BEA WebLogic Server™

Securing A WebLogic Server Deployment

Release 7.0
Document Date: June 2002
Revised: September 6, 2002

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Securing A WebLogic Server Deployment

Part Number	Date	Software Version
N/A	September 6, 2002	BEA WebLogic Server Version 7.0

Contents

About This Document

This document explains how to use the security features of WebLogic Server to protect a WebLogic Server deployment. It is organized as follows:

- Chapter 1, "Security Implications for WebLogic Server," explains why security is important for WebLogic Server and lists questions you need to answer in order to determine the security needs of your WebLogic Server deployment.
- Chapter 2, "Security Best Practices," explains how to use the security features of WebLogic Server to protect your deployment.

Audience

This document is intended for customers who want to use WebLogic Server in a more secure manner.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation.

How to Print this Document

You can print a copy of this document from a Web browser, one file at a time, by using the File—>Print option on your Web browser.

A PDF version of this document is available on the WebLogic Integration documentation CD. You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format.

If you do not have the Adobe Acrobat Reader installed, you can download it for free from the Adobe Web site at `http://www.adobe.com/`.

Contact Us!

Your feedback on the WebLogic Integration documentation is important to us. Send us e-mail at **docsupport@bea.com** if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the WebLogic Integration documentation.

In your e-mail message, please indicate that you are using the documentation for BEA WebLogic Integration Release 7.0.

If you have any questions about this version of BEA WebLogic Integration, or if you have problems installing and running BEA WebLogic Integration, contact BEA Customer Support through BEA WebSUPPORT at **www.bea.com**. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using

- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and filenames and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*

Convention	Item
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
`...`	Indicates one of the following in a command line: ■ That an argument can be repeated several times in a command line ■ That the statement omits additional optional arguments ■ That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER

1 Security Implications for WebLogic Server

This topic explains why security is important for WebLogic Server and lists questions you need to answer in order to determine the security needs of your WebLogic Server deployment. The topic includes the following sections:

Why Is Security Important for WebLogic Server?

An application server resides in the sensitive layer between end users and your valuable data and resources. WebLogic Server provides authentication, authorization, and encryption services with which you can guard your resources. These services cannot provide protection, however, from an intruder who gains access by discovering and exploiting a weakness in your deployment environment.

Whether you deploy WebLogic Server on the Internet or on an intranet, it is a good idea to hire an independent security expert to go over your security plan and procedures, audit your installed systems, and recommend improvements.

Another good strategy is to read as much as possible about security issues. For the latest information about securing Web servers, BEA recommends reading the Security Improvement Modules, Security Practices, and Technical Implementations information available from the CERT™ Coordination Center operated by Carnegie Mellon University.

BEA suggests that you apply the remedies recommended in our security advisories. In addition, you are advised to apply every Service Pack as they are released. Service Packs include a roll up of all bug fixes for each version of the product, as well as each of the previously released Service Packs. As a policy, if there are any security-related issues with any BEA product, BEA will distribute an advisory and instructions with the appropriate course of action. If you are reponsible for security related issues at your site, please register to receive future notifications. BEA has established an e-mail address (`security-report@bea.com`) to which you can send reports of any possible security issues in BEA products.

There are partner products that can help you in your effort to secure the WebLogic Server production environment. For more information, see the BEA Partner's Page.

Tools to automate assement of security are available from the BEA Download Center. PentaSafe VigilEnt Security Agent can help assure the security of your application. For a quick assement of your application, download the free 30 day trial version.

Determine the Security Needs of Your WebLogic Server Deployment

Before securing your WebLogic Server deployment, it is important to understand the security needs of your WebLogic Server environment. To better understand the security needs, ask yourself the following questions:

- What WebLogic Server resources am I protecting?

  There are many resources in the WebLogic Server environment that can be protected including information in the database accessed by WebLogic Server, the availability of the Web site, the performance of the Web site, and the integrity of the Web site. Consider the resources you want to protect when deciding the level of security you must provide.

- From whom am I protecting the WebLogic Server resources?

  For most Web sites, resources must be protected from everyone on the Internet. But should the Web site be protected from the employess on the intranet in your enterprise? Should your employees have access to all WebLogic Server resources? Should the system administrators have access to all WebLogic Server resources? Should the system administrators be able to access all data? You might consider giving access to highly confidential data or strategic resources to only a few well trusted system administrators. Perhaps it would be best to allow no system administrators to access to the data or resources.

- What will happen if the protections on strategic resources fail?

  In some cases, a fault in your security scheme is easily detected and considered nothing more than an inconvenience. In other cases, a fault might cause great damage to companies or individual clients that use the Web site. Understanding the security ramifications of each resource will help you properly protect it.

As you read the suggestions in "Security Best Practices" on page 2-1, keep the answers to these questions in mind.

CHAPTER

2 Security Best Practices

This topic explains how to use the security features of WebLogic Server to protect your deployment. The topic contains the following best practices:

Secure the Machine on Which WebLogic Server Runs

A WebLogic Server deployment is only as secure as the security of the machine on which it is running. Therefore, it is important that you secure the physical machine, the operating system, and all other software that is installed on the host machine. The following are suggestions for securing the deployment machine, however, you should check with the manufacturer of the machine, operating system, and installed software for additional suggestions:

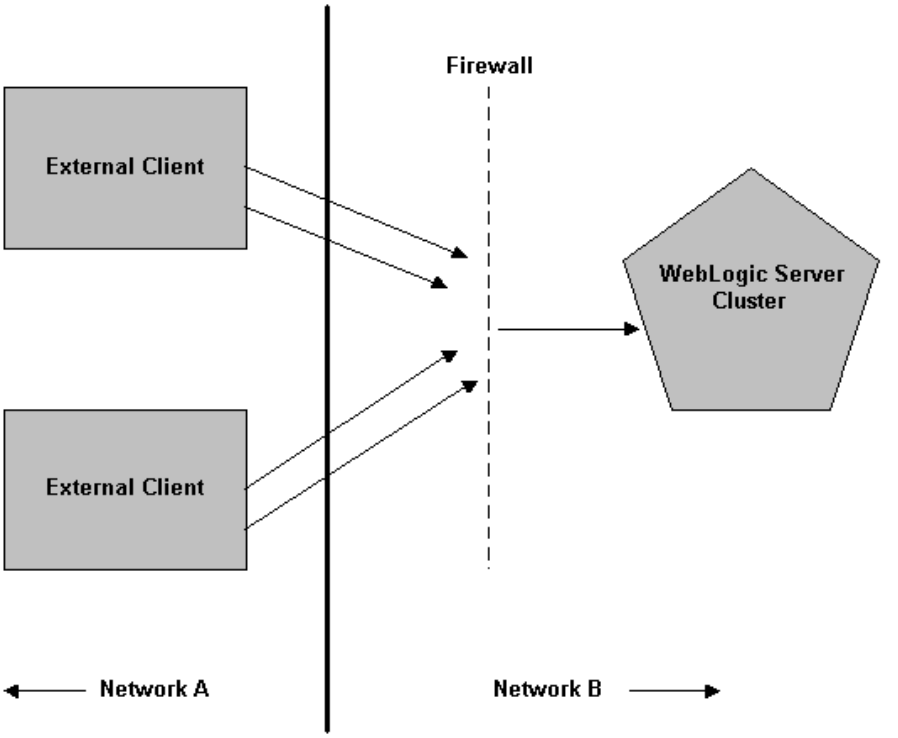
- Keep your hardware in a secured area to prevent unauthorized users from tampering with the deployment machine or its network connections.
- Have an expert review network services such as the e-mail program or directory service to ensure that there are no weaknesses that would permit a malicious attacker from accessing the operating system or system-level commands.
- Secure the file systems on the deployment machine, limiting directory and file access to a few, well-monitored user accounts. Some WebLogic Server configuration data (and some of your applications, including Java Server Pages (JSPs) and HTML pages are stored in clear text on the file system. A user or intruder with read access to files and directories can easily defeat any security mechanisms you establish with WebLogic Server authentication and authorization schemes.
- Avoid creating multiple user accounts on deployment machines and avoid sharing file systems with other machines in the enterprise network.
- Create a Weblogic user in the operating system and use the security controls of the operating system to give this user ownership and exclusive access to all files and directories in the WebLogic Server deployment. No other user needs write-access to any files in the WebLogic Server deployment.
- Review active user accounts regularly and when personnel leave. Set a policy to expire passwords periodically. Never code passwords in client applications.

Design Network Connections Carefully

When designing network connections, you want to use the easiest-to-manage solution. This choice must be weighed against the need to protect strategic WebLogic Server resources. Placing WebLogic Server resources further from potential intruders reduces the risk of a security breach. Inserting firewalls carefully in your enterprise increases security and can prevent a small security fault from turning into a security crisis. For example, it is a good idea to protect a database that contains critical data behind a firewall. In addition, protect the host machine for the database as well as the usernames and passwords for the database. Still, if someone acquires the username and passwords for the database, it is not nearly as damaging if the database is protected by a firewall and cannot receive connections from computers on the Internet.

There are many ways to combine firewalls, WebLogic Server, and other network servers. Figure 2-1 illustrates a typical setup with a firewall that filters traffic destined for a WebLogic Server cluster.

Figure 2-1 Typical Firewall Setup

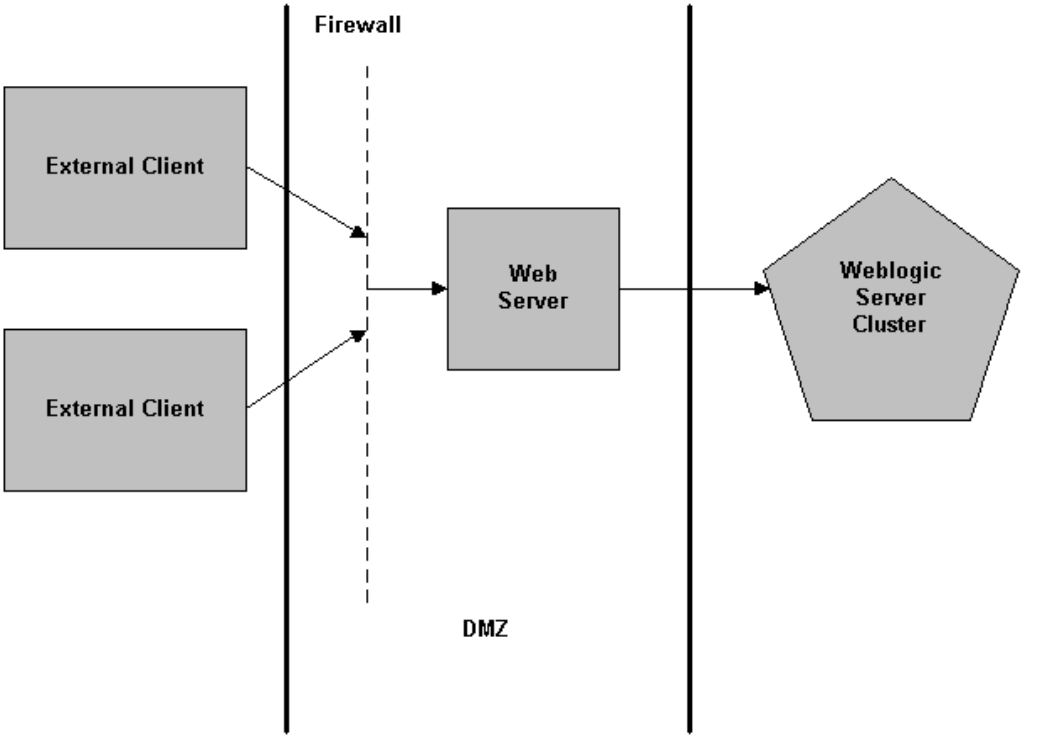


Another common firewall configuration restricts access to only HTTP or HTTPS Web connections. The firewall permits clients to connect only to a Web server which usually runs at the standard HTTP port 80 or HTTPs port 443. The Web server may be a WebLogic Server or a third-party Web server set up to proxy requests to a WebLogic Server. For example, Netscape Enterprise Server, Microsoft Internet Server, and Apache Server can be set up to serve static Web pages and proxy servlet and JSP requests to WebLogic Server. Figure 2-2 illustrates this configuration.

In Figure 2-2, the Web server is a gateway operating in a demilitarized zone (DMZ). In computer networks, a DMZ is a computer host or small network inserted as a neutral zone between a company's private network and the outside public network. It prevents outside users from getting direct access to a server on which company data resides. A DMZ is an optional and more secure implementation of a firewall which can also act as a proxy server. WebLogic Server connections come only from proxied Web server requests, enhancing the security of your WebLogic Server applications and back-end

resources. In the configuration shown in Figure 2-2, clients interact exclusively with the Web server and WebLogic Server connections are made only by proxied Web server requests. As a result, the security of WebLogic Server applications and back-end resources that are configured in this way are enhanced.

Figure 2-2 Firewall with Web Server Gateway



In addition to setting up a firewall, you can restrict who connects to your WebLogic Server deployment by implementing the `weblogic.security.net.ConnectionFilter` interface. This interface allows you to accept or reject a network connection based on the host name and network address of the originating machine as well as the protocol used.

Manage the WebLogic Server Development and Production Environments

For many reasons, development and production are easier when you develop on machines that closely mimic the production environment. However, security concerns suggest the following differences in the deployment and production environments:

- Do not develop on a production machine. Develop first on a development machine and then move code to the production machine when it is completed and tested. This process prevents bugs in the development environment from affecting the security of the production environment.
- Do not install the WebLogic Server sample applications on a production machine.
- The system password of a production machine should be unique within your domains and should be guarded carefully.
- Do not put the development tools on the production machine. These tools include development product components including the javac, rmic, and ejbc compilers as well as other development tools you may use. Keeping the development tools off the production machine, reduces the leverage an intruder has should they get partial access to a WebLogic Server production machine.
- Protect your source code. Getting access to your source code allows an intruder to find security holes. Always keep source code off of the production machine. Comments in JSP files that are not meant for the end user should use the JSP syntax of `<%/* ... */%>` rather than the HTML syntax of `<!-- ... -->` because the JSP comments are deleted when the JSP is compiled and therefore cannot be viewed. Also, disable the Case Sensitive Extensions field on the File tab of the Administration Console to further protect your JSP source.
- BEA does not recommend using the Servlet servlet in a production environment. You should remove all existing mappings between WebLogic servlets and the Servlet servlet from all Web applications before using the applications in a production environment.
- Do not make the File servlet the default servlet in a production environment.

Use Encryption

Encryption is the process of taking text or other data and scrambling it so that it cannot be understood. Decryption reverses the process making the text or data understandable. The decryption process always requires knowledge of a secret key or password. The secret key is a long string of bits that is required as an argument to the decryption algorithm to make it work correctly. The strength of an encryption algorithm is measured by the number of bits in its key.

There are many types of encryption and each type of encryption comes in many strengths. The biggest differences between the algorithms is how much CPU time it takes to decrypt the data and how many keys there are (symmetric key algorithms have just one key that is used to both encryption and decrypt while public key algorithms have two keys, one to encryption and one to decrypt).

Encryption is typically used in places where sensitive information is stored or communicated. These places can include but are not limited to information on network machines, on disk, in a database, in memory, and in legacy systems.

There are drawbacks to using encryption:

- Encryption and decryption are computationally expensive algorithms that take CPU time to perform.
- Encryption can make debugging harder as you cannot review encrypted data to verify that it is correct.
- The loss of a secret key can render all encrypted data useless. Even the temporary loss of a secret key (for example, all the people who know the secret key are on vacation) can render a Web site useless until the secret key can be retrieved.
- Key management is an awkward problem.Who should know the secret key, where the secret key is stored, and whether the secret key itself should be encrypted are just some of the issues that must be addressed.

The questions to ask when designing the encryption for a WebLogic Server deployment are:

- What needs to be encrypted?
- What algorithm and strength should be used to encrypt data?

- Where will the keys be stored?

Use the SSL Protocol

Data that is sent over the network (either the Internet or an intranet) can be viewed by other parties on the network. This is unavoidable because of the design of networks. To prevent sensitive data from being compromised, the data should be encrypted.

To send encrypted data over the Internet you should use the HTTPS protocol (HTTP over the Secure Sockets Layer (SSL)) rather than the HTTP protocol. To configure your Web application for the SSL protocol you must use the `user-data-constraint` tag in the `web.xml` file and set the transport-guarantee to `CONFIDENTIAL`.

The demonstration digital certificates provided with WebLogic Server are for testing only. Everyone who downloads WebLogic Server has the private keys for these digital certificates. Do not use these digital certificates in a deployed WebLogic Server.

Use the strongest encryption WebLogic Server supports: 1024-bit keys, 128-bulk data encryption on your data. The WebLogic Server version you download allows just 512-bit keys and 40-bit bulk encryption. Contact your BEA sales representative to request the stronger version.

Prevent Man-in-the-Middle Attacks

When using the SSL protocol, the data sent between the communicating parties can be vulnerable to man-in-the-middle attacks. A man-in-the-middle attack occurs when a machine inserted into the network captures, modifies, and retransmits messages to the unsuspecting parties. One way to avoid man-in-the-middle attacks is to validate that the host to which a connection is made is the intended or authorized party. An SSL client can compare the host name of the SSL server with the digital certificate of the SSL server to validate the connection. WebLogic Server provides a HostName Verifier to protect SSL connections from man-in-the-middle attacks.

By default, the HostName Verifier is enabled. However, during the implementation of WebLogic Server at your site, this functionality may have been disabled. To ensure a Hostname Verifier is being used with your WebLogic Server deployment, check that the Hostname Verification Ignore attribute on the SSL tab of the Servers node of the Administration Console is enabled.

Prevent Denial of Service Attacks

A Denial of Service attack leaves a Web site running but unusable. Hackers accomplish this attack by depleting or deleting one or more critical resources of the Web site. While a denial of service attack can happen if a hacker gets administrative privileges to your Weblogic Server, it usually occurs when an unprivileged user removes a required resource from a WebLogic Server deployment.

To perpetrate a Denial of Service attack on a WebLogic Server, an intruder bombards with many requests that are either very large in size, are slow to complete, or never complete so that the client stops sending data before completing the request. To prevent Denial of Service attacks, WebLogic Server allows you to restrict the size of a message as well as the maximum time it takes a message to arrive. You can set this information individually for each of the three protocols used by WebLogic Server: T3, HTTP and IIOP. See the online help for the Administration Console for information on setting the MaxT3MessageSize, CompleteT3MessageTimeout, MaxHTTPMessageSize, CompleteHTTPMessageTimeout, MaxIIOPMessageSize, and CompleteIIOPMessageTimeout fields. These fields have a default of 2 gigabytes for the maximum message size and 480 seconds for the complete message timeout. A value of 0 for any of the fields disables that check.

Protect User Accounts

In a dictionary attack, a hacker sets up a script to attempt logins using passwords out of a “dictionary”. WebLogic Server provides a set of configurable attributes which protect users accounts from dictionary attacks. These attributes are configurable in a number of ways (for example, you can disable all the attributes, increase the number

of invalid login attempts required before locking the account, increase the time period in which invalid login attempts have to take place before locking the account, and change the amount of time the user account is locked). It is up to site administrators to determine how these attributes should be set. Use this feature to protect user accounts for maximum security. WebLogic Server ships with the maximum security enforced.

Note: If during development you reduce security by changing these attributes, remember to reset the attributes before deploying.

For more information, see Protecting User Accounts.

Protect Application Content

By default, WebLogic Server uses a single directory, known as the web document root directory, as the location that contains static application content (HTML files and images) and dynamic application content (JSP and jHTML files). A potential vulnerability may occur if an application is allowed to create or modify files containing dynamic content within the web document root directory.

If an application is capable of modifying existing files in the web document root directory, there is the potential that the application could insert executable code in the form of JSP or jHTML tags in an existing file. If the particular file provides dynamic content, the inserted code would be executed the next time the file was served to a client.

To prevent the scenario under which this vulnerability could occur, BEA recommends the following supplemental security measures:

- WebLogic Server should only be installed on disks that support the ability to control access to specific directories and files (e.g., secure file system) to one or more specific user accounts. The use of an encrypted file system can be used to heighten the level of security at the cost of performance.
- A special operating system-specific user account (for example, `wls_owner`) should be established specifically to run WebLogic Server. This user account should be granted only the minimum operating system rights and privileges that are essential for successful execution of an application.

- The operating system-specific user account (`wls_owner`) should be the only user account that can access, create, or modify files in the web document root directory. This protection limits the ability of other applications executing on the same machine as WebLogic Server to access the web root directory.
- Directories containing JSP or jHTML files should be protected so that they can only be accessed or modified by the operating system-specific user account (`wls_owner`) under which WebLogic Serve is executed. Read-only access can be granted for administrative accounts such as `root` or `Administrator` for the purpose of archiving.
- The operating system-specific user account (`wls_owner`) that is used to create JSP and jHTML files should be granted only read and execute permissions to the JSP and jHTML files. This protective measure will prevent the operating system-specific user account from knowingly writing to these files.
- Remove any unnecessary applications from the machine(s) that are used to run WebLogic Server. If it is not possible to remove an application, review the security environment under which the application executes. You need to understand which directories applications that execute with privileges (for example, under a privileged user account or applications with the setuid privilege) can access. BEA advises that no other application use the operating system-specific user account (`wls_owner`) under which WebLogic Server runs.
- If the operating system on which WebLogic Server runs supports security auditing of file and directory access, BEA recommends using audit logging to track any denied directory or file access violations.
- Consider the use of an Intrusion Detection System (IDS) to detect attempts to modify the production environment.

Use Protected EJBs to Limit Access to Business Logic

Some parts of your Web application are more sensitive than other parts. For example, the part of your application that renders HTML is less sensitive than the part of the application that accesses a key database table. More effort should be placed on

protecting the sensitive parts of your Web application. One way to protect the sensitive parts of your Web application is to wrap them in Enterprise JavaBeans (EJBs) and use security policies to protect the EJBs. This security technique ensures that only properly authenticated and authorized users can execute the EJBs.

The following is an example of how to use EJBs and security policies to protect sensitive parts of your Web application:

- Code that allows a user to place an order on your Web site might be in a security policy protected EJB that is only accessible to registered users of your Web site.
- Code that searches and displays the catalog of products on your Web site could be in an EJB that is accessible to all users.
- Code that authorizes a return of merchandise may be in a security policy protected EJB that is only accessible to customer service personnel.

The exact choice of what to protect and to whom to grant access to specific operations must be considered on a per-application basis.

Remember your Web application is going to evolve over time. This makes hard-to-understand schemes even harder to manage. One way to help prevent future mistakes is to organize security by package. For example, you could have one package where all methods on all classes are available to registered users and another package where all methods on all classes are available only to customer service personnel. The final decision as to whom has what access is up to the EJB deployer but a security scheme based on package is an easy mechanism for the deployer to implement.

Use Security Policies

In WebLogic Server 7.0, security policies replace ACLs and answer the question "who has access" to a WebLogic resource. A WebLogic resource has no protection until you assign it a security policy. WebLogic Server provides a set of default security policies for the following WebLogic resources:

- Administrative resources such as the WebLogic Server Administration Console (referred to as the Administration Console) and the `weblogic.Admin` tool.
- Application resources

- COM resources
- EIS resources
- JDBC resources
- JNDI resources
- JMS resources
- MBean resources
- Server resources
- Web resources
- Web Services resources

Review the default security policies and create new security policies that better control access for your WebLogic resources.

Security policies for WebLogic EJBs and servlets differ from security policies for other kinds of WebLogic Server resources. EJB and servlet policies are configured in the access control properties of the EJB and Web application deployment descriptors as well as in the WebLogic Server Administration Console.

For more information, see Deployment Descriptors and Setting Protections for WebLogic Resources.

Secure Your Database

Most Web applications use a database to store their data. Common databases used with WebLogic Server are Oracle, Microsoft's SQL Server, and Informix. The databases frequently hold the Web application's sensitive data including customer lists, customer contact information, credit card information, and other proprietary data. When creating your Web application you must consider what data is going to be in the database and how secure you need to make that data. You also need to understand the security mechanisms provided by the manufacturer of the database and decide whether they are sufficient for your needs. If the mechanisms are not sufficient, you can use other security techniques to improve the security of the database. One common technique is

to encrypt sensitive data before writing it to the database. For example, you might leave all customer data in the database in plain text except for the credit card information which is encrypted.

Use Auditing

Auditing is the process of recording key security events in your WebLogic Server environment. The audit record is usually kept separate from the WebLogic Server log file. Reviewing the auditing records can help you determine whether there has been a security breach or an attempted breach. Noting things such as repeated failed logon attempts or a surprising pattern of security events may be the key to preventing serious problems. By default, the WebLogic Auditing provider is enabled and audit events generated by the provider are saved in *WL_HOME*\mydomain\DefaultAuditRecorder.log.