WebSphere Unix	<b>4</b>	<b>3.75</b>	4	2	5	4
Authentication Methods <i>1 best practice, 5 test cases</i>	.NET Windows	<b>4</b>	<b>4.15</b>	4	4	4.6	4
	WebSphere Linux	<b>3</b>	<b>2.85</b>	3	2.4	4.2	1.8
	WebSphere Unix	<b>3</b>	<b>2.85</b>	3	2.4	4.2	1.8
Credential Handling <i>3 best practices, 3 test cases</i>	.NET Windows	<b>3.67</b>	<b>4.08</b>	3.67	4.67	3.67	4.33
	WebSphere Linux	<b>3.67</b>	<b>3.67</b>	4	3.67	3	4
	WebSphere Unix	<b>3.67</b>	<b>3.67</b>	4	3.67	3	4
Digital Certificates <i>3 best practices, 5 test cases</i>	.NET Windows	<b>5</b>	<b>4.65</b>	4.6	4.2	4.8	5
	WebSphere Linux	<b>3</b>	<b>3.75</b>	4.4	3.2	3.8	3.6
	WebSphere Unix	<b>3</b>	<b>3.75</b>	4.4	3.2	3.8	3.6
External Authentication <i>2 best practices, 2 test cases</i>	.NET Windows	<b>5</b>	<b>4.63</b>	5	4.5	4	5
	WebSphere Linux	<b>4</b>	<b>3.88</b>	4.5	4	4	3
	WebSphere Unix	<b>4</b>	<b>3.88</b>	4.5	4	4	3
Platform Integrated Authentication <i>1 best practice, 2 test cases</i>	.NET Windows	<b>5</b>	<b>5</b>	5	5	5	5
	WebSphere Linux	<b>4</b>	<b>3.75</b>	4	3	4	4
	WebSphere Unix	<b>3</b>	<b>3.63</b>	3.5	2.5	5	3.5

This area of analysis covered a broad range of authentication issues common to all web and web service based applications. Industry studies and @stake experience have shown that the difficulties and subtleties of implementing robust access control mechanisms are often lost on typical application developers. Given the scope of the importance of authentication mechanisms in sound application design, @stake expected that both platforms would perform well under evaluations. The relatively close scores in the section of the analysis bear this fact out.

The individual sections for each analysis topic describe @stake's findings in further detail.

## 4.4.2.1 Authentication Input Validation

---

### Authentication Input Validation

Credential handling is a requirement for all types of applications so it is weighted uniformly for the three scenarios. It is weighted low for all three scenarios due to its limited impact in the overall security of an application. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 1
- Web Service: 1
- Intranet: 1

The sections that follow contain the results of @stake's analysis of the Authentication Input Validation topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=5]. User-entered input is verified and cleansed before being accepted as valid. Unconstrained and unchecked data such as escape characters, excessive string lengths or dynamic query strings can lead to serious application compromises. Ideally, the application should only accept meaningful input.	Several options are available to validate user input: using the validators provided by the framework, using the regular expression engine, and implementing a manual validation routine.	<b>Developer extends (3)</b> . At best, the user input validation still needs to be configured.	Although documentation provided by Microsoft sites does not describe best practices for authentication input validation, the .NET Framework provides all necessary building blocks to do so.

**Best Practice Compliance score**, based on mean of 1 best practice: 3

### Level of Effort Analysis

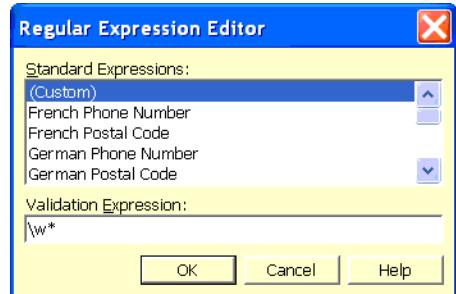
Test Case	Criteria	Rating
[id=5.1]. Create authentication widget that only accepts alpha-numeric usernames.	Implementation Complexity	<b>Wizard (5)</b> . Essentially, using VS.net controls and property sheet.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . No single documentation on this topic.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited .NET exposure.
	Time to Implement	<b>Low (5)</b> . Only a few operations are needed to create a validator.
[id=5.2]. Create authentication widget that constrains the length of usernames and password.	Implementation Complexity	<b>Wizard (5)</b> . This can easily be achieved with a validator object
	Documentation and Examples	<b>Adequate (3)</b> . A range validator is documented but no best practice information is available.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited .NET exposure.
	Time to Implement	<b>Low (5)</b> . As can be seen below, creating a validator is simple.

Ease of Securing score, based on mean of 2 test cases: 4.13

### Notes

#### Test Case 5.1

The validation expression property does not provide for simple expressions such as alphanumeric or numeric strings. This could be improved to add more than phone numbers and zip codes.



See also: .NET Framework QuickStarts - .NET Samples - ASP.NET Security

#### Test Case 5.2

One could have implemented this with a custom validator or regular expression object. Both cases are well documented and do not present any complexity.

The regular expression syntax could use more samples.

If the application wants to limit the username between 2 and 10 characters, the regular expression is:

".{2,10}"

Implementing this feature with a custom validator could be done as follows:

```
Private Sub CustomValidator1_ServerValidate(ByVal source As System.Object
    ,ByVal args As System.Web.UI.WebControls.ServerValidateEventArgs)
Handles CustomValidator1.ServerValidate
If (Len(Username.Text) > 2 And Len(Username.Text) < 10) Then
    args.IsValid = False
Else
    args.IsValid = True
End If
End Sub
```

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=5]. User-entered input is verified and cleansed before being accepted as valid. Unconstrained and unchecked data such as escape characters, excessive string lengths or dynamic query strings can lead to serious application compromises. Ideally, the application should only accept meaningful input.	Using the Struts Validation routines included with WebSphere 5.0 creating constrained authentication widgets is a fairly straight forward endeavor. By extending previously created classes and configurations it was possible to accomplish this best-practice in short order	<b>Wizard (4)</b> . By taking advantage of an application written in a previous test case, implementing the two test cases of this best practice was a simple exercise of modifying one XML configuration file containing input validation rules.	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 4, Unix 4

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=5.1]. Create authentication widget that only accepts alpha-numeric usernames.	Implementation Complexity	<b>Wizard+ (4)</b> . By taking advantage of an application written in a previous test case, the implementation of this best practice was a simple exercise of modifying one XML configuration file containing input validation rules.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . The vendor supplied documentation on Struts Validators was scant and had an enormous learning curve. The implementer was expected to possess a profound knowledge of the Apache Struts environment and comprehensive understanding of the WebSphere deployment environment. The documentation did not provide information that could lead to insecure designs and deployment environments, its overall uselessness warranted this rating.
	Implementor Competence	<b>Novice (5)</b> . The tester had no prior Struts implementation experience. After six hours with the documentation, it was possible to use the functionality to create fairly sophisticated user input filtering routines.
	Time to Implement	<b>Low to Medium (4)</b> . Extending the existing <code>validator.xml</code> file for a previously created application took less than fifteen minutes

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=5.2]. Create authentication widget that constrains the length of usernames and password.	Implementation Complexity	<b>Wizard+ (4)</b> . By taking advantage of an application written in a previous test case, the implementation of this best practice was a simple exercise of modifying one XML configuration file containing input validation rules.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . The vendor supplied documentation on Struts Validators was scant and had an enormous learning curve. The implementer was expected to possess a profound knowledge of the Apache Struts environment and comprehensive understanding of the WebSphere deployment environment. The documentation did not provide information that could lead to insecure designs and deployment environments, its overall uselessness warranted this rating.
	Implementor Competence	<b>Novice (5)</b> . The tester had no prior Struts implementation experience. After six hours with the documentation, it was possible to use the functionality to create fairly sophisticated user input filtering routines.
	Time to Implement	<b>Low to Medium (4)</b> . Extending the existing <code>validator.xml</code> file for a previously created application took less than fifteen minutes.

**Ease of Securing scores**, based on mean of 2 test cases: Linux 3.75, Unix 3.75

### Notes

#### Test Case 5.1

For the sake of brevity, only modifications to the previously created `validator.xml` configuration file is included below. The following modifications were made to the validation rules:

- The dependency rules for existing, pattern matching and size limitations are applied in the order specified. This order is arbitrary.
- A regular expression pattern allowing only alpha-numeric characters is created as a mask `<var>` validator configuration directive. **NB:**It is theoretically possible to constrain the length of a given input pattern with a regular expression, however this was deemed too expensive and would not have best illustrated the test case.
- A numeric constant to limit the size of an acceptable password is define as a `<var> maxlength` validator configuration directive

```

<field property="password"
    depends="required,mask,maxlength">
    <msg name="mask" key="logonForm.password.maskmsg"/>
    <arg0 key="logonForm.password.displayname"/>
    <arg1 name="maxlength" key="${var:maxlength}" resource="false"/>
    <var>
        <var-name>mask</var-name>
        <var-value>^[a-zA-Z][0-9]*$</var-value>
    </var>
    <var>
        <var-name>maxlength</var-name>
        <var-value>8</var-value>
    </var>

```

```
</field>
```

### Test Case 5.2

The requirements of this test case were accomplished simultaneously with the previous test case. The ratings and assessment were made as if both task were done separately.

## 4.4.2.2 Authentication Methods

---

### Authentication Methods

Support for web standard authentication is a requirement for all types of applications so it is weighted uniformly for the three scenarios. It is weighted medium for all three scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Authentication Methods topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=6]. Authentication methods are chosen based on the threat profile or security requirements of a given application. If Basic Authentication is used, it must be used in concert with SSL to be secure. Using Basic Auth without communication security is a severe security risk.	A number of the authentication options are available: Basic, Digest, Passport, Certificate-based, and Integrated Windows authentication.	<b>Wizard (4)</b> . Microsoft provides all the relevant information and functionality to select the most appropriate authentication method.	No recommendation.

**Best Practice Compliance score**, based on mean of 1 best practice: 4

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=6.1]. Create authentication widget that uses Basic Authentication (HTTPBasic) and SSL.	Implementation Complexity	<b>Wizard+ (4)</b> . Mostly driven by wizards with the exception of the actual vendor certificate request.
	Documentation and Examples	<b>Suitable (4)</b> . Microsoft has an extensive article on authentication alternatives and how to determine the most appropriate option given the context.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited .NET exposure.
	Time to Implement	<b>Low to Medium (4)</b> . Minimal time is needed to specify an authentication method.
[id=6.2]. Create authentication widget that uses Digest Authentication (HTTPDigest).	Implementation Complexity	<b>Wizard (5)</b> . This is part of IIS 6's Directory Security settings.
	Documentation and Examples	<b>Adequate (3)</b> . The configuration panel warns appropriately about the limitations of Digest authentication.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited .NET exposure.
	Time to Implement	<b>Low (5)</b> . A single checkbox is all that's needed to configure this option.
[id=6.3]. Create authentication widget that uses Forms-Based Authentication and SSL.	Implementation Complexity	<b>Small amount of code (3)</b> . Although no single wizard helps with this functionality, the amount of available documentation and building blocks compensates for its lack of automated assistance.
	Documentation and Examples	<b>Best practice (5)</b> . Configuring an SSL server is a well documented process.
	Implementor Competence	<b>Novice (5)</b> . The developer has wide-ranging Windows and web development experience with limited .NET exposure.
	Time to Implement	<b>Medium (3)</b> . A number of steps need to be performed in order to configure a working SSL server. No wizard is available to configure Forms-based authentication.
[id=6.4]. Create authentication widget that uses Integrated host OS Authentication.	Implementation Complexity	<b>Wizard (5)</b> . A checkbox provides for integrated authentication and is configured by default.
	Documentation and Examples	<b>Suitable (4)</b> . The documentation is clear but does not provide enough information about how to retrieve a user's credentials via ASP.NET.
	Implementor Competence	<b>Novice (5)</b> . The developer has wide-ranging Windows and web development experience with limited .NET exposure.
	Time to Implement	<b>Low (5)</b> . No changes needed to be made.

### Level of Effort Analysis

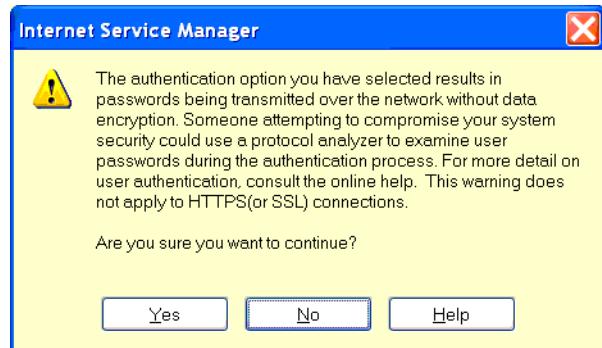
Test Case	Criteria	Rating
[id=6.5]. Create authentication widget that uses Certificate Based Authentication.	Implementation Complexity	<b>Small amount of code (3)</b> . Turning on the client certificates requirement is done through the settings wizards. The client certification processing proved to have a higher level of complexity: user mappings, programmatic access...
	Documentation and Examples	<b>Suitable (4)</b> . See related link for step-by-step setup and usage of client certificates.
	Implementor Competence	<b>Novice (5)</b> . The developer has wide-ranging Windows and web development experience with limited .NET exposure.
	Time to Implement	<b>Medium (3)</b> . Creating a client certificate was the longest part of this test case.

**Ease of Securing score**, based on mean of 5 test cases: 4.15

### Notes

#### Test Case 6.1

Forcing the web server to use basic authentication is straight forward and appropriately presents the following warning:



Accessing the credentials from asp.NET code is as easy as:

```
Auth Type=<%=Request( "AUTH_TYPE" )%>
Auth User=<%=Request( "AUTH_USER" )%>
Auth Password=<%=Request( "AUTH_PASSWORD" )%>
```

Yielding: Auth Type=Basic

Auth User=FREDDY\Test account

Auth Password=Password

See also: Authentication in ASP.NET: .NET Security Guidance.

### Test Case 6.2

Digest authentication is only available when:

- The user and the server running IIS must be members of, or be trusted by, the same domain.
- Users must have a valid Windows user account stored in Active Directory® on the domain controller.
- The domain must have a Windows 2000 or later domain controller.

This does limit deployment of this authentication method. For example public sites or sites using a database to store user accounts can not benefit from digest authentication.

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=6]. Authentication methods are chosen based on the threat profile or security requirements of a given application. If Basic Authentication is used, it must be used in concert with SSL to be secure. Using Basic Auth without communication security is a severe security risk.	The IBM HTTP Server supports a number of authentication mechanisms: basic, digest, LDAP, and client certificate. Integrated host OS authentication is not supported although modules that implement it are available on the Internet.	<b>Developer extends (3)</b> . Though many authentication mechanisms are available, most require moderate to extensive configuration.	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 3, Unix 3

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=6.1]. Create authentication widget that uses Basic Authentication (HTTPBasic) and SSL.	Implementation Complexity	<b>Wizard+ (4)</b> . Broad access control at the directory level can be accomplished with the administrative interface to the IBM HTTP Server, but more granular control requires configuration file changes or creation of .htaccess files. SSL must be configured separately.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . The documentation describes how to use the Administration Server to define password files and create users, but for setting authentication rules the IBM documentation makes an out of context reference to Apache documentation, which is potentially confusing. The Administration Server does support the creation of authentication rules in a broad sense, though this does not appear to be documented.
	Implementor Competence	<b>Novice/intermediate (4)</b> . This requires some understanding of web server configuration, particularly if granular access controls are desired.
	Time to Implement	<b>Medium to High (2)</b> . The implementation time is for a first time configuration, and is related to the quality of the documentation; subsequent implementations would be much more rapid.

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=6.2]. Create authentication widget that uses Digest Authentication (HTTPDigest).	Implementation Complexity	<b>Small amount of code (3)</b> . Enabling Digest Authentication requires manually adding some lines to the web server configuration file. More granular usage of Digest Authentication would require the creation of .htaccess files. The complexity was increased due to the lack of some necessary components in the IBM installation.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Digest Authentication is briefly mentioned in some documentation, such as the IBM WebSphere V5.0 Security Redbook, but no details are provided on how to configure it and no references are provided to related Apache documentation.
	Implementor Competence	<b>Novice/intermediate (4)</b> . This requires some understanding of web server configuration, particularly if granular access controls are desired.
	Time to Implement	<b>Medium to High (2)</b> . The implementation time is for a first time configuration, and is related to the quality of the documentation and presence of needed components; subsequent implementations would be much more rapid.
[id=6.3]. Create authentication widget that uses Forms-Based Authentication and SSL.	Implementation Complexity	<b>Wizard+ (4)</b> . This can be implemented using either of two tools, the Application Assembly Tool or WebSphere Studio, where the resources to be protected are selected and a login form is specified. In addition to performing the basic configuration with one of these tools, the implementation requires creation of the appropriate form, which can be a simple HTML or JSP file. The documentation notes that the credentials are passed in plaintext and recommends using SSL. No coding is required.
	Documentation and Examples	<b>Suitable (4)</b> . The documentation covers the configuration changes required, and provides an example of a suitable login page.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer needs a basic understanding of Web based forms.
	Time to Implement	<b>Medium (3)</b> . Implementation is fairly rapid, with the requirement of creating the appropriate form page taking the most time.
[id=6.4]. Create authentication widget that uses Integrated host OS Authentication.	Implementation Complexity	<b>Large amount of code (1)</b> . Implementing host OS authentication on the IBM HTTP Server requires integrating a free, third-party module.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . IBM's documentation does not appear to mention how to achieve host OS authentication.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering IBM HTTP Server/Apache web servers.
	Time to Implement	<b>High (1)</b> . The time required to build and integrate OS authentication into the IBM HTTP Server would definitely be high.

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=6.5]. Create authentication widget that uses Certificate Based Authentication.	Implementation Complexity	<b>Small amount of code (3)</b> . There is no code to write in order to create a client certificate but a number of command-line utilities must be executed.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . Although adequate documentation exists regarding enabling client certificate authentication, none of it covers the creation and self-signing of client certificates, which is important since the certificate lies in the ikeyman key store.
	Implementor Competence	<b>Novice (5)</b> . The developer is a novice with respect to PKI and the OpenSSL tools in particular.
	Time to Implement	<b>High (1)</b> . A significant amount of time was spent learning about the OpenSSL suite and attempting to integrate it with ikeyman.

**Ease of Securing scores**, based on mean of 5 test cases: Linux 2.85, Unix 2.85

### Notes

#### Test Case 6.1

All ratings assume that the developer is not already familiar with Apache.

#### Test Case 6.2

The Apache Documentation that is included with the IBM HTTP Server does provide information on how to configure Digest Authentication, but there is no reference to this documentation in the main body of IBM documentation and an implementer would need to know which modules to look for. A developer already familiar with Apache would have much less difficulty finding the appropriate information.

Apache supports Digest Authentication with two modules. The module mod\_digest supports an older version of the standard, and the module mod\_auth\_digest supports the most recent version of the standard but is marked experimental. The IBM HTTP Server ships only with mod\_digest.

In order to configure the web server to use Digest Authentication, the following lines were added to a Directory block within the configuration file:

```
AuthType digest
AuthDigestFile /opt/IBMHttpServer/conf/digest-users.passwd
```

The following lines were required to load the appropriate module:

```
LoadModule digest_module libexec/mod_digest.so
AddModule mod_digest.c
```

(Note that the loading of the appropriate modules could have been configured using the Administration Server, though this does not appear true for the configuration of Digest Authentication in general).

The final step for configuring digest authentication involves creating a file creating the appropriate users and passwords. This is accomplished using the htdigest tool, which does not appear to install by default with the IBM HTTP Server. The htdigest tool from the Red Hat system's default Apache installation was used instead.

### Test Case 6.3

Forms based authentication is handled at the application server rather than directly by the web server.

The following is an example of an extremely basic form that would allow for form based authentication:

```
<form method="post" action="/application/j_security_check">
  Userid: <input size="20" type="text" name="j_username" maxlength="25"><br>
  Password:<input size="20" type="password" name="j_password" maxlength="25"><br>
  <input type="submit" name="action" value="Login">
</form>
```

### Test Case 6.5

The creation of client certificates is fairly involved. This test simulated an enterprise wanting to sign its own certificates. Purchasing certificates from a commercial CA should make this process a lot smoother. At a high-level, the following steps need to be performed:

- Create a self-signed certificate (henceforth called atstake cert) with ikeyman and verify that the IBM HTTP Server operates correctly (this was done in test case 18).
- Configure the IBM HTTP Server to require client certificate authentication.
- Export the atstake cert as a pkcs12 certificate using ikeyman.
- Convert the atstake pkcs12 cert to a .pem format certificate using OpenSSL.
- Configure OpenSSL to use the atstake .pem cert's public and private keys as its CA keys.
- Use OpenSSL to create a certificate request (henceforth called user cert).
- Use OpenSSL to sign the user cert with the atstake cert.
- Use OpenSSL to convert the resulting .pem user cert into the pkcs12 format.
- Import the pkcs12 user cert into Internet Explorer and use it to access the web site.

See also: Authenticating clients: how to enable client certification authentication on the IBM HTTP Server. OpenSSL documentation: this documentation is also installed on the Red Hat Advanced Server.

#### 4.4.2.3 Credential Handling

---

### Credential Handling

Credential handling is an important requirement for all types of applications so it is weighted uniformly for the three scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Credential Handling topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=2]. Passwords are never displayed or stored in clear text, including when being entered by users.	Passwords can be obscured via the HTML password field, and via the username/password window associated with Basic, Digest, and Passport authentication.	<b>Developer extends (3)</b> . No wizard is available to create an authentication form, it has to be manually coded.	No recommendation.
[id=3]. All passwords are one-way encrypted (hashed) before storage to prevent them from being compromised. SHA-1 is preferred over MD5. A random value, or salt, should be applied to each password prior to encryption to counter the threat of dictionary attacks.	The cryptographic hash functions and RNG utilities exposed by .NET support the proper handling of authentication credentials.	<b>Developer extends (3)</b> . A small amount of code is required to correctly hash user passwords.	
[id=4]. Authentication credentials are not transmitted in plain text.	A number of the authentication options (Digest, Passport, Certificate-based, etc.) available do not transmit passwords in the clear. Microsoft warns the administrator appropriately if he/she chooses configuration settings resulting in clear text credentials.	<b>Transparent (5)</b> . .NET meets all security best practices in this area.	

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
<b>Best Practice Compliance score</b> , based on mean of 3 best practices: 3.67			

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=2.1]. Create authentication widget that properly obscures password input field.	Implementation Complexity	<b>Small amount of code (3)</b> . A small amount of code and configuration options must be specified.
	Documentation and Examples	<b>Best practice (5)</b> . Implementation of a simple forms-based authentication is well documented and fosters security best practices.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited .NET exposure.
	Time to Implement	<b>Low to Medium (4)</b> . A short amount of time was spent implementing this test case.
[id=3.1]. Create and a hash of a password or phrase.	Implementation Complexity	<b>Small amount of code (3)</b> . The implementation requires a straightforward use of a cryptographic hash function via the SHA1CryptoServiceProvider and the System.Random class.
	Documentation and Examples	<b>Suitable (4)</b> . The documentation clearly describes how to use the Random class and SHA1CryptoServiceProvider. No explanation is given regarding the proper handling of user passwords using these utilities.
	Implementor Competence	<b>Intermediate (3)</b> . The developer has extensive experience with cryptographic hash and RNG functions, but has not worked extensively with Microsoft implementations.
	Time to Implement	<b>Low to Medium (4)</b> . The implementation is a simple combination of two system utilities.
[id=4.1]. Ensure that built-in authentication functions do not transmit credentials in the clear.	Implementation Complexity	<b>Wizard (5)</b> . Configured securely out of the box.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation is clear and mentions the best practices in this area.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited .NET exposure.
	Time to Implement	<b>Low (5)</b> . No changes needed to be made.

**Ease of Securing score**, based on mean of 3 test cases: 4.08

### Notes

#### Test Case 2.1

At a high level, the steps involve:

- Setting web.config to handle forms authentication and deny anybody.
- Creating a web form with a username, password and submit button.

- Implementing the submit event to actually validate the credentials.

The following link could have included a regular expressions validator to ensure the password was within the appropriate boundaries, such as length and character space.

It could also have used `Page.IsValid()` to ensure server side validation of the input fields is performed.

See also: .NET Framework Developer's Guide - Simple Forms Authentication

### Test Case 3.1

See `hash_functions.cs` for the source.

### Test Case 4.1

Using the MSDN guidelines for selecting an authentication mechanism should never yield clear text credentials handling.

See also: Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=2]. Passwords are never displayed or stored in clear text, including when being entered by users.	Support for the <code>password</code> text attribute is natively supported by the IBM HTTP Server. Any input field using this attribute as a password entry is automatically masked.	<b>Transparent (5)</b> . Support for this best practice is transparently supported by the platform	No recommendation.
[id=3]. All passwords are one-way encrypted (hashed) before storage to prevent them from being compromised. SHA-1 is preferred over MD5. A random value, or salt, should be applied to each password prior to encryption to counter the threat of dictionary attacks.	The cryptographic hash functions and RNG utilities exposed by J2EE support the proper handling of authentication credentials.	<b>Developer extends (3)</b> . A small amount of code is required to correctly hash user passwords.	
[id=4]. Authentication credentials are not transmitted in plain text.	Through its Administrative Console, the IBM HTTP Server supports Basic and LDAP authentication. A digest authentication module is also included in the libexec directory but isn't accessible via the administrative interface. Certificate authentication is covered in the next section.	<b>Developer extends (3)</b> . The IBM documentation does not mention the lack of credential encryption for Basic and LDAP authentication.	Ensure that Basic and LDAP authentication are only used with an SSL-enabled connection. Additionally, use <code>mod_ibm_ldap128</code> to SSL-encrypt lookups to the LDAP store.

**Best Practice Compliance scores**, based on mean of 3 best practices: Linux 3.67, Unix 3.67

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=2.1]. Create authentication widget that properly obscures password input field.	Implementation Complexity	<b>Wizard (5)</b> . There is effectively no additional complexity added by complying with this best practice
	Documentation and Examples	<b>Best practice (5)</b> . No specific documentation had to be referenced in order to comply with this best practice. In fairness to the platform under evaluation, this functionality is supplied by the underlying HTML standard and the vendor implementation, not a specific function or procedure. There are literally thousands of documentation sources on the Internet that describe how to implement the "masking" of password fields in HTML forms.
	Implementor Competence	<b>Intermediate (3)</b> . The tester has wide experience in configuring and deploying web based applications.
	Time to Implement	<b>Low (5)</b> . No modifications to the Web Server were required to implement this best practice.
[id=3.1]. Create and a hash of a password or phrase.	Implementation Complexity	<b>Small amount of code (3)</b> . The implementation requires a straightforward use of a cryptographic hash function via the Message Digest object and the java.util.Random class.
	Documentation and Examples	<b>Suitable (4)</b> . The documentation clearly describes how to use the java.util.Random class and Message Digest object. No explanation is given regarding the proper handling of user passwords using these utilities.
	Implementor Competence	<b>Expert/intermediate (2)</b> . The developer has extensive experience with cryptographic hash and RNG functions and has worked previously with Java implementations.
	Time to Implement	<b>Low to Medium (4)</b> . The implementation is a simple combination of two system utilities.
[id=4.1]. Ensure that built-in authentication functions do not transmit credentials in the clear.	Implementation Complexity	<b>Wizard+ (4)</b> . Enabling Basic authentication is simple. Activating LDAP authentication requires specifying and editing a configuration file.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . IBM's documentation for Basic authentication links to the Apache site. While some mention of security is made there, nowhere is it explained that credentials will be sent in plaintext. IBM's LDAP authentication is fairly basic and also doesn't mention that credentials are sent in plaintext unless SSL-secured.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering IBM HTTP Server/Apache web servers.
	Time to Implement	<b>Medium (3)</b> . Enabling Basic authentication is simple but LDAP requires more configuration work.

**Ease of Securing scores**, based on mean of 3 test cases: Linux 3.67, Unix 3.67

### Notes

#### Test Case 2.1

The example below illustrated the use of a simple HTML form to accept a user name and password combination. Note the use of `html:password` element on line 9.

```
<html:form action="/logonsubmit" focus="username">
  <table border="0" width="100%">
    <tr>
      <th align="right">username:</th>
      <td align="left"><html:text property="username"/></td>
    </tr>
    <tr>
      <th align="right">password:</th>
      <td align="left"><html:password property="password"/></td>
    </tr>
    <tr>
      <td align="right"><html:submit/></td>
      <td align="left"><html:reset/></td>
    </tr>
  </table>
</html:form>
```

## Documentation Sources

Information regarding the use of text attributes in HTML INPUT Tags is found in section 17.4.1 Control types created with INPUT of the HTML 4.01 Specification>

## Test Case 3.1

See hash\_functions.java for the full source.

#### 4.4.2.4 Digital Certificates

---

### Digital Certificates

Digital certificates are most important for authentication in unattended application scenarios and authentication models where trust is external or cross-organizational. These properties are both found in web services which are weighted high for digital certificate support. Web applications may also have these properties. This scenario is weighted medium. Digital certificate support is not as important for intranet applications, so it is weighted low. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 3
- Intranet: 1

The sections that follow contain the results of @stake's analysis of the Digital Certificates topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=8]. A secure location stores certificates. Generally this is a protected location in the file system or Windows registry. Certificates are stored in an industry accepted format such as PKCS12 and/or can be accessed through standardized APIs (JCE, CAPI, PKCS11).		<b>Transparent (5)</b> . Certificates are stored using the certificate import wizard. The user opens the certificate file, selects "import certificate" and steps through the wizard. The wizard default settings installs the certificate in the most appropriate place. The certificate store is protected by operating system ACLs so that only administrative users can access the store. Certificates can be stored using base-64 encoded x.509 or PKCS #12 format. A command line interface, certmgr.exe, is also available.	Most secure applications will require a server certificate installed. This can be done with the certificate wizard. If user certificates are used for authentication a certificate authority certificate may need to be installed. This can also be done with the certificate wizard.

---

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=9]. Certificate authentication can be configured to recognize client certificates issued by named public and private certificate authorities (CAs). Certificates can be mapped to users stored in the application's identity store.	IIS provides 3 different client certificate mapping options. The first is to use Directory Service mapping. This uses Active Directory features to authenticate client certificates. This option is simple to use and offers one to one mapping between users with account information in Active Directory and their certificates which are stored there. The second option is one to one mapping. This maps a client certificate to a particular user account. You must have the user's client certificate file available and the user's account password. The third is one to many mapping. This mapping lets the administrator select a create a rule that matches part of a certificate such as company name to a string.	<b>Transparent (5)</b> . Client certificate configuration and mapping is configured with a wizard. The options available allow the administrator to easily configure most mapping scenarios.	
[id=10]. Platform supplies certificate revocation checking and verification functionality via industry-standard methods such as certificate revocation lists (CRLs), OCSP, or XKMS.		<b>Transparent (5)</b> . The platform checked CRLs by default at the distribution point listing in the CA certificate.	

**Best Practice Compliance score**, based on mean of 3 best practices: 5

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=8.1]. Store a certificate in a secure manner.	Implementation Complexity	<b>Wizard (5)</b> . All certificate operations could be done with the certificate installation wizard.
	Documentation and Examples	<b>Best practice (5)</b> . The help documentation for the web server had clear instructions for installing certificates.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The test case implementer has wide experience
	Time to Implement	<b>Low (5)</b> . It took less than 10 minutes to find the correct information and use the wizard to store a certificate.
[id=8.2]. Ensure that the certificate store is properly protected via container or operating system file permissions.	Implementation Complexity	<b>Wizard+ (4)</b> . Secure storage was verified by checking the permissions on the file where the certificates were stored.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . The documentation of where the certificates were stored was vague. To verify the exact location file system monitoring tools needed to be used to ascertain the exact location.
	Implementor Competence	<b>Novice (5)</b> . Not applicable.
	Time to Implement	<b>Low (5)</b> . Not applicable.

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=9.1]. Create a user in external credential store (e.g., Active Directory or LDAP).	Implementation Complexity	<b>Wizard (5)</b> . Users are created through the Microsoft Management Console. The dialogs were easy to use.
	Documentation and Examples	<b>Best practice (5)</b> . The online help was easy to understand but was not necessary to complete this task.
	Implementor Competence	<b>Novice (5)</b> . No developer skills required.
	Time to Implement	<b>Low (5)</b> . No implementation necessary.
[id=9.2]. Create an x.509 certificate, and configure the container to recognize the certificate.	Implementation Complexity	<b>Wizard+ (4)</b> . A multi-step wizard was available to guide the administrator through this process. It was fairly complex to understand. The option of mapping one to one to a user's certificate required access to the certificate file and knowing the user's account password which may not be readily available.
	Documentation and Examples	<b>Suitable (4)</b> . Documentation was available in the online help system. A few different pages needed to be consulted and understood.
	Implementor Competence	<b>Novice (5)</b> . No developer skills required.
	Time to Implement	<b>Low (5)</b> . No implementation required.
[id=10.1]. Configure certificate revocation. Attempt to login with a revoked certificate verify that it is rejected.	Implementation Complexity	<b>Wizard (5)</b> . Certificate revocation was enabled by default.
	Documentation and Examples	<b>Best practice (5)</b> . No documentation needed to be consulted to accomplish the task. The documentation of certificate services explained CRLs clearly.
	Implementor Competence	<b>Novice (5)</b> . No developer skills required.
	Time to Implement	<b>Low (5)</b> . Nothing to implement.

**Ease of Securing score**, based on mean of 5 test cases: 4.65

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=8]. A secure location stores certificates. Generally this is a protected location in the file system or Windows registry. Certificates are stored in an industry accepted format such as PKCS12 and/or can be accessed through standardized APIs (JCE, CAPI, PKCS11).	The IBM HTTP Server stores certificates in an encrypted file called key.kds that is typically located in /opt/IBMHttpServer/bin. By default, this file belongs to root and is only readable/writable by this user. The bin directory and its folder hierarchy also belong to root.	<b>Wizard (4)</b> . The key database file permissions are as secure as possible, however the developer is withholding the maximum rating given the weak encryption of the stash file.	Ensure that the permissions on the key files aren't weakened, otherwise an attacker could easily decrypt the key database (see below).

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=9]. Certificate authentication can be configured to recognize client certificates issued by named public and private certificate authorities (CAs). Certificates can be mapped to users stored in the application's identity store.	Users are created using the tools associated with the identity store being used. This will typically be LDAP, although the operating system user registry tools can also be used as an alternative. Key management for authentication is straightforward and implemented via a GUI.	<b>Wizard (4)</b> . User management is completely delegated to the identity store, although the authentication subsystem (which leverages the identity store) is not turned on by default.	Users should enable <i>Global Security</i> at their first opportunity. For an easy-to-use but secure initial security posture, IBM should configure WebSphere to use local OS authentication out of the box.
[id=10]. Platform supplies certificate revocation checking and verification functionality via industry-standard methods such as certificate revocation lists (CRLs), OCSP, or XKMS.	Certificate revocation is configurable on the HTTPD server, but not on the application server. It is unclear how CRLs are obtained and processed.	<b>Not possible (1)</b> . CRL checking for client certificates, simply put, could not be made to work. The documentation states that HTTPD looks up CRLs in an LDAP directory. Passwords used to bind to the LDAP server are stored in an encrypted "stash file."	IBM's documentation was appallingly bad. Certificate revocation is important for web services (machine-to-machine) communications. IBM should fix this problem, or else recommend that customers use the popular (and much better documented) mod_ssl as a replacement for IBM's SSL package.

**Best Practice Compliance scores**, based on mean of 3 best practices: Linux 3, Unix 3

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=8.1]. Store a certificate in a secure manner.	Implementation Complexity	<b>Wizard (5)</b> . Creating and/or storing a certificate is performed through the ikeyman graphical utility.
	Documentation and Examples	<b>Suitable (4)</b> . The ikeyman documentation steps the user through all the ikeyman functionality.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering IBM HTTP Server/Apache web servers.
	Time to Implement	<b>Low to Medium (4)</b> . Use of ikeyman is very simple. The time noted here mainly consists of reading the relevant parts of the documentation.
[id=8.2]. Ensure that the certificate store is properly protected via container or operating system file permissions.	Implementation Complexity	<b>Wizard (5)</b> . The default permissions on key.kdb and its related files are sufficient to prevent unauthorized access to the key file.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . The documentation does not cover the security of the certificate database files. Although the file permissions are secure by default, the documentation should mention how to maintain this security.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering IBM HTTP Server/Apache web servers.
	Time to Implement	<b>Low (5)</b> . No changes need be made.

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=9.1]. Create a user in external credential store (e.g., Active Directory or LDAP).	Implementation Complexity	<b>Wizard (5)</b> . We used an external LDAP server (iPlanet) as the identity store. Adding a user to iPlanet was accomplished through a simple dialog box.
	Documentation and Examples	<b>Best practice (5)</b> . The on-line iPlanet documentation was clear and complete.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The evaluator has limited systems administration experience. A rudimentary understanding of LDAP should be assumed.
	Time to Implement	<b>Low (5)</b> . Adding a user took less than a minute.
[id=9.2]. Create an x.509 certificate, and configure the container to recognize the certificate.	Implementation Complexity	<b>Wizard (5)</b> . IBM's iKeyMan tool (a stand-alone Java Swing application) allows stand-alone key files to be created. There are two versions of this tool, one each for IBM HTTPD (the web server) and WebSphere (the application server) due to slight differences at the binary level in key store formats; the interfaces are identical.
	Documentation and Examples	<b>Suitable (4)</b> . The documentation provides numerous examples on how to create self-signed certificates for clients and servers, and instructions on how to map user identities in the certificate to those in an LDAP identity store.
	Implementor Competence	<b>Novice/intermediate (4)</b> . This evaluator is a relatively inexperienced J2EE server administrator, but has broad general knowledge of PKI concepts.
	Time to Implement	<b>Medium (3)</b> . Configuring the application server's trust store to accept client certificates was trivial, not more than a minute. The certificate mapping took some time to troubleshoot.
[id=10.1]. Configure certificate revocation. Attempt to login with a revoked certificate verify that it is rejected.	Implementation Complexity	<b>Medium amount of code (2)</b> . IBM's SSL package supposedly supports CRL checking, but we could not make it work. Ralf Engelschall's mod_ssl package, however, works well as an alternative.
	Documentation and Examples	<b>Incorrect or insecure (1)</b> . Documentation was incomplete to the point of being completely useless.
	Implementor Competence	<b>Intermediate (3)</b> . This reviewer has approximately three years of using web server technologies, including Apache configuration.
	Time to Implement	<b>High (1)</b> . This reviewer exceeded the maximum amount of time allowed to implement CRL checking. If mod_ssl had been used instead, the rating would have been a three .

**Ease of Securing scores**, based on mean of 5 test cases: Linux 3.75, Unix 3.75

### Notes

#### Test Case 8.2

In order for the IBM HTTP Server to read the key.kdb file, it needs to know the key.kdb's encryption password. This information is available in key.sth (the password stash file created by ikeyman) and is read by the IBM HTTP Server. The stash file looks encrypted but is simply obfuscated with an XOR cipher. IBM could have implemented this more securely, although (assuming the key.sth permissions aren't weakened) an

attacker able to read the stash file would already have root access.

See also: Stash file password cracker.

### Test Case 9.1

For this analysis, the choice of identity store for use with WebSphere is an arbitrary one. Although we used iPlanet, we could have just as easily used IBM SecureWay or Active Directory.

### Test Case 9.2

This test case focused primarily on application server-side certificate mapping, rather than the web server. To configure client certificate authentication, three settings needed to be changed. Using the WebSphere Administrative Console (**Security... SSL**), the SSL "configuration repertoire" used for the application server needed to have a checkbox marked for "Client Authentication." In addition, on the configuration page for the CSIV2 authentication protocol (**Security... Authentication Protocol... CSIV2 Inbound Authentication**), we selected "Required" for the setting Client Certificate Authentication. Lastly, we imported the client CA certificate into the server's store using the ikeyman utility.

To map client certificates to the LDAP identity store, we added a certificate map mode to the LDAP user registry configuration (**Security... User Registries... LDAP... Advanced LDAP Settings**). Two options are available: `EXACT_DN` and `CERTIFICATE_FILTER`. The former is used when the LDAP directory exactly mirrors the hierarchy of the certificate DN. Since this is highly unlikely in most enterprises, we selected the filter option:

## Advanced LDAP Settings

Advanced LDAP User Registry settings are used when users and groups reside in an external LDAP directory. When security is enabled and any of these properties are changed, please go to the GlobalSecurity panel and click Apply or OK to validate the changes. 

Configuration		
General Properties		
User Filter	<code>(&amp;(uid=%v)(objectclass=inetOrgPers)</code>	An LDAP filter clause for searching the registry for users.
Group Filter	<code>(&amp;(cn=%v)(!(objectclass=groupOfNames))</code>	An LDAP filter clause for searching the registry for groups.
User ID Map	<code>inetOrgPerson:uid</code>	An LDAP filter that maps the short name of a user to an LDAP entry.
Group ID Map	<code>*:cn</code>	An LDAP filter that maps the short name of a group to an LDAP entry.
Group Member ID Map	<code>groupOfNames:member;groupOfUniqueMember:member</code>	An LDAP filter that identifies User to Groups memberships.
Certificate Map Mode	<code>CERTIFICATE_FILTER</code> ▾	Whether to map X.509 Certificates into an LDAP directory by EXACT_DN or CERTIFICATE_FILTER. Specify CERTIFICATE_FILTER to use the specified Certificate Filter for the mapping.
Certificate Filter	<code>certificateDN=\${SubjectDN}</code>	If you specified the filter Certificate Mapping, use this property to specify the LDAP filter to use to map attributes in the client certificate to entries in LDAP.
<input type="button" value="Apply"/> <input type="button" value="OK"/> <input type="button" value="Reset"/> <input type="button" value="Cancel"/>		

For each user in the LDAP identity store, it is necessary to add an attribute containing the DN of the user's certificate, in this case `certificateDN`

See also:

WebSphere V5 Security Redbook (Chapter 10, section 10.10.3)

### Test Case 10.1

The `httpd.conf` file contains several directives that control how CRLs are obtained. The supported method is via an LDAP lookup.

```
SSLCRLHostname arj-hoover
SSLCRLPort 389
```

We could not get the IBM HTTPD server to successfully perform a revocation check on a client certificate. We were able to successfully configure the LDAP server to host the CRL and get HTTPD to find it, as evidenced by this debug log from iPlanet:

```
[16/Apr/2003:22:05:51 -0400] conn=60 fd=1068 slot=1068 connection from 172.120.32.95
to 172.120.32.95
```

```
[16/Apr/2003:22:05:51 -0400] conn=60 op=0 BIND dn="" method=128 version=2
[16/Apr/2003:22:05:51 -0400] conn=60 op=0 RESULT err=0 tag=97 nentries=0 etime=0 dn=""
[16/Apr/2003:22:05:51 -0400] conn=60 op=1 SRCH base="OU=Test Certificate
Division,O=Customer
Corp,CN=Test Certificate Authority" scope=0
filter="(|(certificaterevocationlist;binary=*)
(certificateRevocationList=*))" attrs="certificaterevocationlist;binary
certificateRevocationList"
[16/Apr/2003:22:05:51 -0400] conn=60 op=1 RESULT err=0 tag=101 nentries=1 etime=0
[16/Apr/2003:22:05:51 -0400] conn=60 op=2 UNBIND
[16/Apr/2003:22:05:51 -0400] conn=60 op=2 fd=1068 closed - U1
```

However, we could not get HTTDPD to do anything with the CRL we provided. With the server's LogLevel set to debug, here is the complete set of error messages available from IBM HTTDPD:

```
[Wed Apr 16 22:05:47 2003] [notice] Instance name is Apache 2696
[Wed Apr 16 22:05:47 2003] [info] BytesRead = 372 WSAProtocolInfo = 2006620
[Wed Apr 16 22:05:51 2003] [info] SSL0240I: Handshake Failed, Socket has been closed.
[Wed Apr 16 22:05:51 2003] [error] SSL0247E: Handshake Failed, LDAP server not
available.
```

Clearly, this is unacceptable. We searched on IBM's website and in the documentation, and we could not find any information whatsoever on what the errors signified or how to fix the problem.

## 4.4.2.5 External Authentication

---

### External Authentication

External authentication is weighted highest for web applications as they need to scale to larger numbers of users and may be hosted in environments where external authentication is a requirement. This is also true of web services. Intranet applications will typically make less use of external authentication. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 3
- Web Service: 3
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the External Authentication topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=80]. The application server should provide flexible support for external identity stores such as LDAP servers. Proprietary identity stores (e.g., Active Directory) may also be used, provided the access protocols are well-documented and supported by the industry.	IIS 6 offers four options for authenticating against an external store: Basic authentication, Integrated Windows authentication (utilizing either NTLM or Kerberos), Digest authentication for Windows domain servers, and .NET Passport authentication.	<b>Transparent (5)</b> . Integrated Windows authentication is enabled by default.	Basic authentication is not recommended even with the addition of SSL encryption since attacks such as cross-site tracing may still be able to recover a user's credentials.
[id=81]. If an identity store is used for authentication, binding and lookup operations should be performed using a secure protocol.	Secure protocols for authenticating to an external store are NTLM, Kerberos, and .NET Passport.	<b>Transparent (5)</b> . NTLM, Kerberos, and .NET Passport are all suitable choices.	

**Best Practice Compliance score**, based on mean of 2 best practices: 5

### Level of Effort Analysis

---

Test Case	Criteria	Rating
[id=80.1]. Configure the application server to use an external identity store for user and group lookup.	Implementation Complexity	<b>Wizard (5)</b> . Enabling any of the authentication mechanisms described above is easily accomplished via IIS' Authentication Methods configuration panel.
	Documentation and Examples	<b>Best practice (5)</b> . The various options are clearly explained along with security recommendations.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering IIS.
	Time to Implement	<b>Low (5)</b> . The default configuration is secure and can quickly be changed if an alternative authentication method is preferred.
[id=81.1]. Configure secure protocols for identity lookup.	Implementation Complexity	<b>Wizard (5)</b> . The default configuration is secure.
	Documentation and Examples	<b>Suitable (4)</b> . Documentation is clear but does not give in-depth information on the security features and impacts of the various protocols.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering IIS.
	Time to Implement	<b>Low (5)</b> . The default configuration is secure and can quickly be changed if an alternative authentication method is preferred.

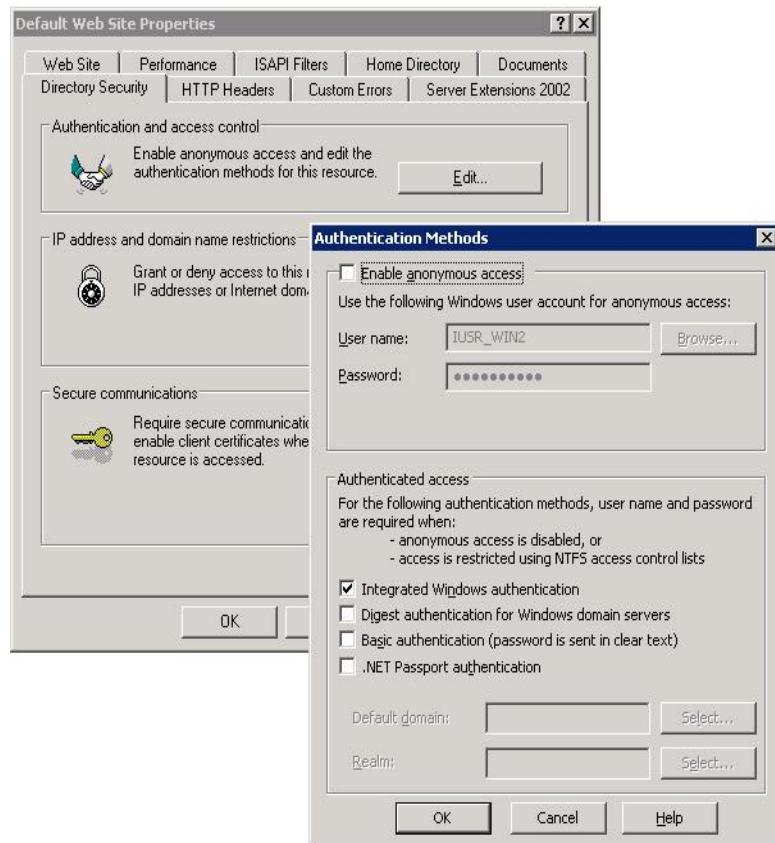
**Ease of Securing score**, based on mean of 2 test cases: 4.63

---

### Notes

#### Test Case 80.1

The screen shot below shows the IIS 6 Authentication Methods:



See also: ASP.NET Authentication and Authorization.

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=80]. The application server should provide flexible support for external identity stores such as LDAP servers. Proprietary identity stores (e.g., Active Directory) may also be used, provided the access protocols are well-documented and supported by the industry.	WebSphere offers two main options for authenticating against external identity stores: the application server's local OS and LDAP. It also provides a defined Java interface in case additional methods (e.g., external database) are desired. SSL can be used to secure the LDAP lookup operations.	<b>Wizard (4)</b> . Local OS support is built-in. External LDAP configuration is performed using a simple property sheet in the Administration Console. WebSphere includes LDAP "templates" for matching users and groups for 6 LDAP servers, including iPlanet/NetScape and Active Directory.	Users will need to carefully examine their LDAP configuration and make sure that the default mappings provided in WebSphere are appropriate for their situation. WebSphere should include a "live lookup" function in the Administration console to test LDAP configurations prior to committing changes.

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=81]. If an identity store is used for authentication, binding and lookup operations should be performed using a secure protocol.	WebSphere supports identity lookups using secure protocols; the container can be configured to use SSL-enabled LDAP. Re-use of connections is supported. When LDAP lookups are performed, the server must use a named identity and password for binding.	<b>Wizard (4)</b> . WebSphere application server provides the essential components system administrators will need to configure LDAP lookups over SSL. The fact that the server recycles SSL connections ensures that encrypted sessions will not burden the container's performance.	No recommendations for deployers; platform implements best practice. IBM should make the process of testing LDAP connections (on the server) more intuitive.

**Best Practice Compliance scores**, based on mean of 2 best practices: Linux 4, Unix 4

## Level of Effort Analysis

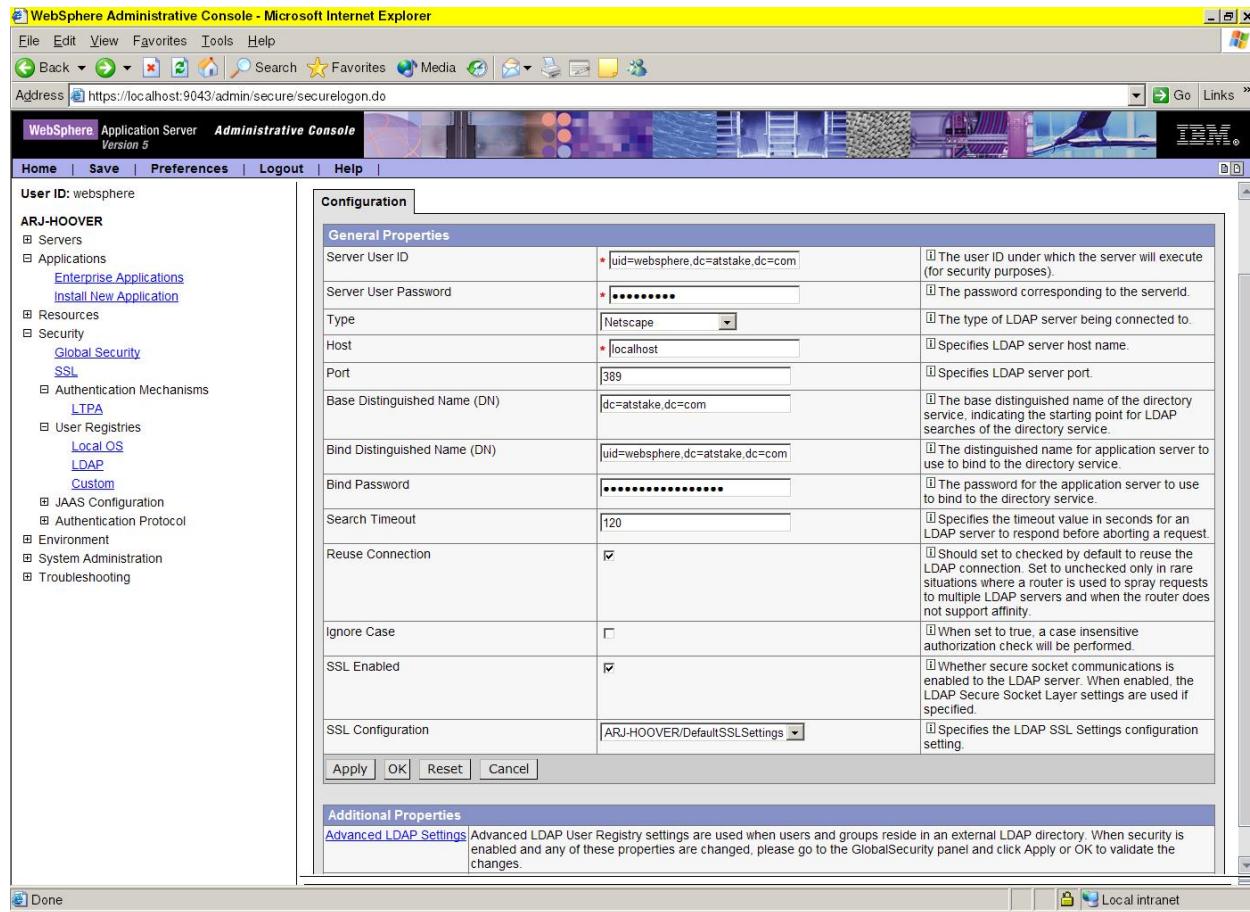
Test Case	Criteria	Rating
[id=80.1]. Configure the application server to use an external identity store for user and group lookup.	Implementation Complexity	<b>Wizard (5)</b> . A property sheet in the Administration Console configures the external identity store (see below). This should suit 90% of users, but if custom mappings are used, a "live lookup" tool would help verify that settings are correct. Most users will want to dive into the <i>Advanced Settings</i> page prior to committing changes.
	Documentation and Examples	<b>Suitable (4)</b> . Documentation provided many examples, in detail, for leading LDAP servers (a notable omission was OpenLDAP). In our tests with iPlanet LDAP, we found that settings for <i>Server User ID</i> and <i>Binding User ID</i> required full Distinguished Names, rather than a relative DN as stated in the documentation. IBM should provide more technical documentation on how lookups actually work, and how to troubleshoot them when they go wrong.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The implementer needs to know how LDAP servers work and what the target schema is. In some cases, familiarity with basic LDAP search syntax would be helpful, if customization of the user and group mappings are required. Knowledge of third-party LDAP browsers helped troubleshoot an implementation error.
	Time to Implement	<b>Medium (3)</b> .
[id=81.1]. Configure secure protocols for identity lookup.	Implementation Complexity	<b>Wizard+ (4)</b> . The application server administrator follows a three-step process for importing the LDAP SSL certificate and associating its use with LDAP lookups. All configuration is done through GUIs.
	Documentation and Examples	<b>Suitable (4)</b> . Documentation provides richly detailed examples on using SSL with several popular LDAP servers, including Active Directory. However, there is no documentation on what to do should something go wrong (for instance, if the SSL certificate is corrupt or lacks a CA chain).
	Implementor Competence	<b>Novice/intermediate (4)</b> . PKI operations (in general) demand knowledge of keys and key management; WebSphere is no different. Implementers should have previous background with PKI if troubleshooting is required.
	Time to Implement	<b>Medium (3)</b> . Process was tedious, but straightforward. Some time was spent troubleshooting some non-WebSphere-related SSL issues with the LDAP server.

**Ease of Securing scores**, based on mean of 2 test cases: Linux 3.88, Unix 3.88

## Notes

### Test Case 80.1

WebSphere uses the term "user registry" to refer to external identity stores. In this case, we used iPlanet Directory Server, v5.1. Setting up the connection is via a straightforward property sheet in the Administration console:



After configuring the LDAP user registry, users must change the container's default user registry to be LDAP. This is done via a combo box on the **Global Settings** page.

Users should take care to use the full DN for the *Binding User ID*. Not doing so can cause WebSphere to restart without being able to properly authenticate, which essentially forces the server to halt at startup. It also disables the Administration Console, leaving the operator with no obvious way to reverse the change! Fortunately, the (undocumented) file `%WEBSPHERE_HOME%/config/cells/server/security.xml` can be tweaked to get around this problem.

### Test Case 81.1

To enable LDAP lookups over SSL, system administrators follow a tedious, but reasonably straightforward, path. First, a "trust store" for storing the trusted server certificates and certificate roots is created. This is done with **ikeyman**, essentially a GUI wrapper around Sun's keytool. Keystores can be PKCS12, Sun's JKS format, or use native crypto storage libraries (we only tested JKS). Then the SSL server certificates for the LDAP server, and for the issuing CA, are imported. In this case, we used OpenSSL to generate a self-signed CA and SSL server certificate.

Second, after setting up the trust store, in the WebSphere Administration Console (**Security... SSL**) the server administrator creates a new *SSL Configuration Repertoire* that references the trust store. A property sheet style interface specifies the path to the new trust store, trust store passwords and the relevant certificate alias to use.

Finally, the LDAP configuration is modified through a form interface to use the new SSL configuration repertoire (**Security... User Registries... LDAP**).

As with the *External Authentication* test cases, we found it useful to back up the `websphere_root/config/cells/server_name/security.xml` file before making the LDAP configuration change. We had some initial problems with the SSL configuration on the LDAP server side. As it turned out, the problem had nothing to do with WebSphere — we had simply mis-configured the SSL server certificates — but we needed to roll back the WebSphere configuration to a "safe" state while the LDAP problem was solved.

## 4.4.2.6 Platform Integrated Authentication

---

### Platform Integrated Authentication

Platform integrated authentication is typically easier to implement than external authentication and often provides additional account management features. It is most likely used in intranet applications. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 1
- Web Service: 1
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Platform Integrated Authentication topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=7]. Passwords expire every 30-180 days. Password history prevents re-use of the previous 6-12 passwords		<b>Transparent (5)</b> . The platform integrated authentication has default behavior that meets best practices for password expiration and quality. The default value for password expiration is 42 days. The default value for number of passwords remembered is 24. Changing these values is accomplished through an easy to understand dialog box.	These defaults are appropriate for most applications.

**Best Practice Compliance score**, based on mean of 1 best practice: 5

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=7.1]. Implement password expiration policy.	Implementation Complexity	<b>Wizard (5)</b> . Modifications to the default settings were easily accomplished through a system dialog box.
	Documentation and Examples	<b>Best practice (5)</b> . Settings were self-explanatory.
	Implementor Competence	<b>Novice (5)</b> . No development skills required.
	Time to Implement	<b>Low (5)</b> . It took less than a minute to find the setting and modify it.
[id=7.2]. Implement password reuse policy.	Implementation Complexity	<b>Wizard (5)</b> . Modifications to the default settings were easily accomplished through a system dialog box.
	Documentation and Examples	<b>Best practice (5)</b> . Settings were self-explanatory.
	Implementor Competence	<b>Novice (5)</b> . No development skills required.
	Time to Implement	<b>Low (5)</b> . It took less than a minute to find the setting and modify it.

Ease of Securing score, based on mean of 2 test cases: 5

### Notes

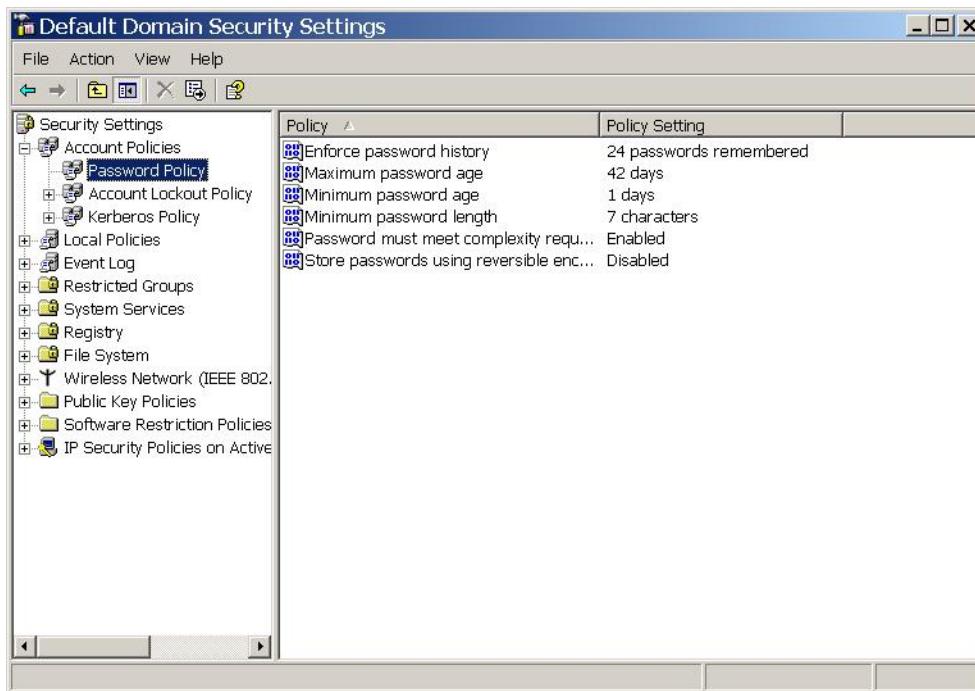
#### Test Case 7.1

Password expiration setting:



## Test Case 7.2

Password reuse setting:



## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
<i>Linux. [id=7]. Passwords expire every 30-180 days. Password history prevents re-use of the previous 6-12 passwords</i>	Red Hat Advanced Server supports sophisticated password checks through its PAM (Pluggable Authentication Module) interface but few of these options are available through a GUI.	<b>Wizard (4)</b> . Password expiration is easy to implement but password history requires a change in a PAM configuration file. Neither are configured by default.	
<i>Unix. [id=7]. Passwords expire every 30-180 days. Password history prevents re-use of the previous 6-12 passwords</i>	Unix supports password expiration but only appears to support password history when Kerberos authentication is used.	<b>Developer extends (3)</b> . Password expiration is simple to configure while password history requires substantially more effort. Neither is installed by default.	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 4, Unix 3

### Level of Effort Analysis

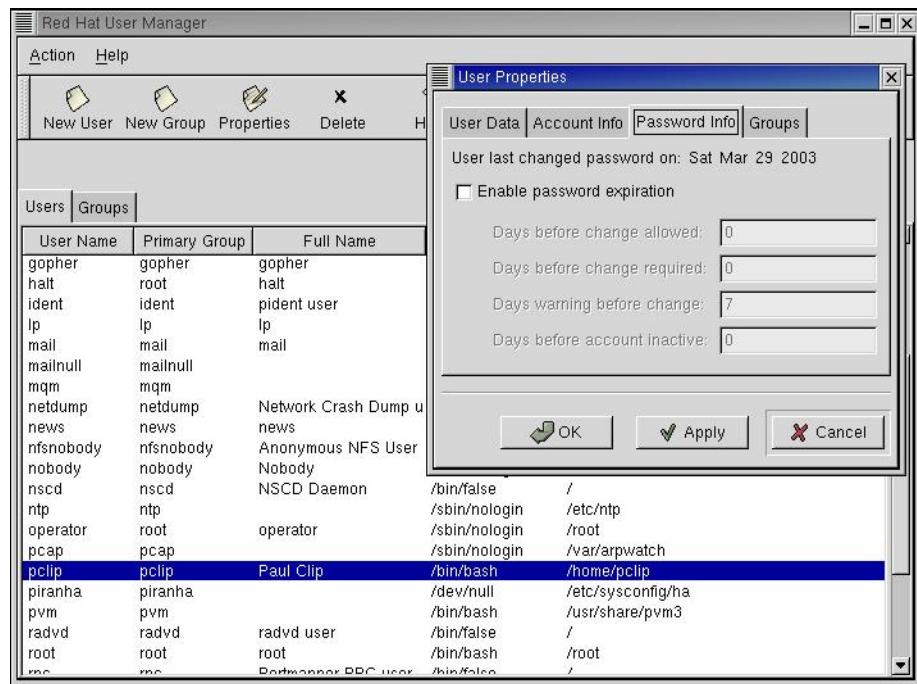
Test Case	Criteria	Rating
<i>Linux. [id=7.1]. Implement password expiration policy.</i>	Implementation Complexity	<b>Wizard (5)</b> . Forcing password expiration is achieved through the User Manager.
	Documentation and Examples	<b>Adequate (3)</b> . Red Hat documentation is functional and its instructions simple to apply.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering Red Hat Advanced Server.
	Time to Implement	<b>Low (5)</b> . The User Manager makes this a painless operation.
<i>Linux. [id=7.2]. Implement password reuse policy.</i>	Implementation Complexity	<b>Small amount of code (3)</b> . In order to implement password history, the user has to modify the PAM configuration files.
	Documentation and Examples	<b>Adequate (3)</b> . The PAM documentation is clear, if a little terse.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering Red Hat Advanced Server.
	Time to Implement	<b>Medium (3)</b> . Making the configuration change is simple, understanding which setting to change took some time.
<i>Unix. [id=7.1]. Implement password expiration policy.</i>	Implementation Complexity	<b>Wizard (5)</b> . Forcing password expiration is achieved through the Management Console.
	Documentation and Examples	<b>Adequate (3)</b> . Unix documentation is terse but functional given the Management Console's ease of use.
	Implementor Competence	<b>Novice (5)</b> . The developer has extensive experience developing web applications and is a novice in configuring/administering Unix.
	Time to Implement	<b>Low (5)</b> . The Management Console makes this a painless operation.
<i>Unix. [id=7.2]. Implement password reuse policy.</i>	Implementation Complexity	<b>Medium amount of code (2)</b> . The most common way Unix administrators implement password history is through a free software package called <code>npasswd</code> . Another mechanism is available by implementing the Unix vendor's Kerberos authentication.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . The Unix vendor's documentation mentions some options, such as implementing a custom PAM module, or deploying Kerberos, but it's light on details.
	Implementor Competence	<b>Novice (5)</b> . The developer has extensive experience developing web applications and is a novice in configuring/administering Unix.
	Time to Implement	<b>Medium to High (2)</b> . Significant time was spent trying to determine whether Unix could or couldn't support password history out of the box.

**Ease of Securing scores**, based on mean of 2 test cases: Linux 3.75, Unix 3.63

### Notes

#### Linux Test Case 7.1

The following screen shot highlights the User Manager's password expiration functionality:



See also:

[Linux: Modifying User Properties.](#)

### Linux Test Case 7.2

Implementing password history requires adding the `remember=N` parameter to the password `pam_unix.so` entry in `/etc/pam.d/system-auth`, where `N` is the number of passwords to remember.

See also: The change is documented in [/usr/local/doc/pam-0.75/html/pam-6.html](#), which is also available here.

### Unix Test Case 7.1

In addition to using the Management Console, password aging can also be enforced by editing the `/etc/default/passwd` file. The following is an excerpt of the `passwd` man file:

```

/etc/default/passwd
Default values can be set for the following flags in
/etc/default/passwd. For example: MAXWEEKS=26

MAXWEEKS
    Maximum time period that password is valid.

MINWEEKS
    Minimum time period before the password can be
    changed.

```

**PASSLENGTH**

Minimum length of password, in characters.

**WARNWEEKS**

Time period until warning of date of password's ensuing expiration.

See also:

Management Console, Setting password aging.

## 4.4.3 Communication Security

---

### Communication Security

Communication Security covers the different security options for an external source to communicate with an application securely. Options included message integrity, authentication, encryption and non-repudiation.

The Communication Security area of analysis includes this topic:

- Session Encryption

The table below describes the topic at a high level, and lists prevailing best practices. The right-hand columns show the weights @stake assigned for each of the scenarios.

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Session Encryption</b> . Session encryption is a critical requirement for applications where sensitive information is transmitted over an uncontrolled medium such as the Internet. Session encryption is rated high for web applications and web services and medium for intranet applications.	The application only accepts SSLv3 and TLSv1 encryption. Minimum key size is 128 bits. The server can be configured to use IPSEC between components. Management interfaces use secure methods to ensure that credentials and data are encrypted.	3	3	2

Scenario weights range from 1 (relatively unimportant) to 3 (extremely important). A scenario weight of zero means that the topic does not apply.

### Findings

The table below summarizes @stake's quantitative findings. For each analysis topic, the two high-level metrics (Best Practice Compliance and Ease of Securing) are shown. These are the arithmetic means of the underlying best practice and test case scores. The Ease of Securing metric is further broken down into four sub-metrics.

Topic	Platform	Best Practice Compliance	Ease of Securing				
			Overall	Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Session Encryption <i>3 best practices, 7 test cases</i>	.NET Windows	<b>3.67</b>	<b>3.89</b>	3.86	4	3.71	4
	WebSphere Linux	<b>3.33</b>	<b>3.82</b>	4.14	3.57	3.86	3.71
	WebSphere Unix	<b>4.33</b>	<b>4</b>	4.57	3.86	3.43	4.14

This area of analysis focused on services and protocols that provide secure remote access to web based applications. @stake developed seven (7) test cases to evaluate the efficacy and support for Secure Sockets Layer (SSL) and Transport Layer Security (TLS). The test cases focused primarily in the area of session level encryption.

The overall scores for both platforms were remarkably close, both vendors support secure communications channels to one degree or another. The disparity in scores was largely due to the lack of native support for IPSEC on the Linux platform, and the superiority of the Microsoft documentation and configuration examples.

The individual sections for each analysis topic describe @stake's findings in further detail.

### 4.4.3.1 Session Encryption

---

## Session Encryption

Session encryption is a critical requirement for applications where sensitive information is transmitted over an uncontrolled medium such as the Internet. Session encryption is rated high for web applications and web services and medium for intranet applications. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 3
- Web Service: 3
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Session Encryption topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

## Microsoft .NET

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=18]. The application only accepts SSLv3 and TLSv1 encryption. Minimum key size is 128 bits.	Support for 128bit encryption is built into the management console.	<b>Developer extends (3)</b> . Changing the settings requires manual edits in the registry and a restart of IIS.	Although the cryptographic API does provide the necessary flexibility to configure acceptable protocols and ciphers, IIS does not provide a user interface to easily configure them.
[id=19]. The server can be configured to use IPSEC between components.	Windows 2003 Server support for IPSEC is built into the operating system.	<b>Wizard (4)</b> . IPSEC configuration is straightforward.	The built-in OS support for IPSEC should be sufficient for most users.
[id=20]. Management interfaces use secure methods to ensure that credentials and data are encrypted.	Microsoft provides the .NET Framework 1.1 Configuration snap-in that allows you to configure assemblies, remoting services, and code access security policy specific to version 1.1 of the .NET Framework.	<b>Wizard (4)</b> . The .NET Framework 1.1 Configuration snap-in can only be run from the local machine running the .NET Framework.	Enabled Remote Desktop and add the appropriate user account to the Remote Desktop Users group. Use the Terminal Services Configuration snap-in to configure RDP-tcp connections to use the High encryption level. Connect to the server and run the .NET Framework 1.1 Configuration snap-in locally.

**Best Practice Compliance score**, based on mean of 3 best practices: 3.67

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=18.1]. Install server certificate in preparation for using SSL, TLS, and IPSEC.	Implementation Complexity	<b>Wizard+ (4)</b> . Process is driven by wizards. The most complex part is requesting a certificate from a CA.
	Documentation and Examples	<b>Suitable (4)</b> . Documentation adequate
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has wide-ranging Windows and web development experience with limited .NET exposure.
	Time to Implement	<b>Low to Medium (4)</b> . Except for the CA delay, this is a quick process
[id=18.2]. Configure system to use SSL.	Implementation Complexity	<b>Wizard+ (4)</b> . Simple wizard
	Documentation and Examples	<b>Best practice (5)</b> .
	Implementor Competence	<b>Novice/intermediate (4)</b> .
	Time to Implement	<b>Low (5)</b> .
[id=18.4]. Configure system to use TLS for web services.	Implementation Complexity	<b>Small amount of code (3)</b> .
	Documentation and Examples	<b>Adequate (3)</b> .
	Implementor Competence	<b>Novice/intermediate (4)</b> .
	Time to Implement	<b>Medium (3)</b> .
[id=18.5]. Ensure that the web server supports latest versions of SSL and TLS.	Implementation Complexity	<b>Wizard (5)</b> . This is the case out of the box.
	Documentation and Examples	<b>Suitable (4)</b> .
	Implementor Competence	<b>Novice/intermediate (4)</b> . This is not the kind of functionality an administrator would be exposed to on a daily basis, therefore expectations relative to pre-existing knowledge are low.
	Time to Implement	<b>Low (5)</b> . Already implemented.
[id=18.6]. Ensure that the web server supports minimum key size (128 bits).	Implementation Complexity	<b>Wizard (5)</b> . The web site security properties provide a checkbox to enforce a minimum of 128 bits.
	Documentation and Examples	<b>Adequate (3)</b> . Could provide more information on the consequences of checking the box
	Implementor Competence	<b>Novice/intermediate (4)</b> .
	Time to Implement	<b>Low (5)</b> .
[id=19.3]. Configure system to support IPSEC.	Implementation Complexity	<b>Medium amount of code (2)</b> . No single wizard is available
	Documentation and Examples	<b>Suitable (4)</b> . Adequate documentation is available
	Implementor Competence	<b>Novice/intermediate (4)</b> . Requires a good understanding of networking and cryptographic concepts
	Time to Implement	<b>Medium to High (2)</b> . Required some experimenting to get it working

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=20.1]. Ensure that management interfaces use secure transport mechanisms, and require authentication.	Implementation Complexity	<b>Wizard+ (4)</b> . Enabling and configuring Remote Desktop requires the administrator to use the System Properties sheet to enable Remote Desktop, the Terminal Services Configuration snap-in to enable High encryption, and Local Users and Groups snap-in to add the appropriate users.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation available locally in the Help and Support Center is clear and easy to follow for enabling Remote Desktop and using the .NET Framework 1.1 Configuration snap-in.
	Implementor Competence	<b>Expert/intermediate (2)</b> . The administrator has extensive experience with the Windows OS family. However, the components of the .NET Framework are new; therefore expectations relative to pre-existing knowledge are low.
	Time to Implement	<b>Low to Medium (4)</b> . It takes only a few minutes to enable and configure Remote Desktop. Once connected, the .NET Framework 1.1 Configuration snap-in can be launched from the start menu.

Ease of Securing score, based on mean of 7 test cases: 3.89

### Notes

#### Test Case 18.1

See "HOW TO: Enable SSL for All Customers Who Interact with Your Web Site in Internet Information Services" for step-by-step documentation on how to set SSL.

Documentation could have elaborated on PKI best practices and self signed certificates limitations.

#### Test Case 18.4

Usage of TLS is automatically turned on and will be used following SSL protocol negotiations. Forcing HTTPS over TLS only requires the same steps as the next text case.

#### Test Case 18.5

See Microsoft Knowledge Base Article - 245030: How to Restrict the Use of Certain Cryptographic Algorithms and Protocols in Schannel.dll for a step-by-step description on how to change the supported ciphers.

#### Test Case 18.6

In order to force 128 bit encryption, one has to check "Require secure channel". @stake failed to understand the relationship between these two checkboxes.

#### Test Case 19.3

This is a feature of the OS.

See Technet - IPsec for a detailed explanation of what IPsec is, how to deploy it, and its limitations.

See also: How To: Use IPSec to Provide Secure Communication Between Two Servers

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=18]. The application only accepts SSLv3 and TLSv1 encryption. Minimum key size is 128 bits.	SSL is a supported option for the IBM HTTP Server through a proprietary Apache module <code>ibm_ssl_module</code> .	<b>Wizard (4)</b> . Configuring the IBM HTTP Server for secure communications is a simple and relatively straightforward operation.	The IBM-supplied SSL package should be sufficient for most users.
<i>Linux</i> . [id=19]. The server can be configured to use IPSEC between components.	Red Hat Advanced Server (RHAS) does not support IPsec. If needed, the Red Hat documentation recommends using FreeS/Wan, a free IPsec implementation. Options installed with RHAS that replicate some of the IPsec functionality are OpenSSH and Crypto IP Encapsulation(CIPE).	<b>Developer implements (2)</b> . The FreeS/Wan package has to be installed in order to support IPsec on RHAS.	If IPsec support is required, FreeS/Wan is recommended.
<i>Unix</i> . [id=19]. The server can be configured to use IPSEC between components.	The Unix vendor's current distribution supports IPsec natively. The ipsecconf utility is used to manage IPsec, and the <code>/etc/inet/ipsecinit.conf</code> configuration file is the default IPsec policy configuration file.	<b>Transparent (5)</b> . IPsec is enabled by default on the Unix vendor's current distribution.	The Unix vendor-supplied IPsec package should be sufficient for most users.
[id=20]. Management interfaces use secure methods to ensure that credentials and data are encrypted.	The WebSphere Administrative Console by default creates an SSL-encrypted management site. Once the server is installed, authentication for administration must be configured.	<b>Wizard (4)</b> . The encrypted site is enabled by default, and the administrative configuration is straightforward.	Use the SSL-encrypted Administrative Console for WebSphere administration.

**Best Practice Compliance scores**, based on mean of 3 best practices: Linux 3.33, Unix 4.33

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=18.1]. Install server certificate in preparation for using SSL, TLS, and IPSEC.	Implementation Complexity	<b>Wizard (5)</b> . The management of SSL certificates is accomplished through the ikeyman program shipped with the IBM HTTP Server.
	Documentation and Examples	<b>Suitable (4)</b> . The documentation clearly steps the user through the creation of a key database and a self-signed certificate.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering IBM HTTP Server/Apache web servers.
	Time to Implement	<b>Low to Medium (4)</b> . The creation of a certificate is straightforward.

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=18.2]. Configure system to use SSL.	Implementation Complexity	<b>Wizard+ (4)</b> . Configuring an SSL server is fairly straightforward but requires following numerous steps.
	Documentation and Examples	<b>Suitable (4)</b> . The documentation clearly walks the user through the creation of an SSL server.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering IBM HTTP Server/Apache web servers.
	Time to Implement	<b>Medium (3)</b> . Implementation is simple but executing all the necessary steps takes time.
[id=18.4]. Configure system to use TLS for web services.	Implementation Complexity	<b>Wizard (5)</b> . SSL v2, SSL v3, and TLS v1 can be independently specified at the directory level via the Administrative Console. SSL v2 can also be disabled at a server level by only allowing SSL v3/TLS v1 ciphers.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . The documentation inadequately covers the configuration of SSL ciphers at the directory level, and explains which ciphers are associated with which protocols but does not provide step-by-step guidelines on how to specify ciphers in the Administrative Console.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering IBM HTTP Server/Apache web servers.
	Time to Implement	<b>Low to Medium (4)</b> . Once the required ciphers are selected, they are specified through the Host Authorization section of the IBM Administration Server.
[id=18.5]. Ensure that the web server supports latest versions of SSL and TLS.	Implementation Complexity	<b>Wizard (5)</b> . By default the IBM HTTP Server supports SSL v2, SSL v3, and TLS v1.
	Documentation and Examples	<b>Adequate (3)</b> . The documentation describes the protocols supported but does not cover security issues or shortcomings associated with a weak protocol such as SSL v2.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering IBM HTTP Server/Apache web servers.
	Time to Implement	<b>Low (5)</b> . No changes need be made.
[id=18.6]. Ensure that the web server supports minimum key size (128 bits).	Implementation Complexity	<b>Wizard (5)</b> . By default the IBM HTTP Server supports a number of 128 bit ciphers.
	Documentation and Examples	<b>Suitable (4)</b> . The documentation clearly lists the encryption strength of each cipher.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering IBM HTTP Server/Apache web servers.
	Time to Implement	<b>Low (5)</b> . No changes need be made.

### Level of Effort Analysis

Test Case	Criteria	Rating
<i>Linux. [id=19.3]. Configure system to support IPSEC.</i>	Implementation Complexity	<b>Large amount of code (1)</b> . Installing, configuring, and testing FreeS/Wan is a complex endeavor.
	Documentation and Examples	<b>Adequate (3)</b> . The Red Hat documentation explains in detail the IPSec alternatives it supports and refers users to FreeS/Wan in cases where IPSec is required.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering Red Hat Advanced Server.
	Time to Implement	<b>High (1)</b> . Implementing FreeS/Wan is a substantial undertaking.
<i>Unix. [id=19.3]. Configure system to support IPSEC.</i>	Implementation Complexity	<b>Wizard+ (4)</b> . IPSec can be administered via a single file, /etc/inet/ipsecinit.conf.
	Documentation and Examples	<b>Best practice (5)</b> . Online manual (man) and vendor web pages clearly document the configuration of IPSec on the Unix vendor's current distribution.
	Implementor Competence	<b>Expert (1)</b> . The implementer is very familiar with Linux, Unix operating systems, security, and IPSec configuration.
	Time to Implement	<b>Low to Medium (4)</b> . Implementation time will depend on the complexity of the configuration, but only a single file needs to be edited.
<i>[id=20.1]. Ensure that management interfaces use secure transport mechanisms, and require authentication.</i>	Implementation Complexity	<b>Wizard+ (4)</b> . The encrypted site is enabled by default; administrative configuration is straightforward.
	Documentation and Examples	<b>Best practice (5)</b> . The Administrative Console clearly steps the user through the process of implementing authentication for the Console.
	Implementor Competence	<b>Intermediate (3)</b> . The administrator is familiar with Unix operating systems and web administration but is unfamiliar with WebSphere administration.
	Time to Implement	<b>Low to Medium (4)</b> . The encrypted site was already enabled, and configuring authentication took only a few minutes. The WebSphere application server needs to be restarted to effect the changes.

**Ease of Securing scores**, based on mean of 7 test cases: Linux 3.82, Unix 4

### Notes

#### Test Case 18.4

Selecting SSL v2, SSL v3, and TLS v1 is done at the directory level:

The screenshot shows the 'IBM Administration Server' interface with the 'Directory Authorization' section selected in the left sidebar. The main panel displays the 'Scope' configuration for 'Directory'. It includes settings for enabling fake basic authentication (set to 'No'), selecting SSL/TLS protocols (SSLV2, SSLV3, TLSV1, All, Unset, with TLSV1 selected), and defining cipher specifications for permit and reject access. There are also fields for Boolean expressions and a summary of the current configuration.

**IBM Administration Server**

**Directory Authorization**

IBM HTTP Server

Ready

**Scope:** <Directory >

Enable fake basic authentication:  Yes  No

Set SSL protocol version:  SSLV2  SSLV3  TLSV1  All  Unset

Cipher specifications used to permit access:

Cipher specifications used to reject access:

Boolean expressions to enable extended validation of client certificate information:

**Submit** **Reset**

Specifying strong ciphers (128 bits or more) supporting only SSL v3 or TLS v1 can be performed at a server level via Host Authorization:

The screenshot shows the 'Host Authorization' configuration page for an IBM HTTP Server. The left sidebar contains a navigation tree with various server settings like Basic Settings, Configuration Structure, Security, and Performance. The 'Host Authorization' option under Security is selected. The main panel displays the 'Host Authorization' configuration with the following details:

- Scope:** <VirtualHost 192.168.106.100:443 (localhost.localdomain)>
- Enable SSL:** Yes (radio button selected)
- Mode of client authentication to use:** None (radio button selected)
- Server certificate to use for this virtual host:** (empty input field)
- Cipher specifications that you can use in a secure transaction:**
  - 34 : RC4 MD5 (128 bit) (selected)
  - 35 : RC4 SHA (128 bit)
  - 3A : Triple-DES SHA (168 bit)
- Client certificate groups:** (empty input field)

At the bottom of the panel are 'Add', 'Delete', 'Up', and 'Down' buttons for managing cipher specifications and client certificate groups. Below the main panel are 'Submit' and 'Reset' buttons.

### Test Case 20.1

By default, connections to the Administrative Console do not require authentication. An appropriate User Registry must be chosen, and Global Security must be enabled to enable authentication for the Administrative Console.

## 4.4.4 Information Disclosure

---

### Information Disclosure

Information disclosure is the unauthorized presentation of sensitive or confidential information. Proper application design should safeguard against information disclosure to protect both application end users and intellectual property kept by the application. Attackers often utilize information disclosure vulnerabilities to stage more serious attacks.

The Information Disclosure area of analysis includes these topics:

- Error Messages and Exception Handling
- Logging
- URL Content Protection

The table below describes the topics at a high level, and lists prevailing best practices. The right-hand columns show the weights @stake assigned for each of the scenarios.

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Error Messages and Exception Handling</b> . Proper error message handling is an important feature of all web applications. It is more important in scenarios that are externally accessible than internal applications. Web application and web service scenarios are weighted medium and intranet application are weighted low.	The platform can be configured to return concise error message to the user and log more specific details securely.	2	2	1
<b>Logging</b> . Secure logging is an important feature of all applications. All scenarios are weighted medium.	The platform supports industry standard log file formats to ensure compatibility with analysis tools and host based intrusion detection. Log file size and message verbosity can be managed in a granular fashion. Logs are written by a process with write permission only. Logs are read by users with administrative access. Logging occurs on a separate, protected server to ensure that logs cannot be tampered with by attackers.	2	2	2
<b>URL Content Protection</b> . Content protection is an important feature of all web applications. It is more important in scenarios that are externally accessible than internal applications. Web application and web service scenarios are weighted medium and intranet application are weighted low.	The platform restricts directory listings upon receipt of a URL with a terminating slash, or with a URL containing a context or sub context that does not otherwise map to a named document or welcome page. The platform granularly restricts objects under the webroot. The platform supports restricted file systems and webroots.	2	2	1

Scenario weights range from 1 (relatively unimportant) to 3 (extremely important). A scenario weight of zero

means that the topic does not apply.

## Findings

The table below summarizes @stake's quantitative findings. For each analysis topic, the two high-level metrics (Best Practice Compliance and Ease of Securing) are shown. These are the arithmetic means of the underlying best practice and test case scores. The Ease of Securing metric is further broken down into four sub-metrics.

Topic	Platform	Best Practice Compliance	Ease of Securing				
			Overall	Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Error Messages and Exception Handling <i>1 best practice, 1 test case</i>	.NET Windows	<b>5</b>	<b>4.5</b>	5	4	4	5
	WebSphere Linux	<b>4</b>	<b>3.75</b>	5	3	3	4
	WebSphere Unix	<b>4</b>	<b>3.75</b>	5	3	3	4
Logging <i>4 best practices, 4 test cases</i>	.NET Windows	<b>4.25</b>	<b>4.44</b>	4.75	4.5	4	4.5
	WebSphere Linux	<b>4</b>	<b>3.88</b>	4.5	4	2.25	4.75
	WebSphere Unix	<b>4</b>	<b>3.88</b>	4.5	4	2.25	4.75
URL Content Protection <i>3 best practices, 5 test cases</i>	.NET Windows	<b>3.67</b>	<b>4.6</b>	4.6	4.6	4.6	4.6
	WebSphere Linux	<b>3.67</b>	<b>3.65</b>	4	3.4	3.2	4
	WebSphere Unix	<b>3.67</b>	<b>3.65</b>	4	3.4	3.2	4

This area of analysis covered two (2) test cases pertaining to the platform's ability to limit unauthorized presentation of sensitive or confidential information. The topics under evaluation included Error Handling and Stack Traces and Debugging. Under both test cases, Microsoft scored slightly higher. The default configuration of the .NET Framework limits information disclosure.

The individual sections for each analysis topic describe @stake's findings in further detail.

#### 4.4.4.1 Error Messages and Exception Handling

---

### Error Messages and Exception Handling

Proper error message handling is an important feature of all web applications. It is more important in scenarios that are externally accessible than internal applications. Web application and web service scenarios are weighted medium and intranet application are weighted low. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 1

The sections that follow contain the results of @stake's analysis of the Error Messages and Exception Handling topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

---

Best Practice	Options	Rating	Recommendation
[id=29]. The platform can be configured to return concise error message to the user and log more specific details securely.	By default the IIS 6 errors (e.g. 404, 403, etc.) are concise and reveal no sensitive information. Each HTTP error code can be set to display a default message, an HTML page to return, or a URL to redirect to.	<b>Transparent (5)</b> . The error messages address information disclosure concerns.	

**Best Practice Compliance score**, based on mean of 1 best practice: 5

---

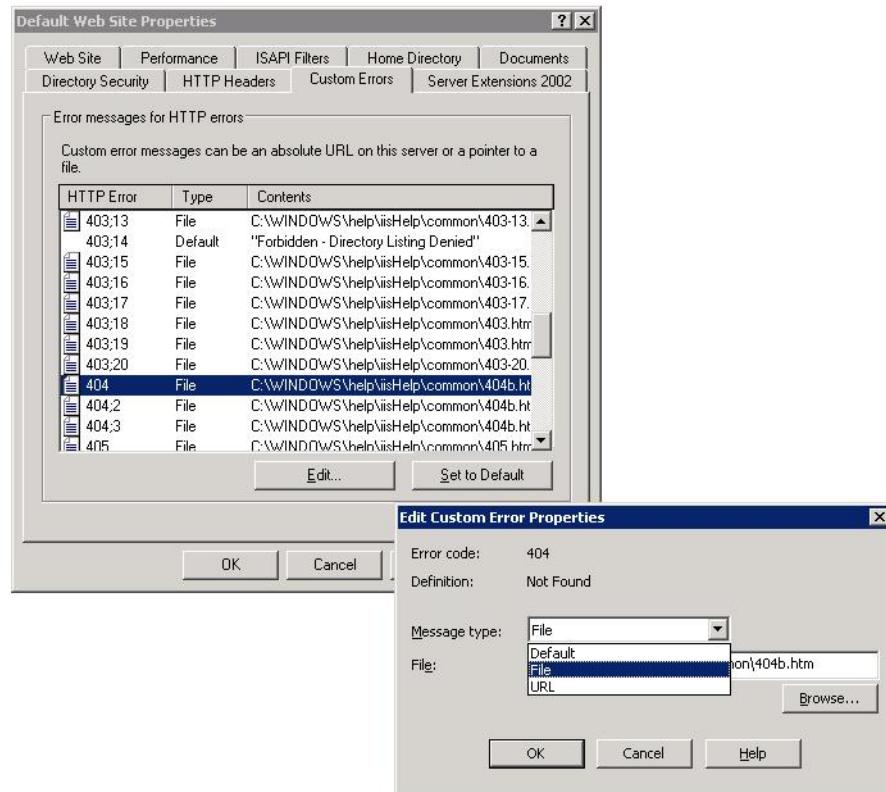
### Level of Effort Analysis

Test Case	Criteria	Rating
[id=29.1]. Ensure that the platform default error handling and reporting mechanisms may be limited to prevent information disclosure.	Implementation Complexity	<b>Wizard (5)</b> . Changing error pages is easy to accomplish via the Custom Errors tab in the IIS 6 properties panel.
	Documentation and Examples	<b>Suitable (4)</b> . The configuration panel is intuitive enough that no documentation needs to be read to use it. The documentation provided with IIS 6 is clear but does not mention any information disclosure risks.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web application and limited skills in configuring/administering IIS web servers.
	Time to Implement	<b>Low (5)</b> . The default configuration is secure, no changes need to be made.
<b>Ease of Securing score</b> , based on mean of 1 test case: 4.5		

### Notes

#### Test Case 29.1

IIS 6 custom errors:



## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=29]. The platform can be configured to return concise error message to the user and log more specific details securely.	By default, the IBM HTTP Server returns terse HTTP Status codes that do not reveal sensitive information. Each HTTP error code can be set to display a default message, an HTML page, or a URL to redirect to.	<b>Wizard (4)</b> .	

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 4, Unix 4

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=29.1]. Ensure that the platform default error handling and reporting mechanisms may be limited to prevent information disclosure.	Implementation Complexity	<b>Wizard (5)</b> . Assigning specific HTTP status-codes to customized error-page output was easily accomplished using a property-sheet in the IBM Administration Server.
	Documentation and Examples	<b>Adequate (3)</b> . IBM Redbook documentation was accurate, though moderately terse. It was found to be adequate for this test-case
	Implementor Competence	<b>Intermediate (3)</b> . The tester has wide experience in configuring and deploying web servers.
	Time to Implement	<b>Low to Medium (4)</b> . The modifications to the Web Server configuration file was accomplished in short order

**Ease of Securing scores**, based on mean of 1 test case: Linux 3.75, Unix 3.75

### Notes

#### Test Case 29.1

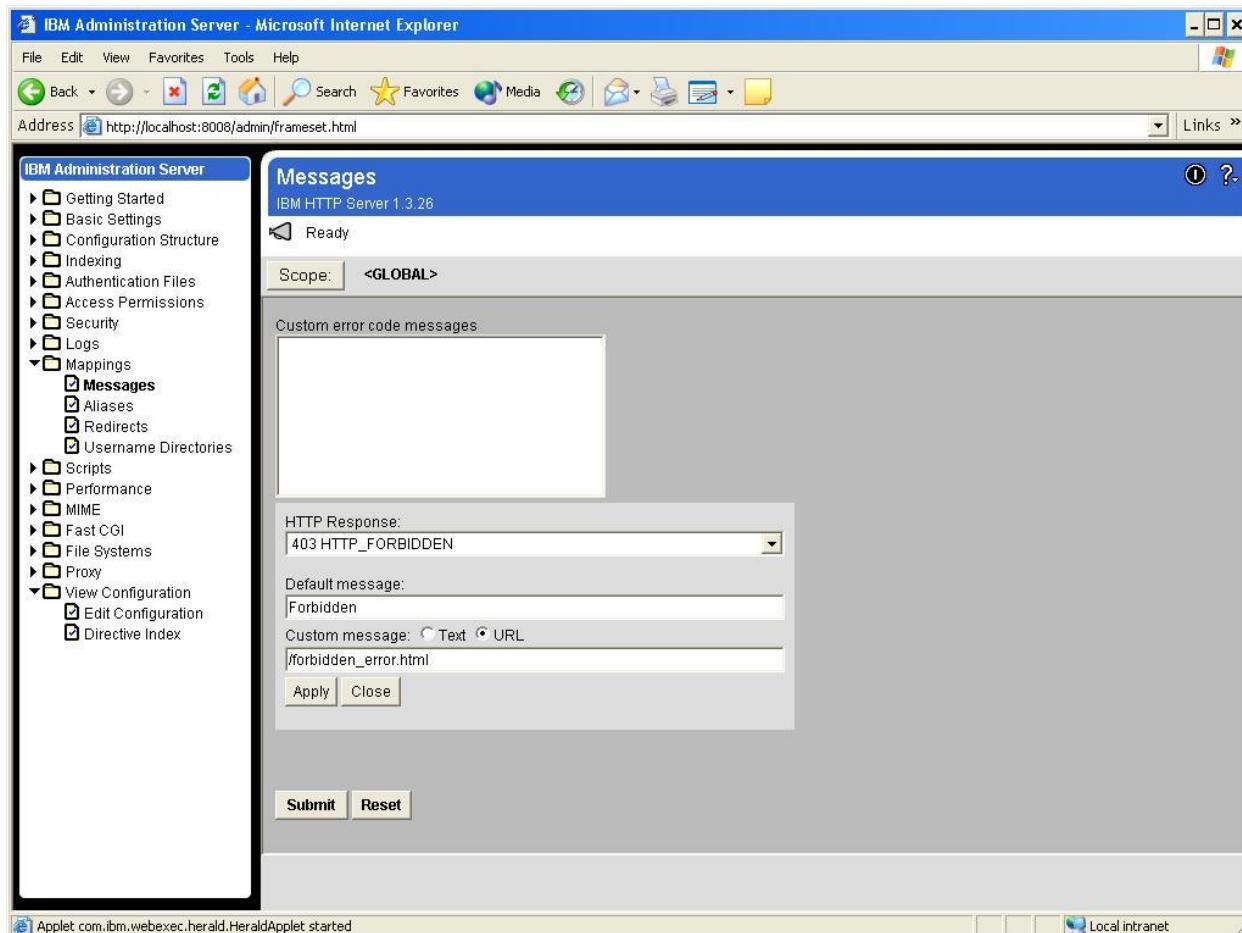
The IBM HTTP Server supports custom error messages in its core. The server can be configured to take a number of actions on an error condition, e.g.:

- Return the default message
- Return a customized message
- Redirect the request to another local or remote URL (including CGI programs)

The first option is the default, and takes place if no other option is specified. The final option, may be used to construct sophisticated logging and reporting schemes. Though not implemented as part of this test-case, it may be assumed that URL redirection could be used to *log more specific details securely* in keeping with the spirit of the best practice.

An example of modifying a single ErrorDocument directive by means of a property sheet wizard is

illustrated below:



## Documentation Sources

The Apache web site provided documentation on the httpd.conf `ErrorDocument` directive

The IBM Web Server - Powered by Apache Redbook provided a step-by-step procedure for implementing custom HTTP status-code handling

#### 4.4.4.2 Logging

---

## Logging

Secure logging is an important feature of all applications. All scenarios are weighted medium. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Logging topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

## Microsoft .NET

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=30]. The platform supports industry standard log file formats to ensure compatibility with analysis tools and host based intrusion detection.	IIS 6 supports multiple log file formats (configurable per website): W3C Extended Log File Format, ODBC Logging, NCSA Common Log File Format, and Microsoft IIS Log File Format. An additional format, centralized binary logging, is available at a web server level which performs consolidated logging for all web sites running on the same IIS 6 server.	<b>Transparent (5)</b> . The W3C log file format is the IIS 6 default format.	
[id=31]. Log file size and message verbosity can be managed in a granular fashion.	IIS 6 log files have multiple options in terms of file rotation and the information being logged.	<b>Transparent (5)</b> . Logging options are very easy to specify via the IIS 6 properties configuration panel.	
[id=32]. Logs are written by a process with write permission only. Logs are read by users with administrative access.	By default, members of the Administrators and SYSTEM groups are given full control over the log directory. No other groups or users have rights on this directory.	<b>Wizard (4)</b> . Configuring write permissions on the log directory and its contents requires using the file explorer security panel.	The default log file permissions are appropriate. If the information being logged is so sensitive that it requires write-only access, that information should probably be stored in a more secure location altogether.

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=33]. Logging occurs on a separate, protected server to ensure that logs cannot be tampered with by attackers.	While IIS 6 does not support syslog-like UDP logging, two options are available. The first is logging to an ODBC data source, the second is mapping a drive to a remote server and logging to that drive.	<b>Developer extends (3)</b> . Remote logging is possible but not optimal in terms of performance.	Logging to an ODBC data source will likely be too slow for sites with significant traffic. Logging to a remote drive is more efficient but still suffers from some drawbacks. Creating a drive mapping from the web to the log server increases the application's attack surface and requires additional administration work in securing and maintaining this connection. Performance will still suffer compared to a lighter weight solution (e.g. UDP based) as the SMB protocol imposes its own overhead.

**Best Practice Compliance score**, based on mean of 4 best practices: 4.25

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=30.1]. Ensure that the platform supports W3C message format.	Implementation Complexity	<b>Wizard (5)</b> . W3C is the default format.
	Documentation and Examples	<b>Suitable (4)</b> . No documentation was required since the log settings are in a very intuitive location: the web server properties panel. The documentation provided explains the W3C format and links to w3.org for more information.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web application and limited skills in configuring/administering Windows servers.
	Time to Implement	<b>Low (5)</b> . No changes need to be made.
[id=31.1]. Exercise the exposed log file management functionality of the platforms..	Implementation Complexity	<b>Wizard (5)</b> . Changes are simple to make.
	Documentation and Examples	<b>Best practice (5)</b> . The configuration panel is intuitive enough that no documentation needs to be read. The documentation provided with IIS 6 is clear, comprehensive, and discusses the security impacts associated some of the logging options.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web application and limited skills in configuring/administering IIS web servers.
	Time to Implement	<b>Low (5)</b> . The configuration time involved is trivial.

### Level of Effort Analysis

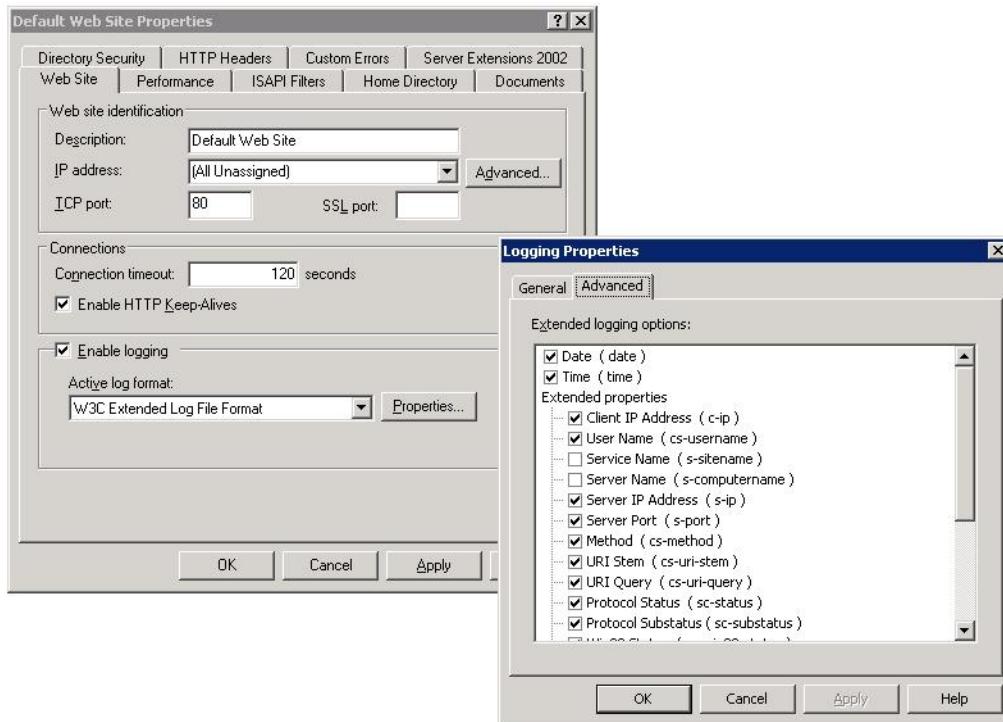
Test Case	Criteria	Rating
[id=32.1]. Configure log file permissions for write-only access by runtime processes.	Implementation Complexity	<b>Wizard (5)</b> . Changes are easy to make via the log file directory security properties.
	Documentation and Examples	<b>Best practice (5)</b> . The security panel is intuitive enough that no documentation needs to be read. The documentation provided with IIS 6 is clear, comprehensive, and discusses the security impacts associated with permissions on the log file directory.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web application and limited skills in configuring/administering IIS web servers.
	Time to Implement	<b>Low (5)</b> . The configuration time involved is trivial.
[id=33.1]. Configure the platform components to support centralized logging to a remote host.	Implementation Complexity	<b>Wizard+ (4)</b> . Using ODBC logging is straightforward. Logging to a remote server requires creating and securing the drive mapping.
	Documentation and Examples	<b>Suitable (4)</b> . ODBC logging is well explained in IIS 6's documentation. Map shares is covered in the Windows documentation.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web application and limited skills in configuring/administering IIS web servers.
	Time to Implement	<b>Medium (3)</b> . In either case the implementation is straightforward and fairly quick. This level of effort estimate does not include the time needed to create a data repository for ODBC logging.

**Ease of Securing score**, based on mean of 4 test cases: 4.44

### Notes

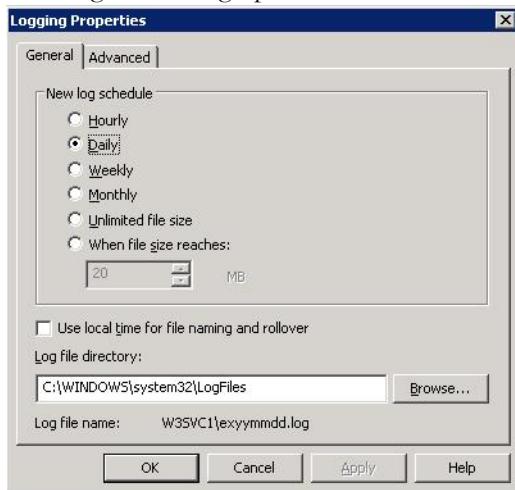
#### Test Case 30.1

IIS 6's log files are easy to customize:



### Test Case 31.1

IIS 6 log scheduling options:



## IBM WebSphere J2EE

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=30]. The platform supports industry standard log file formats to ensure compatibility with analysis tools and host based intrusion detection.	The IBM HTTP Web Server supports W3C standard log file formats in its core. The product provided for extensive modification and customization of log messages and log file manipulation.	<b>Transparent (5)</b> . No specific actions or modifications must be taken to ensure that standards-based log file formats are employed	
[id=31]. Log file size and message verbosity can be managed in a granular fashion.	The IBM HTTP Server comes with a set of simple log-file rotation utilities, documented examples and configurations. Furthermore, log message verbosity can be controlled via simple modifications to the <code>httpd.conf</code> file.	<b>Developer extends (3)</b> . Simple log rotation may be accomplished by use of the tools and configurations supplied with the base IBM HTTP Server	
[id=32]. Logs are written by a process with write permission only. Logs are read by users with administrative access.	By default, IBM HTTP Server logs are owned by root, readable by all (owner/group/other), but only writable by root.	<b>Wizard (4)</b> . Making the log files writable-only requires executing the <code>chmod</code> command.	For most users' needs, the default log settings are secure. For increased security, consider logging to a remote host.
[id=33]. Logging occurs on a separate, protected server to ensure that logs cannot be tampered with by attackers.	The IBM HTTP Server directly supports remote logging via the <code>syslog</code> facility of the ErrorLog Directive .	<b>Wizard (4)</b> . The IBM HTTP Server provides well known, simple to implement, remote logging capabilities.	
<b>Best Practice Compliance scores</b> , based on mean of 4 best practices: Linux 4, Unix 4			

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=30.1]. Ensure that the platform supports W3C message format.	Implementation Complexity	<b>Wizard (5)</b> . The IBM HTTP Server provides this functionality by default
	Documentation and Examples	<b>Suitable (4)</b> . The documentation is clear and concise, providing both example configurations and security commentary
	Implementor Competence	<b>Intermediate (3)</b> . The tester has wide experience in configuring and deploying web servers.
	Time to Implement	<b>Low (5)</b> . No specific modifications were required to bring the server into conformance with this best practice
[id=31.1]. Exercise the exposed log file management functionality of the platforms..	Implementation Complexity	<b>Wizard+ (4)</b> . Moderately sophisticated log rotation schemes can be created with a minimum of effort.
	Documentation and Examples	<b>Suitable (4)</b> . The Apache supplied documentation was found to be concise and accurate. It clearly describes how to implement log rotation functionality and specifically called out potential security implications in detail.
	Implementor Competence	<b>Expert/Intermediate (2)</b> . The tester has wide experience in configuring and deploying web servers.
	Time to Implement	<b>Low to Medium (4)</b> . Implementing a log rotation scheme was accomplished by using a flexible property sheet wizard supplied with that IBM HTTP Server Administrative Console.

### Level of Effort Analysis

---

Test Case	Criteria	Rating
[id=32.1]. Configure log file permissions for write-only access by runtime processes.	Implementation Complexity	<b>Wizard (5)</b> . Changes are trivial to make via well known <i>Unix</i> file manipulation commands.
	Documentation and Examples	<b>Adequate (3)</b> . IBM supplied documentation on this specific best practice was somewhat scant, though adequate to the task. However, the Apache documentation was more thorough and useful
	Implementor Competence	<b>Expert/intermediate (2)</b> . The tester has wide experience in configuring and deploying web servers.
	Time to Implement	<b>Low (5)</b> . The configuration time involved was trivial.
[id=33.1]. Configure the platform components to support centralized logging to a remote host.	Implementation Complexity	<b>Wizard+ (4)</b> . Modifications have to be made in two places. The first modification can be made using a property sheet wizard invoked from the IBM Administration Server. The second, if the default behavior is not satisfactory, must be made using a text editor to the /etc/syslog.conf file.
	Documentation and Examples	<b>Best practice (5)</b> . The Apache documentation was clear and concise, specifically calling-out potential security pitfalls and providing best practice examples to follow.
	Implementor Competence	<b>Expert/intermediate (2)</b> . The tester has wide experience in configuring and deploying web servers.
	Time to Implement	<b>Low (5)</b> . The configuration time involved was trivial.

**Ease of Securing scores**, based on mean of 4 test cases: Linux 3.88, Unix 3.88

---

### Notes

#### Test Case 30.1

An example of the IBM HTTP Server log file format configuration property sheet:

The screenshot shows the 'Other Logs' configuration page in the IBM Administration Server. The left sidebar contains a tree view of server settings. The main panel is titled 'Define additional logs' for a 'Custom' log type. It includes fields for 'Log file name' (set to 'logs/access.log'), 'Log format name' (set to 'common'), and 'Format string' (set to '%h %l %u %t "%r" %>s %b'). There is also an optional section for environment variables. At the bottom are 'Submit' and 'Reset' buttons.

## Documentation Sources

The IBM web site provided documentation describing the various log files and provided descriptions of the W3C standard formats in use.

### Test Case 31.1

Apache httpd is capable of writing error and access log files through a pipe to another process, rather than directly to a file. This capability dramatically increases the flexibility of logging, without adding code to the main server.

In order to write logs to a pipe the filename in the property sheet us replaced with the pipe character "|", followed by the name of the executable which should accept log entries on its standard input.

The IBM documentation specifically calls-out the security implications of using a separate process to manage log file rotation.

The most important use of piped logs is to allow log rotation without requiring a server restart. The IBM HTTP Server includes a simple program called `rotatelogs` for this purpose.

For example, to rotate the logs every 24 hours, the `CustomLog` directive would be set in this manner:

```
CustomLog "| /usr/local/apache/bin/rotatelogs/var/log/access_log 86400" common
```

The IBM web site provided documentation describing the various log file formats and provided descriptions of the W3C standard formats in use.

The Apache web site provided documentation describing the various `LogLevel` directives used to adjust the verbosity of the messages recorded in the logs.

The following *levels* are available, in order of decreasing significance:

Level	Description	Example
emerg	Emergencies - system is unusable.	"Child cannot open lock file. Exiting"
alert	Action must be taken immediately.	"getpwnam: couldn't determine user name from uid"
crit	Critical Conditions.	"socket: Failed to get a socket, exiting child"
error	Error conditions.	"Premature end of script headers"
warn	Warning conditions.	"child process 1234 did not exit, sending another SIGHUP"
notice	Normal but significant condition.	"httpd: caught SIGBUS, attempting to dump core in ..."
info	Informational.	"Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)..."
debug	Debug-level messages	"Opening config file ..."

### Test Case 32.1

An illustration of the default log-file permission settings on the IBM HTTP Server:

```

RPTlinux - SecureCRT
File Edit View Options Transfer Script Tools Window Help
File Edit View Options Transfer Script Tools Window Help
Last login: Mon Apr  7 14:28:53 2003 from rtp-135.atstake.com
[fheidt@linux3 fheidt]$ pwd
/home/fheidt
[fheidt@linux3 fheidt]$ cd /
[fheidt@linux3 /]$ ls
bin  dev  home  lib  misc  opt  root  socket.17526  usr
boot  etc  initrd  lost+found  mnt  proc  shm  tmp  var
[fheidt@linux3 /]$ cd /opt
[fheidt@linux3 opt]$ ls
IBMHttpServer  IBMJava2-131  lost+found  man  WebSphere  wamps
[fheidt@linux3 opt]$ cd IBMHttpServer/
[fheidt@linux3 IBMHttpServer]$ ls
admindocs  conf  icons  license  man  tivready
bin  example_module  include  logs  readme  _uninst
cgi-bin  htdocs  libexec  log.txt  ssl  version.signature
[fheidt@linux3 IBMHttpServer]$ cd logs
[fheidt@linux3 logs]$ ls -la
total 128
drwxr-xr-x  2 root      root          4096 Mar 27 12:58 .
drwxr-xr-x  18 root      root          4096 Mar 19 09:08 ..
-rw-r--r--  1 root      root        30240 Mar 27 23:54 access_log
-rw-r--r--  1 root      root       54590 Mar 27 23:51 admin_access.log
-rw-r--r--  1 root      root       10950 Mar 27 23:51 admin_error.log
-rw-r--r--  1 root      root          6 Mar 27 12:58 admin.pid
-rw-r--r--  1 root      root       6961 Mar 27 23:53 error_log
-rw-r--r--  1 root      root          5 Mar 27 23:53 httpd.pid
[fheidt@linux3 logs]$ 
Ready
ssh2: AES-128 33, 23 33 Rows, 92 Cols VT100

```

## Documentation Sources

The IBM web site provided Redbook documentation describing log file security. The Apache group has documentation regarding Permissions on ServerRoot Directories and Log Files. The security implications of protecting log files and directories figures prominently on this page

#### 4.4.4.3 URL Content Protection

---

### URL Content Protection

Content protection is an important feature of all web applications. It is more important in scenarios that are externally accessible than internal applications. Web application and web service scenarios are weighted medium and intranet application are weighted low. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 1

The sections that follow contain the results of @stake's analysis of the URL Content Protection topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=25]. The platform restricts directory listings upon receipt of a URL with a terminating slash, or with a URL containing a context or sub context that does not otherwise map to a named document or welcome page.		<b>Transparent (5).</b>	This is transparently done by the web server.

---

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=27]. The platform granularly restricts objects under the webroot.	Access to URLs can be limited to a particular role in two ways. The first requires storing users and groups in the Windows' authentication store, either locally or in Active Directory. Permissions are set using NTFS file system ACLs. The limitation is that Windows Integrated Authentication, Digest Authentication, Basic Authentication, or Passport must be used. Forms based authentication also has built-in URL access restrictions. This method does not rely on the Windows SAM but required editing a configuration file, web.config.	<b>Transparent (5)</b> . The Windows platform supports URL restrictions well. It was easy to configure both types of URL restrictions either using Windows users and groups with NTFS permissions or with Forms based authentication.	If your application is able to use Windows authentication store to store your users and groups it is easier to use NTFS permissions.
[id=28]. The platform supports restricted file systems and webroots.	There is no read only file system or chroot on the windows platform. NTFS permissions and creating a document root on a separate partition are the best way to restrict access to the document root.	<b>Not possible (1)</b> . Windows has no concept of a restricted file system such as a read only file system or a chrooted file system under Unix. NTFS does have features which can accomplish some measure of protection. The default permission settings for the web root do protect the content from being written by the web server process and normal users. Directory traversal can be hard stopped by placing the web root in its own separate partition.	The default permissions of the web root are adequate. Install the web root on its own partition if possible.

**Best Practice Compliance score**, based on mean of 3 best practices: 3.67

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=25.1]. Ensure that the default behavior for a given directory is to deny directory listings.	Implementation Complexity	<b>Wizard (5)</b> .
	Documentation and Examples	<b>Best practice (5)</b> .
	Implementor Competence	<b>Novice/intermediate (4)</b> .
	Time to Implement	<b>Low (5)</b> .
[id=25.2]. Configure the platform to restrict directory listing.	Implementation Complexity	<b>Wizard (5)</b> .
	Documentation and Examples	<b>Best practice (5)</b> .
	Implementor Competence	<b>Novice (5)</b> .
	Time to Implement	<b>Low (5)</b> .

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=27.1]. Restrict access to a given role to a file, folder, or object.	Implementation Complexity	<b>Wizard (5)</b> . All URL restriction configuration was done with a simple permissions dialog box.
	Documentation and Examples	<b>Best practice (5)</b> . The help system was well written and easy to understand.
	Implementor Competence	<b>Novice (5)</b> . No developer skills required.
	Time to Implement	<b>Low (5)</b> . It took only a few minutes to restrict access to an object.
[id=28.1]. Ensure that webroots are deployed or recommended to be placed in restricted file system locations by default.	Implementation Complexity	<b>Small amount of code (3)</b> . Configuration is done by dialog but is limited and several different steps must be done.
	Documentation and Examples	<b>Adequate (3)</b> . There is some documentation on locking down web sites on the Microsoft web site that describes how to set permissions and use a separate partition.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . No developer skills required but good OS administration skills required.
	Time to Implement	<b>Medium (3)</b> . It took about 30 minutes to configure permissions and create the separate partition.
[id=28.2]. Ensure that the platform prevents sensitive web service files from being served.	Implementation Complexity	<b>Wizard (5)</b> . The default behavior was to not serve MIME types that are not specifically registered. None of the web service files are registered. This protects these files by default.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation was clear. It provided a warning that 404.3 errors are returned for extension types that are not in the MIME type list.
	Implementor Competence	<b>Novice (5)</b> . No developer skills required, only simple administrator skills.
	Time to Implement	<b>Low (5)</b> . Default behavior protected web service config files.

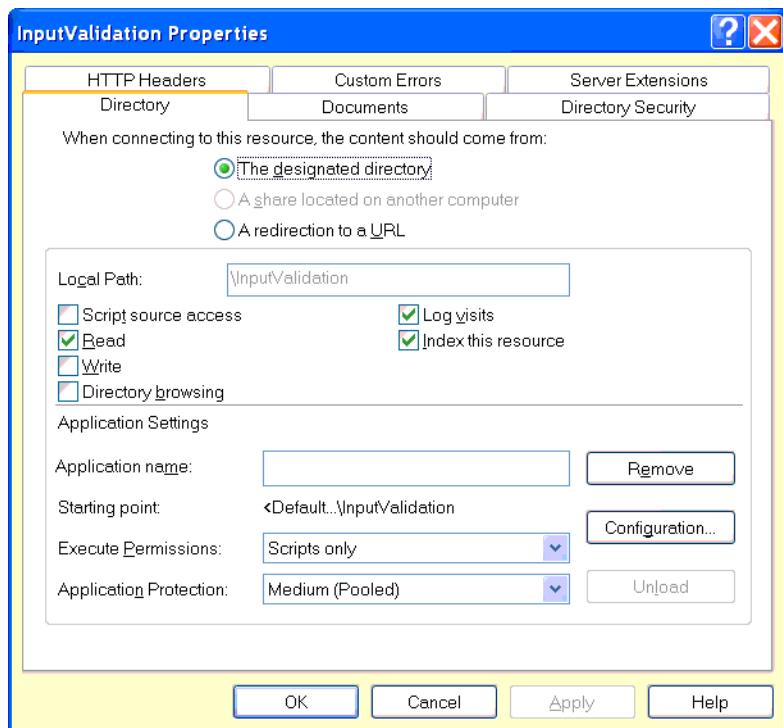
**Ease of Securing score**, based on mean of 5 test cases: 4.6

### Notes

#### Test Case 25.1

This is default functionality of the web server. New directories or virtual servers automatically deny directory listings.

Turning this feature on or off can be done through the following wizard:



### Test Case 25.2

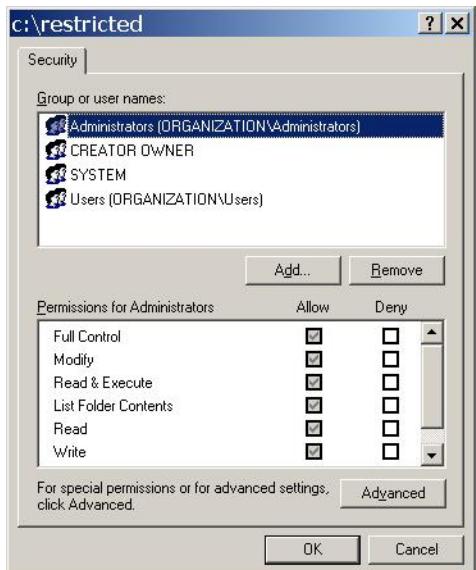
Did not need to change the configuration as it is the default behavior.

### Test Case 27.1

Select an authentication type that works off of the Windows authentication store



Set the permissions on a object to allow a particular Windows group access



## IBM WebSphere J2EE

## Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=25]. The platform restricts directory listings upon receipt of a URL with a terminating slash, or with a URL containing a context or sub context that does not otherwise map to a named document or welcome page.		<b>Wizard (4)</b> . Modifications to directories is accomplished by editing the <code>httpd.conf</code> file.	
[id=27]. The platform granularly restricts objects under the webroot.	The WebSphere application server provides highly granular access control methods over static and dynamic resources. See the analysis topic Role-Based Access Control for more details.	<b>Transparent (5)</b> . See the analysis topic Role-Based Access Control for details.	No recommendations; WebSphere implements the best practice
[id=28]. The platform supports restricted file systems and webroots.	Given the disparities between the platforms as to what is and is not considered a "restricted" file system, the initial portion of the best practice does not apply to platforms where all file system objects are direct descendants of a root node. (e.g.Unix). Under these platforms, the Unix system call <code>chroot()</code> would implement file system restrictions	<b>Developer implements (2)</b> . Though not specifically requiring <i>implementation</i> , the required modifications were deemed so radical as to warrant the rating given	

**Best Practice Compliance scores**, based on mean of 3 best practices: Linux 3.67, Unix 3.67

## Level of Effort Analysis

Test Case	Criteria	Rating
[id=25.1]. Ensure that the default behavior for a given directory is to deny directory listings.	Implementation Complexity	<b>Wizard+ (4)</b> . By default, the IBM HTTP Server does not restrict directory listing to resources under the web root. However, configuring the server to disallow directory listing was easily accomplished by editing the <code>httpd.conf</code> files.
	Documentation and Examples	<b>Suitable (4)</b> . Documentation supplied by the Apache web site was accurate and to the point.
	Implementor Competence	<b>Intermediate (3)</b> . The tester has wide experience in configuring and deploying web servers.
	Time to Implement	<b>Low to Medium (4)</b> . The modifications to the Web Server configuration file was accomplished in short order
[id=25.2]. Configure the platform to restrict directory listing.	Implementation Complexity	<b>Wizard+ (4)</b> . Configuring the server to disallow directory listing was easily accomplished by editing the <code>httpd.conf</code> files.
	Documentation and Examples	<b>Suitable (4)</b> . Documentation supplied by the Apache web site was accurate and to the point.
	Implementor Competence	<b>Intermediate (3)</b> . The tester has wide experience in configuring and deploying web servers.
	Time to Implement	<b>Low to Medium (4)</b> . The modifications to the Web Server configuration file was accomplished in short order

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=27.1]. Restrict access to a given role to a file, folder, or object.	Implementation Complexity	<b>Wizard (5)</b> . See the analysis topic Role-Based Access Control for details.
	Documentation and Examples	<b>Suitable (4)</b> . See the analysis topic Role-Based Access Control for details.
	Implementor Competence	<b>Novice (5)</b> . See the analysis topic Role-Based Access Control for details.
	Time to Implement	<b>Low (5)</b> . See the analysis topic Role-Based Access Control for details.
[id=28.1]. Ensure that webroots are deployed or recommended to be placed in restricted file system locations by default.	Implementation Complexity	<b>Medium amount of code (2)</b> . The platform does not natively support a chroot() environment. In order to fulfill the requirements of the test case, a series of complex steps must be taken. These steps are outlined in detail in the notes section that follows
	Documentation and Examples	<b>Incorrect or insecure (1)</b> . No documentation was supplied by either IBM or the Apache group. Configuration steps required the expertise of @stake consultants to accomplish
	Implementor Competence	<b>Expert/intermediate (2)</b> . Modifications of this nature require a Unix systems administrator with a high level of expertise
	Time to Implement	<b>Medium to High (2)</b> . Implementing a chrooted environment for an expert took slightly more than one hour.
[id=28.2]. Ensure that the platform prevents sensitive web service files from being served.	Implementation Complexity	<b>Wizard (5)</b> . By default, the IBM HTTP Server does not store sensitive configuration files in areas accessible to client browsers. No configuration changes are required
	Documentation and Examples	<b>Suitable (4)</b> . Suitable documentation was found regarding the general security of the server installation. The document covered Permissions on ServerRoot directories, and the protection of system settings issues in a clear and concise manner.
	Implementor Competence	<b>Intermediate (3)</b> . The tester has wide experience in configuring and deploying web servers.
	Time to Implement	<b>Low (5)</b> . No modifications to the Web Server were required

**Ease of Securing scores**, based on mean of 5 test cases: Linux 3.65, Unix 3.65

### Notes

#### Test Case 25.1

For this test case, a sub-directory `../htdocs/TestCase_25_1` was created under the webroot. Using a browser to navigate to the sub-directory showed that directory listing was enabled by default.

When using the property sheet wizard in the IBM Administration Server Console, it was not possible to restrict directory listing to this location.

The property sheet wizard affected the following change to the `httpd.conf` file:

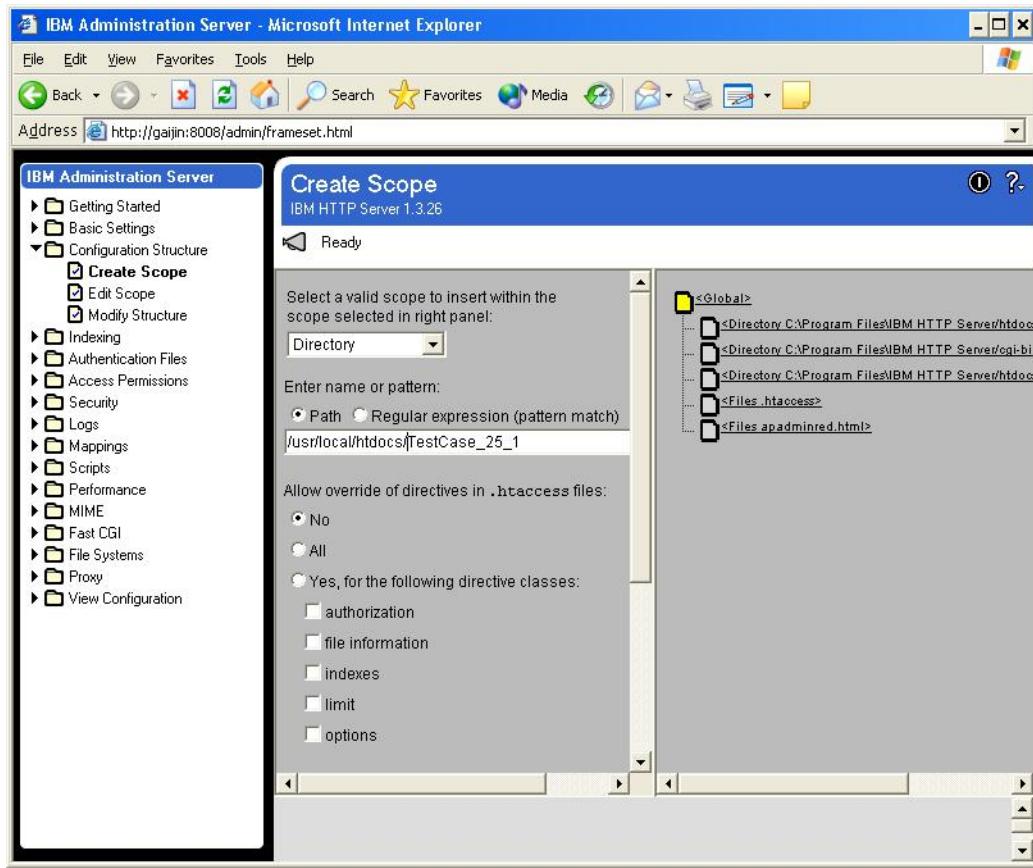
```
<Directory /usr/local/htdocs/TestCase_25_1>
  AllowOverride none
  UseCanonicalName Off
</Directory>
```

By default, the wizard uses a partial *default-allow* security model, that is to say, some aspects of security are addressed by default. For example, the wizard creates configuration directives that disallow the creation of .htaccess files which can override other security configurations, but does not preclude directory browsing. Furthermore, the property sheet wizard is configured to only *disallow* specific directives. There is no facility in the wizard to deny otherwise open object permissions.

A correct implementation of this test case required the following modification to the default <Directory> configuration directive:

```
<Directory /usr/local/htdocs/TestCase_25_1>
  AllowOverride none
  UseCanonicalName Off
  Options -Indexes
</Directory>
```

The Image below shows the simple property sheet wizard used to modify access controls on resources under the webroot.



The Image below illustrates directory listing enabled by default



## Documentation Sources

The Apache web site provided documentation describing Configuring Object Access including the use of the `Directory` configuration directive

### Test Case 25.2

The notes for this test case are aggregated above with test case 1.

### Test Case 28.1

A step-by-step example of the required modifications to the IBM HTTP Server is included below:

**NB:** It is not clear from IBM documentation whether a recompilation of Apache would void any service or support agreements between IBM and a customer. The choice of rebuilding as opposed to coping large quantities of files and directories from a previously build IBM HTTP Server was made for the lack of speed and clarity of the test case

### Steps Required to Prepare Apache to run in a chrooted environment

1. Set up the target directory environment on a non-system disk partition.  
`# mkdir /home/apache/www # ln -s /home/apache/www /www`

2. Create the basic directories; bin will be a symbolic link to `usr/bin`

```
# cd /www # mkdir -p usr/bin usr/lib lib etc tmp dev webhome # ln -s
usr/bin bin
```

3. Due to the requirements of a temporary or scratch location in the file system `/tmp` is given special permissions.

```
# chmod 777 tmp # chmod +t tmp
```

4. Create the `/dev/null` device:

```
# mknod -m 666 dev/null c 1 3 5.
```

5. Set the TIMEZONE that the web server will use:

```
# mkdir -p usr/share/zoneinfo # cp -pi /usr/share/zoneinfo/PST
usr/share/zoneinfo/ # cd etc # ln -s ../usr/share/zoneinfo/PST localtime
```

6. Set the locale the web server will use:

```
# set |grep LANG LANG=en_US # mkdir /www/usr/share/locale# cp -a
/usr/share/locale/en_US /www/usr/share/locale/
```

7. Copy required shared libraries

```
# cp -pi /lib/libtermcap.so.2 /lib/ld-linux.so.2 /lib/libc.so.6 lib/
```

8. Prepare a User id and the Naming Service

```
# cd /www # touch etc/passwd etc/group etc/shadow# chmod 400 etc/shadow#
echo 'www:x:501:501:Web Account:/webhome:/usr/bin/False' > etc/passwd#
echo 'www:x:501:' > etc/group# echo
'www:*:10882:-1:99999:-1:-1:134537804' > etc/shadow
```

9. Mark all binaries execute-only:

```
# chmod 111 usr/bin/*
```

10.The details of creating and setting up a Directory Naming Service (DNS) will be different on the two Unix platforms under evaluation. This portion of the test-case will assume an Apache server executing under linux.

```
# cp -pi /lib/libnss_files.so.2 lib/ # cp -pi /lib/libnss_dns.so.2 lib/
```

**NB:** Three additional files are required to complete the configuration for DNS. Files contents are IP and DNS installation dependent.  
`etc/nsswitch.conf` `etc/resolv.conf` `etc/hosts`

11Create the top-level directory for the Apache install and create a symbolic link to it in the actual directory structure:

```
# mkdir /www/apache # ln -s /www/apache /apache
```

12Compile and install the Apache web server using a non-privileged user identification

```
$ cd /usr/local/src/ $ tar zxf /path/to/apache_2.0.tar.gz $ cd apache_2.0
Additional software build steps removed
```

13Copy shared libraries minimally required by Apache.

```
# cd /www # cp -pi /lib/libm.so.6 /lib/libcrypt.so.1 /lib/libdb.so.3 lib/ #
cp -pi /lib/libdl.so.2 lib/
```

## Documentation Sources

*Specific documentation on implementing this best practice was not available on either vendor web site.*

### Test Case 28.2

In keeping with the spirit of the evaluation, the analysis contrived a test case in which it was possible to store sensitive information in the same directory containing content.

For the lack of simplicity, the directory was assumed to contain static content in the form of HTML files. A simple Apache `<Directory>` configuration directive was constructed that limited access to files that match a regular expression pattern contained in a `<Files>` directive. The entire time to implement the test was under 10 minutes. The directive is included below:

```
<Directory "/usr/local/htdocs/TestCase_28_1">
    Order Deny,Allow
    Deny from all
    <Files ~ "\.htm\$">
        Allow from all
    </Files>
    AllowOverride none
    UseCanonicalName Off
    Options -Indexes
</Directory>
```

## Documentation Sources

The Apache web site provided documentation describing Security tips for protecting the installation

## 4.4.5 Session Management

---

### Session Management

A series of requests to a Web-based application originating from the same user at the same browser is identified by a unique Session Identifier and comprises a session. The Session Identifier can be tracked as cookies or appended to the URL request, and allows only authorized users access to the application.

The Session Management area of analysis includes these topics:

- Cookie Handling
- Session Identifier
- Session Lifetime

The table below describes the topics at a high level, and lists prevailing best practices. The right-hand columns show the weights @stake assigned for each of the scenarios.

Analysis Topic	Best Practices	Scenario Weights		
		Web Application	Web Service	Intranet
<b>Cookie Handling</b> . Secure cookie handling is a requirement for all types of applications, so it is weighted medium for all three scenarios.	Cookies store only a session id, with other state characteristics managed server-side. All cookies set via HTTPS will have the SECURE parameter set, to prevent them from being sent in the clear via HTTP.	2	2	2
<b>Session Identifier</b> . Secure session identifiers is a critical requirement for all types of applications. All three scenarios are weighted high.	Session identifiers are opaque, unpredictable, and unique in order to help prevent session hijacking. The application should not use URL rewriting as a way to track sessions due to the fact that this information gets stored in web server logs, browser history and other areas, creating a window of opportunity for session hijacking.	3	3	3
<b>Session Lifetime</b> . Session lifetime control is most important for applications where session information is transmitted over an uncontrolled medium such as the Internet. Session lifetime is rated high for web applications and web services and low for intranet applications.	Server-side sessions expire after a short amount of time, typically 10-20 minutes to limit the window of opportunity for session hijacking.	3	3	1

Scenario weights range from 1 (relatively unimportant) to 3 (extremely important). A scenario weight of zero means that the topic does not apply.

### Findings

The table below summarizes @stake's quantitative findings. For each analysis topic, the two high-level metrics (Best Practice Compliance and Ease of Securing) are shown. These are the arithmetic means of the underlying best practice and test case scores. The Ease of Securing metric is further broken down into four sub-metrics.

Topic	Platform	Best Practice Compliance	Ease of Securing				
			Overall	Implementation Complexity	Documentation and Examples	Implementor Competence	Time to Implement
Cookie Handling <i>2 best practices, 2 test cases</i>	.NET Windows	<b>3.5</b>	<b>3.5</b>	3.5	3.5	3	4
	WebSphere Linux	<b>4.5</b>	<b>4.5</b>	4.5	4	4.5	5
	WebSphere Unix	<b>4.5</b>	<b>4.5</b>	4.5	4	4.5	5
Session Identifier <i>1 best practice, 1 test case</i>	.NET Windows	<b>5</b>	<b>4.25</b>	5	5	2	5
	WebSphere Linux	<b>5</b>	<b>4.75</b>	5	4	5	5
	WebSphere Unix	<b>5</b>	<b>4.75</b>	5	4	5	5
Session Lifetime <i>1 best practice, 1 test case</i>	.NET Windows	<b>5</b>	<b>4.75</b>	5	4	5	5
	WebSphere Linux	<b>5</b>	<b>3.75</b>	5	1	4	5
	WebSphere Unix	<b>5</b>	<b>3.75</b>	5	1	4	5

Session management greatly enhances the functionality and user experience of otherwise stateless HTTP based applications. Among other services, sessions allow applications to provide transactions, customized user interfaces, and data persistence. @stake developed four (4) test cases to evaluate the platform's ability to provide typical session management functionality. Including:

- Cookie Handling
- Session Identifier
- Session Lifetime

Of the four test cases executed, Microsoft and IBM were closely matched in overall scoring. This result was not un-expected, session management is a mature well understood concept, both vendors provide satisfactory solutions to implement this functionality.

Microsoft fared poorly in both documentation and implementation complexity in the Cookie Handling test cases. IBM could have closed the otherwise small gap in overall scores but received the lowest possible score (1) for Quality of Documentation and Sample Code in the Session Lifetime test cases.

The individual sections for each analysis topic describe @stake's findings in further detail.

## 4.4.5.1 Cookie Handling

---

### Cookie Handling

Secure cookie handling is a requirement for all types of applications, so it is weighted medium for all three scenarios. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 2
- Web Service: 2
- Intranet: 2

The sections that follow contain the results of @stake's analysis of the Cookie Handling topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=13]. Cookies store only a session id, with other state characteristics managed server-side.		<b>Transparent (5)</b> . The default mechanism for managing state across client requests provided by the .NET Framework stores the state information on the server, with only a session identifier contained in the cookie.	None
[id=14]. All cookies set via HTTPS will have the SECURE parameter set, to prevent them from being sent in the clear via HTTP.		<b>Developer implements (2)</b> . There is no global setting for enforcing that the SECURE cookie parameter is set for cookies sent over SSL. The ASP.NET service that form based authentication does not set the SECURE cookie parameter.	System administrators do not have the capability to enforce the SECURE parameter on a global or even an application by application basis. The application developers must provide this functionality. Developers should consider extending the FormsAuthentication or HttpCookie classes to provide this function and reuse those extended classes.

**Best Practice Compliance score**, based on mean of 2 best practices: 3.5

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=13.1]. Create a HTTP session; verify that cookies contain only session-id data.	Implementation Complexity	<b>Wizard (5)</b> . Using sessionId cookies is the default behavior and no program logic is contained therein.
	Documentation and Examples	<b>Best practice (5)</b> . Documentation is adequate.
	Implementor Competence	<b>Expert/intermediate (2)</b> . Tester is an experienced programmer.
	Time to Implement	<b>Low (5)</b> . Using sessionId cookies is the default behavior.
[id=14.1]. Create session; ensure that the cookies received over insecure channels are properly encrypted.	Implementation Complexity	<b>Medium amount of code (2)</b> . The developer can extend the FormAuthentication class to set the SECURE property of the HTTPCookie object. Developers must remember to set the SECURE property on all the HTTPCookies they use if the transport is SSL.
	Documentation and Examples	<b>Vague or incomplete (2)</b> . The documentation for the SECURE property is contained in the HTTPCookie object reference. This is adequate. The documentation for higher level classes that use cookies for security such as forms based authentication makes no mention of the SECURE property.
	Implementor Competence	<b>Novice/intermediate (4)</b> . The code required to add the SECURE property to a cookie is not complex. The developer must know how to extend a base class or handle an event if the SECURE property is added to a higher level class such as FormAuthentication.
	Time to Implement	<b>Medium (3)</b> .
<b>Ease of Securing score</b> , based on mean of 2 test cases: 3.5		

### Notes

#### Test Case 13.1

```
<!-- SESSION STATE SETTINGS By default ASP.NET uses cookies to identify which requests belong to a particular session. If cookies are not available, a session can be tracked by adding a session identifier to the URL. To disable cookies, set sessionState cookieless="true". --> <sessionState mode="InProc" stateConnectionString="tcpip=127.0.0.1:42424" cookieless="false" timeout="20" />
```

### IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=13]. Cookies store only a session id, with other state characteristics managed server-side.	The default mechanism for managing state across client requests provided by the J2EE framework stores the state information on the server, with only a session identifier contained in the cookie.	<b>Transparent (5)</b> . The default mechanism for managing state applies security best practices.	None
[id=14]. All cookies set via HTTPS will have the SECURE parameter set, to prevent them from being sent in the clear via HTTP.	Application cookies are controlled by the application server. When deploying the application, the application server administrator can specify cookie preferences on a server-wide or application-specific basis. By default, the server settings take preference. Cookies can be restricted to HTTPS-only so that they are not sent in the clear over HTTP.	<b>Wizard (4)</b> . Secure cookies are not set automatically. However, a simple checkbox on the Administration Console is all that is required to implement secure cookies.	Application server administrators should enable secure cookies for the server on a global basis. This will prevent the need to do this on an application-by-application basis.

**Best Practice Compliance scores**, based on mean of 2 best practices: Linux 4.5, Unix 4.5

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=13.1]. Create a HTTP session; verify that cookies contain only session-id data.	Implementation Complexity	<b>Wizard (5)</b> . The default mechanism for maintaining state across an HTTP session is via a session-id cookie. Cookies contain only session identifier related data, with possibly some load balancing information.
	Documentation and Examples	<b>Best practice (5)</b> . The documentation includes best practices for using HTTP sessions.
	Implementor Competence	<b>Novice/Intermediate (4)</b> . The test case implementer has wide ranging experience in developing and deploying J2EE based web applications
	Time to Implement	<b>Low (5)</b> . Using server side management of state requires no additional development effort over the normal process of WebSphere application development.
[id=14.1]. Create session; ensure that the cookies received over insecure channels are properly encrypted.	Implementation Complexity	<b>Wizard+ (4)</b> . Secure cookies are enabled through a simple property sheet on the application server's Administration Console.
	Documentation and Examples	<b>Adequate (3)</b> . Documentation consisted of a help file entry in the Administration Console. No examples were given (arguably, this is not really needed). The need to use secure cookies, as a best practice, was not explained.
	Implementor Competence	<b>Novice (5)</b> . No special skill-sets are needed to implement this functionality.
	Time to Implement	<b>Low (5)</b> . If the cookie settings are applied to the server globally, implementation requires a server re-start. If applied to a single application, only that application needs to be re-started. Either way, the total time required is less than five minutes.

**Ease of Securing scores**, based on mean of 2 test cases: Linux 4.5, Unix 4.5

## Notes

### Test Case 14.1

A simple property sheet in the Administration Console implements session cookies for a server or on an application-by-application basis. This screen shot shows how secure session cookies were implemented for the ‘Best Practices’ application.

[Enterprise Applications > BestPractices > Session Management >](#)

#### Cookies

Specify cookie settings for http session management. [i](#)

General Properties	
Cookie name	<input type="text" value="JSESSIONID"/> <a href="#">i</a> A unique name for the cookie to be used for session management. The servlet specification requires this name to be JSESSIONID. However, for flexibility this value is configurable.
Secure cookies	<input checked="" type="checkbox"/> Restrict cookies to HTTPS sessions <a href="#">i</a> Whether session cookies include the secure field. Enabling the feature will restrict the exchange of cookies only to HTTPS sessions.
Cookie domain	<input type="text"/> <a href="#">i</a> The value of the domain field of a session tracking cookie. This value will dictate to the browser whether or not to send a cookie to particular servers. For example, if you specify a particular domain, session cookies will be sent only to hosts in that domain. The default domain is the server.
Cookie path	<input type="text" value="/"/> <a href="#">i</a> This dictates browser whether cookie is sent to the URI requested based on the path. Specify any string representing a path on the server. “/” indicates root directory. Specify a value in order to restrict the paths to which the cookie will be sent. By restricting paths, you can keep the cookie from being sent to certain URLs on the server. If you specify the root directory, the cookie will be sent no matter which path on the given server is accessed.
Cookie maximum age	<input type="radio"/> Current browser session <input checked="" type="radio"/> Set maximum age <input type="text" value="60"/> seconds <a href="#">i</a> The amount of time that the cookie will live on the client browser. This value corresponds to the Time to Live (TTL) value described in the Cookie specification.
<input type="button" value="Apply"/> <input type="button" value="OK"/> <input type="button" value="Reset"/> <input type="button" value="Cancel"/>	

## 4.4.5.2 Session Identifier

---

### Session Identifier

Secure session identifiers is a critical requirement for all types of applications. All three scenarios are weighted high. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 3
- Web Service: 3
- Intranet: 3

The sections that follow contain the results of @stake's analysis of the Session Identifier topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=17]. Session identifiers are opaque, unpredictable, and unique in order to help prevent session hijacking. The application should not use URL rewriting as a way to track sessions due to the fact that this information gets stored in web server logs, browser history and other areas, creating a window of opportunity for session hijacking.	Though some curious characteristics exist, after cursory analysis the session identifiers created by the .NET Framework appear to be unique, unpredictable, opaque, and resistant to attack. URL rewriting is not in use by default.	<b>Transparent (5)</b> . The sessionId cookies appear to be three independently radix32 encoded values. The individual values appear to be suitably random.	None.

**Best Practice Compliance score**, based on mean of 1 best practice: 5

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=17.1]. Ensure that session identifiers are statistically random.	Implementation Complexity	<b>Wizard (5)</b> . The default behavior of .NET applications use statistically random sessionIds.
	Documentation and Examples	<b>Best practice (5)</b> . Documentation on the usage of cookies is adequate.
	Implementor Competence	<b>Expert/intermediate (2)</b> . Tester is an experienced developer.
	Time to Implement	<b>Low (5)</b> . The implementation of cookies is automatic.

**Ease of Securing score**, based on mean of 1 test case: 4.25

### Notes

#### Test Case 17.1

'MUST' is a utility for determining the randomness of a bitstream. The expected value, for a random stream or suitable pseudo-random stream such as /dev/random, is 7.183. Testing on the decoded cookies file consistently returned results approximately 6.85. Because this value was consistent, it may be due to a byte boundary or some other error. Even with this curious nature and the characteristics listed below, it was deemed the cookie data is random enough to cause attacks against it to be impractical.

Below is an analysis of the radix32-encoded cookie, without converting it to binary. It was determined the character set for the encoding was [0-5a-z].

```
Cookies display a fairly even distribution of characters, with twice the average rate of incidence for '4' and four times the average for '5'. The normal percentage of incidence is 2.53-2.94, while '4' occurred 5.66% and '5' occurred 11.92% 

Rates of incidence for all positions within the cookie were fairly statistically spread with the following notes:

Position 7 - '4' is 25.10% of the values and '5' is 24.95% of the values
Only 18 values are present: 2345abefijmnqruvyz

Position 8 - '5' accounts for 51.57% of the values

Position 15 - '4' is 24.82% and '5' is 25.11% of the values
Only 18 values are present: 2345abefijmnqruvyz

Position 16 - '5' accounts for 51.47% of the values

Position 23 - '4' is 24.96% and '5' is 25.03% of the values
Only 18 values are present: 2345abefijmnqruvyz

Position 24 - '5' accounts for 51.55% of the values.
```

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=17]. Session identifiers are opaque, unpredictable, and unique in order to help prevent session hijacking. The application should not use URL rewriting as a way to track sessions due to the fact that this information gets stored in web server logs, browser history and other areas, creating a window of opportunity for session hijacking.	The J2EE framework provides a default session identifier mechanism that has the desired properties for preventing session hijacking attacks, including appearing sufficiently random to prevent attackers from easily guessing valid identifiers. While the default mechanism is to use cookies for passing session identifiers, URL rewriting is possible through a simple configuration change, and the documentation does not specifically identify the security implications of using URL rewriting.	<b>Transparent (5)</b> . The default mechanism for session management applies security best practices. However, it is possible to avoid these best practices by changing the default configuration, and the documentation does not appear to identify the security implications of those changes.	Consider documenting the security implications of using URL rewriting for session security so that developers understand the tradeoffs.

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 5, Unix 5

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=17.1]. Ensure that session identifiers are statistically random.	Implementation Complexity	<b>Wizard (5)</b> . WebSphere provides a session identifier mechanism, and no additional effort is required to ensure that the session identifiers are statistically random.
	Documentation and Examples	<b>Suitable (4)</b> . The documentation does contain some suggested best practices for using HTTP sessions, and suggests the use of SSL, but does not appear to suggest avoiding URL rewriting for maintaining sessions. There is also no discussion of why the session identifiers need to be random.
	Implementor Competence	<b>Novice (5)</b> . Use of sessions in WebSphere will by default use the statistically random session identifiers provided by the application environment.
	Time to Implement	<b>Low (5)</b> . Using random session identifiers requires no additional development effort over the normal process of WebSphere application development.

**Ease of Securing scores**, based on mean of 1 test case: Linux 4.75, Unix 4.75

### Notes

#### Test Case 17.1

Session identifier cookies from one tested WebSphere 5.0 installation look as follows:

```
Set-Cookie: JSESSIONID=000050F5Z3A0LTA34UPZUKP0I5Q:-1;Path=/
Set-Cookie: JSESSIONID=0000CL4SJ1XMNR3HZDCTJ5KYPMQ:-1;Path=/

```

```
Set-Cookie: JSESSIONID=0000JSTZCSHS0FCDKFOMLQMZBGI:-1;Path=/  
Set-Cookie: JSESSIONID=0000XBBEX3IZYSTZRQZEB3A1VLI:-1;Path=/  
Set-Cookie: JSESSIONID=0000HDTAQXQCCAXRLGGJ5OW0IAI:-1;Path=/
```

The structure of the identifier is four 0s followed by 22 characters in the set [A-Z0-5] (32 distinct characters, indicating base 32 encoding) followed by a character in the set [AIQY].

The characters in positions 8, 16, and 24 exhibit the interesting property that characters in the set [YZ0-5] appear approximately twice as often as the other characters.

Other than the characteristics noted above, an examination of ten thousand sample session identifiers identified no statistical anomalies. If there are in fact no hidden patterns in the generated identifiers, the size of the session identifier space is well over 295 ( $3.9 \times 1028$ ), making attacks that rely on guessing identifiers impractical.

The session identifiers are generated using the Java SecureRandom class. This is a software-based form of entropy generation, and is therefore less well understood from a statistical modeling perspective than current hardware-based methods. However, the conservative approach in the size of the session identifier significantly reduces any potential risk related to poor entropy.

#### 4.4.5.3 Session Lifetime

---

### Session Lifetime

Session lifetime control is most important for applications where session information is transmitted over an uncontrolled medium such as the Internet. Session lifetime is rated high for web applications and web services and low for intranet applications. @stake weighted the calculated scores per-scenario as follows:

- Web Application: 3
- Web Service: 3
- Intranet: 1

The sections that follow contain the results of @stake's analysis of the Session Lifetime topic. Each target of evaluation has two exhibit tables: the Best Practice Analysis and the Level of Effort Analysis.

- Microsoft .NET
- IBM WebSphere J2EE

### Microsoft .NET

#### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=16]. Server-side sessions expire after a short amount of time, typically 10-20 minutes to limit the window of opportunity for session hijacking.	All ASP.NET applications have a sessionId associated with them. The sessionId timeout defaults to 20 minutes. It can be modified by editing an application specific configuration file, <code>web.config</code> . The value can also be modified at run time by setting a value in the <code>HttpSessionState</code> object. The session is maintained by a 120 bit unique identifier encoded as a cookie. If forms based authentication is used, the <code>FormsAuthentication</code> class uses a cookie mechanism to implement an authentication session. This session object defaults to a 30 minute timeout. It can be modified via a per application configuration file or at run time by setting the expiration time in the <code>FormsAuthenticationTicket</code> object.	<b>Transparent (5)</b> . The sessionId session state mechanism built into .NET provides a default timeout of 20 minutes. This value can be easily changed via an application specific configuration file, <code>web.config</code> . The value can also be modified at run time by setting a value in the <code>HttpSessionState</code> object.	Most application will not need to modify the default timeout values.

---

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
<b>Best Practice Compliance score</b> , based on mean of 1 best practice: 5			

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=16.1]. Create a session; set expiration limit.	Implementation Complexity	<b>Wizard (5)</b> . No code is required to implement timeout functionality or set the timeout value. The default behavior of the built in session object meets best practices.
	Documentation and Examples	<b>Suitable (4)</b> . There was good documentation concerning the state management options and function available to the developer. The timeout mechanism defaults and how to modify them was well documented. There was no good sample code demonstrating how to actually manage session timeout.
	Implementor Competence	<b>Novice (5)</b> . The test case implementer is an expert in the development and deployment of Windows based web applications.
	Time to Implement	<b>Low (5)</b> . No implementation necessary.

**Ease of Securing score**, based on mean of 1 test case: 4.75

### Notes

#### Test Case 16.1

The timeout mechanism was verified using the `FormAuthentication` sample application by logging in and setting the system time into the future. Once the time was set outside the timeout window the session failed and re authentication was required. The web traffic and cookies were verified using a the `@stake WebProxy` tool.

## IBM WebSphere J2EE

### Best Practice Compliance Analysis

Best Practice	Options	Rating	Recommendation
[id=16]. Server-side sessions expire after a short amount of time, typically 10-20 minutes to limit the window of opportunity for session hijacking.	WebSphere enables administrators to specify session timeout via the Administrative Console. The default timeout is 30 minutes, with a minimum of 2 minutes. J2EE's <code>HttpSession.setMaxInactiveInterval(int)</code> method supports specifying session timeout in seconds.	<b>Transparent (5)</b> . The default 30 minute session timeout should address most users' security needs.	Set session time outs through the Administrative Console rather than programmatically to ensure consistency and control of session lifetimes.

**Best Practice Compliance scores**, based on mean of 1 best practice: Linux 5, Unix 5

### Level of Effort Analysis

Test Case	Criteria	Rating
[id=16.1]. Create a session; set expiration limit.	Implementation Complexity	<b>Wizard (5)</b> . The default is not only reasonably secure, but simple to change.
	Documentation and Examples	<b>Incorrect or insecure (1)</b> . While the most of the session-related documentation is well-written and extensive, at least one best practice mentioned could lead developers to build insecure applications (see below).
	Implementor Competence	<b>Novice/intermediate (4)</b> . The developer has extensive experience developing web applications and limited skills in configuring/administering WebSphere application servers.
	Time to Implement	<b>Low (5)</b> . Only one setting need be changed to modify the session timeout.

**Ease of Securing scores**, based on mean of 1 test case: Linux 3.75, Unix 3.75

### Notes

#### Test Case 16.1

The Administrative Console enables users to set session timeout parameters:

[Application Servers > server1 > Web Container >](#)**Session Management**

Session manager configuration properties allow you to control the behavior of HTTP session support [i](#)

Configuration		
General Properties		
Session tracking mechanism:	<input type="checkbox"/> Enable SSL ID tracking <input checked="" type="checkbox"/> <a href="#">Enable Cookies</a> <input type="checkbox"/> Enable URL Rewriting <input type="checkbox"/> Enable protocol switch rewriting	<a href="#">i</a> Specify a mechanism for HTTP session management.
Maximum in-memory session count:	1000 sessions	<a href="#">i</a> Specifies the maximum number of sessions to maintain in memory.
Overflow:	<input checked="" type="checkbox"/> Allow overflow	<a href="#">i</a> Whether to allow the number of sessions in memory to exceed the value specified by Max In Memory Session Count property. This is valid only in non-persistent sessions mode.
Session timeout:	<input type="radio"/> No timeout <input checked="" type="radio"/> Set timeout 30 minutes	<a href="#">i</a> Specifies how long a session is allowed to go unused before it will be considered valid no longer. Specify either "Set timeout" or "No timeout." If you select to set the timeout, the value must be at least two minutes, specified in minutes.
Security integration	<input type="checkbox"/> Enable	<a href="#">i</a> When security integration is enabled, the Session Manager will associate the identity of users with their HTTP sessions.
Serialize session access:	<input type="checkbox"/> Allow serial access Maximum wait time : 5 seconds <input checked="" type="checkbox"/> Allow access on timeout	<a href="#">i</a> Serialize session access indicates whether to disallow concurrent session access in a given server (JVM).
<input type="button" value="Apply"/> <input type="button" value="OK"/> <input type="button" value="Reset"/> <input type="button" value="Cancel"/>		
Additional Properties		
<a href="#">Distributed Environment Settings</a> <a href="#">Specify sessions persistence type.</a>		

The documentation listed below mentions the following best practice:

HTTP sessions are identified by session IDs. A session ID is a pseudo-random number generated at the runtime. Session hijacking is a known attack HTTP sessions and can be prevented if all the requests going over the network are enforced to be over a secure connection (meaning, HTTPS).

This is incorrect and could lead developers to build insecure applications: hijacking an SSL-encrypted session is still possible through attacks such as cross-site scripting/tracing, guessing session ids, etc.

See also: Best Practices for Using HTTP Sessions.

## 5.1 References

---

### References

The Microsoft and IBM technologies evaluated in this study are complex subjects. Helpful books and published works consulted during the analysis included:

- **Building Secure Microsoft ASP.NET Applications.** Microsoft Press (January 2003)
- **IBM WebSphere V5.0 Security,** WebSphere Handbook Series. IBM Corporation (December 2002)
- **IBM WebSphere Application Server V5.0 System Management and Configuration,** WebSphere Handbook Series. IBM Corporation (Draft version, April 2003)
- **Improving Web Application Security.** Microsoft Press (September 2003)
- **Operating .NET Framework-based Applications.** Microsoft Press (July 2003)
- **WebSphere Version 5 Web Services Handbook,** WebSphere Handbook Series. IBM Corporation (March 2003)

Some of these works had not been published as of the period in which the analysis was conducted; in these cases, @stake reviewed pre-publication versions.

## 5.2 About the Authors

---

### About @stake

@stake, Inc., the premier digital consulting firm, provides security services and award-winning products to assess and manage risk in complex enterprise environments. The company's SmartRisk services cover key aspects of security, including applications, critical infrastructure, wireless and wired networks, storage systems, education, and incident readiness. @stake consultants combine technical expertise with a business focus to create comprehensive security solutions for industry leading companies in financial services, information technology, energy, utilities, healthcare, and telecommunications. As the first company to develop an empirical model that measures Return On Security Investment (ROSI), @stake keeps security investments in line with business requirements. Headquartered in Cambridge, MA, @stake has offices in London, New York, Raleigh, San Francisco, and Seattle. For more information, go to [www.atstake.com](http://www.atstake.com).

### About the Principal Authors

**Andrew Jaquith** has fifteen years of experience in information technology consulting. His professional experience includes systems integration, management consulting, application development and program management, with particular domain expertise in supply chain management and eCommerce strategy. At @stake, he serves as Program Director and manages strategic client relationships. He also directs the Hoover Project, an initiative within the firm to improve clients' abilities to quantify the risks and returns of security investment. His application security research has been featured in *CIO*, *CSO* and *Information Week*.

Andrew's recent Java experience includes auditing the security architecture of an embedded entertainment device; he also designed and developed a Jakarta Struts-based registration portal for Exostar, a major aerospace exchange. The Exostar service, for which @stake served as the security integrator of record, recently received Forbes "Best of the Web" and ComputerWorld's Honors awards. He has contributed to the Apache Jakarta Project, and is an unabashed fan of open source solutions.

Prior to @stake, Andrew was a Senior Project Manager at Cambridge Technology Partners, where he completed multiple assignments with Fortune 1000 and other corporate clients. Andrew's formative years were spent at the \$600M logistics division of FedEx Corporation, where he was an information technology analyst, project manager, and technology implementer. He has a BA from Yale University.

**Frank Heidt** is a Managing Security Architect for @stake's Pacific Northwest region, and is a recognized expert in the field of information assurance, network security and systems penetration. Over the last five years, Frank has been engaged in various computer security related work for the Department of Defense. In his capacity as one of the Department of Defense's few civilian Information Systems Security Officers (ISSO), he was responsible for securing several large networks for the Department of the Navy. Previous to

his military experience, Frank was a Developer at MCI Research specializing in Enhanced 911 system, principally The Geographic Information Systems Components and Automatic Number Information/Automatic Location Information [ANI/ALI] aspects of emergency response systems.

Frank is currently a Visiting distinguished lecturer at the United States Army War College, on the subject of defensive information warfare, and military computer systems security. Prior to @stake, he was a General Partner and Founding Member of Five By Five LLC, a security partnership that subsequently joined @stake as part of their corporate growth strategies.

**Chris Wysopal** is the @stake's Director of Research and Development. Though he has a computer engineering degree from Rensselaer Polytechnic Institute, almost all of what he knows about computer security he learned from his exploration of computers as a hacker for the past 15 years. As an associate of L0pht Heavy Industries he has worked to expose the "snake oil" in the computer security industry and tried to make the general public aware of the just how fragile the Internet and many computer security products are. He has released numerous public security advisories detailing security flaws in operating systems and web servers. He co-authored the famous Windows NT password cracking program, L0phtCrack. In May of 1998 he testified as a computer security expert before the Senate Governmental Affairs Committee and has appeared on several TV documentaries and network news programs.

Chris has worked on all sides of the fence. He started his career as a software engineer developing commercial software for over 10 years for Lotus, AT&T, and Radnet. He worked as a corporate IT security engineer for GTE trying to securely implement and maintain large networks and applications. Finally, as a research scientist for L0pht and @stake he has worked with vendors to identify and fix security problems.

Chris' areas of expertise are the Windows platform including NT and Windows 2000, web servers and web application software, password cracking, and networking.

## 5.3 Colophon

---

### Colophon

The printed (PDF) version of this document uses two principal typefaces. The body text typeface is Monotype Garamond. The display face, used for the table of contents, chapter headings, subheadings and tables is Helvetica.

This online and printed versions of this document were created entirely with open source Java and XML tools. The core document website and document generation package is based on Jakarta Maven, an Apache Software Foundation project. Certain static content, such as the Executive Summary, Objectives, Colophon and other pages were created as static XML files following the `xdoc` document format, which uses a markup grammar based on a subset of HTML. The rest of the document chapters, including all of the Analysis Findings, were dynamically generated from two XML files. The “topics” file defines the structure and methodology for the analysis, while the “results” file contains the ratings, notes, and detailed findings for each target of evaluation. Custom XSLT stylesheets use these two files to build the individual analysis pages subsequently processed by Maven.

Noteworthy tools used in the production of this document include:

- Apache Batik, for chart image rasterizing
- Apache FOP, for PDF generation
- Apache Maven, for document generation
- Apache Xerces Java 2, for XML parsing
- Apache Xalan Java 2, for XSLT transformation
- Concurrent Versioning Systems (CVS) for document revision management
- Various CVS support tools including TortoiseCVS, WinMerge, and CVS Conflict Editor
- JFreeChart, for charting numerical results
- JTidy and Langdale’s Styler, for XML syntax checking
- XSLT stylesheets, custom-built by @stake for this analysis
- Andy Clark’s NekoConv, a Java-based document transcoder, for text file Unicode conversions
- TextPad and JEdit with the Jazzy, JTidy, XML andWhiteSpace plugins, for authoring and markup