BEA WebLogic Server®

Securing a Production Environment

Version 10.0
Revised: March 30, 2007

Contents

CHAPTER 1

Introduction and Roadmap

The following sections describe the contents and organization of this guide—*Securing a Production Environment*.

Document Scope and Audience

This guide describes how to secure a WebLogic Server® production environment.

It is intended for the following audiences:

- Application Architects—Architects who, in addition to setting security goals and designing the overall security architecture for their organizations, evaluate WebLogic Server security features and determine how to best implement them. Application Architects have in-depth knowledge of Java programming, Java security, and network security, as well as knowledge of security systems and leading-edge, security technologies and tools.
- Security Developers—Developers who focus on defining the system architecture and infrastructure for security products that integrate into WebLogic Server and on developing custom security providers for use with WebLogic Server. Security Developers have a solid understanding of security concepts, including authentication, authorization, auditing

(AAA), in-depth knowledge of Java (including Java Management eXtensions (JMX), and working knowledge of WebLogic Server and security provider functionality.

- Application Developers—Java programmers who develop and add security to Web applications and Enterprise JavaBeans (EJBs), and work with other engineering, quality assurance (QA), and database teams to implement security features. Application Developers have in-depth/working knowledge of Java (including Java Platform, Enterprise Edition (Java EE) Version 5 components such as servlets/JSPs and JSEE) and Java security.
- Server Administrators—Administrators who work closely with Application Architects to design a security scheme for the server and the applications running on the server, to identify potential security risks, and to propose configurations that prevent security problems. Related responsibilities may include maintaining critical production systems, configuring and managing security realms, implementing authentication and authorization schemes for server and application resources, upgrading security features, and maintaining security provider databases. Server Administrators have in-depth knowledge of the Java security architecture, including Web services, Web application and EJB security, Public Key security, SSL, and Security Assertion Markup Language (SAML).
- Application Administrators—Administrators who work with Server Administrators to implement and maintain security configurations and authentication and authorization schemes, and to set up and maintain access to deployed application resources in defined security realms. Application Administrators have general knowledge of security concepts and the Java Security architecture. They understand Java, XML, deployment descriptors, and can identify security events in server and audit logs.

Guide to This Document

This document is organized as follows:

- This chapter, Chapter 1, "Introduction and Roadmap," introduces the organization of this guide.
- Chapter 2, "Determining Your Security Needs" explains how to determine the security needs for your particular environment and describes basic measures to ensure that those needs are being met.
- Chapter 3, "Ensuring the Security of Your Production Environment" highlights essential security measures to consider before you deploy WebLogic Server into a production environment and describes how to use different security settings to secure a production environment.

Related Information

The following BEA WebLogic Server documents contain information that is relevant to the WebLogic Security Service:

- *Securing WebLogic Server*—explains how to configure security for WebLogic Server and how to use Compatibility security.
- *Developing Security Providers for WebLogic Server*—explains how vendors and application developers can develop custom security providers that can be used with WebLogic Server.
- *Understanding WebLogic Security*—provides an overview of the features, architecture, and functionality of the WebLogic Security Service. It is the starting point for understanding the WebLogic Security Service.
- *Securing WebLogic Resources Using Roles and Policies*—describes how to secure WebLogic resources. It primarily focuses on securing URL (Web) and Enterprise JavaBean (EJB) resources.
- *Upgrading WebLogic Application Environments*—provides procedures and other information you need to upgrade 6.x and earlier versions of WebLogic Server to WebLogic Server 9.x. It also provides information about moving applications from a 6.x or earlier version of WebLogic Server to 9.x. For specific information on upgrading WebLogic Server security, see *Upgrading a Security Provider* in the *Upgrading WebLogic Application Environments*.
- *Javadocs for WebLogic Classes*—is reference documentation for the WebLogic security packages that are provided with and supported by this release of WebLogic Server.

Security Samples and Tutorials

BEA Systems provides code samples for Java Authentication and Authorization Service and for Outbound and Two-way SSL for Security developers. The examples and tutorials illustrate WebLogic Server Security in action, and provide practical instructions on how to perform key Security development tasks.

BEA recommends that you run some or all of the Security examples before developing your own Security configurations.

Avitek Medical Records Application (MedRec) and Tutorials

MedRec is an end-to-end sample Java EE application shipped with WebLogic Server that simulates an independent, centralized medical record management system. The MedRec application provides a framework for patients, doctors, and administrators to manage patient data using a variety of different clients.

MedRec demonstrates WebLogic Server and Java EE features, and highlights BEA-recommended best practices. MedRec is included in the WebLogic Server distribution, and can be accessed from the Start menu on Windows machines. For Linux and other platforms, you can start MedRec from the *WL_HOME*\samples\domains\medrec directory, where *WL_HOME* is the top-level installation directory for WebLogic Platform.

MedRec includes a service tier consisting primarily of Enterprise Java Beans (EJBs) that work together to process requests from Web applications, Web services, workflow applications, and future client applications. The application includes message-driven, stateless session, stateful session, and entity EJBs.

Security Examples in the WebLogic Server Distribution

WebLogic Server optionally installs API code examples in *WL_HOME*\samples\server\examples\src\examples, where *WL_HOME* is the top-level directory of your WebLogic Server installation. You can start the examples server, and obtain information about the samples and how to run them from the WebLogic Server Start menu.

Additional Security Examples Available for Download

Additional API examples for download at http://dev.bea.com/code/index.jsp. These examples are distributed as ZIP files that you can unzip into an existing WebLogic Server samples directory structure.

You build and run the downloadable examples in the same manner as you would an installed WebLogic Server example. See the download pages of individual examples for more information at http://dev.bea.com/code/index.jsp.

CHAPTER 2

Determining Your Security Needs

Before you deploy WebLogic Server and your Java EE applications into a production environment, determine your security needs and make sure that you take the appropriate security measures, as described in the following sections:

Understand Your Environment

To better understand your security needs, ask yourself the following questions:

- Which resources am I protecting?

  Many resources in the production environment can be protected, including information in databases accessed by WebLogic Server and the availability, performance, applications, 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 resources?

  For most Web sites, resources must be protected from everyone on the Internet. But should the Web site be protected from the employees on the intranet in your enterprise? Should your employees have access to all resources within the WebLogic Server environment?

Should the system administrators have access to all WebLogic 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 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 protect it properly.

Hire Security Consultants or Use Diagnostic Software

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. BEA partners offer services and products that can help you to secure a WebLogic Server production environment. See the BEA Partner's Page at http://www.bea.com/partners.

Read Security Publications

Read about security issues:

- For the latest information about securing Web servers, BEA recommends the "Security Practices & Evaluations" information available from the CERT™ Coordination Center operated by Carnegie Mellon University.
- For BEA security advisories, refer to the BEA Advisories & Notifications page on the dev2dev Web site at http://dev2dev.bea.com/advisories. Here, you can download security-related patches and register to receive notifications of newly available security advisories.

  Report possible security issues in BEA products to `secalert@bea.com`.

Install WebLogic Server in a Secure Manner

Currently, the WebLogic Server installation includes the entire JDK and some additional WebLogic Server development utilities (for example, beasvc). These development programs

could be a security vulnerability. The following are recommendations for making a WebLogic Server installation more secure:

- Do not install the WebLogic Server sample applications. When installing WebLogic Server, select the Custom option and unclick the Samples option.
- Minimize the WebLogic Server installation by doing the following:

  Note: There is always a potential of making mistakes when deleting executables, files, and directories from the WebLogic Server installation. Therefore, BEA recommends testing your changes in a secure, development environment before implementing them in a production environment.

  – Run with the JRE instead of the Java SDK. The Javasoft SDK offers a JRE download and installation. When installing WebLogic Server, use the Configuration Wizard and select the JRE option. This option eliminates the Java compiler and other development tools.

  – When using BEA JRockit®, delete the software components of the Java SDK that are not in the BEA JRockit JRE.

  – Delete development tools such as the Configuration Wizard, WebLogic Builder, and the jCOM tools if you don't plan to use them in production.

  – Delete the Pointbase database which is included for evaluation purposes and it is not supported in the production environments.

CHAPTER 3

Ensuring the Security of Your Production Environment

BEA recommends that you implement the following actions to ensure the security of your production environment:

An Important Note Regarding Null Cipher Use in SSL

SSL clients start the SSL handshake by connecting to the server. As part of the connection, the client sends the server a list of the cipher suites it supports.

A cipher suite is an SSL encryption method that includes the key exchange algorithm, the symmetric encryption algorithm, and the secure hash algorithm. A cipher suite is used to protect the integrity of a communication. For example, the cipher suite called RSA_WITH_RC4_128_MD5 uses RSA for key exchange, RC4 with a 128-bit key for bulk encryption, and MD5 for message digest.

The server selects a mutually-supported cipher suite from the list supplied by the client for the client and server to use for this session.

However, a misconfigured client might specify a set of cipher suites that contain only null ciphers.

A null cipher passes data on the wire in clear-text. (An example of a cipher suite with a null cipher is TLS_RSA_WITH_NULL_MD5.) Using a null cipher makes it possible to see the SSL messages by using a network packet sniffer. In essence, SSL is used but does not provide any security.

The server selects the null cipher only when it is the only cipher suite they have in common.

If the server selects a null cipher from the client's cipher suite list the log contains the following message: "SSL has established a session that uses a Null cipher."

This message is output only when the server has selected a null cipher from the client's list.

Note: If there is any potential whatsoever that an SSL client might use a null cipher to inappropriately connect to the server, you should check the log file for this message. It is recommended that new client configurations be given extra attention with respect to the use of a null cipher to ensure that they are properly configured.

It is unlikely that an existing client configuration would suddenly start using null ciphers if it had not been doing so previously. However, an existing client configuration that is unknowingly misconfigured could be using null ciphers.

Other SSL errors unrelated to null ciphers are possible as well, and each will display an appropriate error message in the log.

For information on configuring SSL, see Configuring SSL in *Securing WebLogic Server*. For information on viewing log files, see View and configure logs in the online help.

Securing the WebLogic Server Host

A WebLogic Server production environment is only as secure as the security of the machine on which it is running. 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 a WebLogic Server host in a production environment. Also check with the manufacturer of the machine and operating system for recommended security measures.

Note: To enable a WebLogic Server instance to use a FIPS-compliant (FIPS 140-2) crypto module in the server's SSL implementation, make sure that the server start script (for example, `startWebLogic.cmd/sh`) contains the following:

- `jsafeFIPS.jar` is added to the `PRE_CLASSPATH` variable.

- The command line argument `-Dweblogic.security.SSL.nojce=true` is specified.

FIPS 140-2 is a standard that describes U.S. Federal government requirements for sensitive, but unclassified use.

Table 3-1 Securing the WebLogic Server Host

Security Action	Description
Physically secure the hardware.	Keep your hardware in a secured area to prevent unauthorized operating system users from tampering with the deployment machine or its network connections.
Secure networking services that the operating system provides.	Have an expert review network services such as e-mail programs or directory services to ensure that a malicious attacker cannot access the operating system or system-level commands. The way you do this depends on the operating system you use. Sharing a file system with other machines in the enterprise network imposes risks of a remote attack on the file system. Be certain that the remote machines and the network are secure before sharing the file systems from the machine that hosts WebLogic Server.
Use a file system that can prevent unauthorized access.	Make sure that the file system on each WebLogic Server host can prevent unauthorized access to protected resources. For example, on a Windows computer, use only NTFS.

Table 3-1 Securing the WebLogic Server Host

Security Action	Description
Set file access permissions for data stored on disk.	Set operating system file access permissions to restrict access to data stored on disk. This data includes, but is not limited to, the following: • The security LDAP database, which by default is in `\domains\domain-name\servers\server-name\data\ldap\ldapfiles`. • The directory and filename location of a private keystore, as described in Securing WebLogic Server. • The directory and filename location of a Root Certificate Authority (CA) keystore, as described in Securing WebLogic Server. For example, operating systems such as Unix and Linux provide utilities such as umask and chmod to set the file access permissions. At a minimum, consider using "umask 066", which denies read and write permission to Group and Others.

Table 3-1 Securing the WebLogic Server Host

Security Action	Description
Set file access permissions for data stored in persistent store.	• Set operating system file access permissions to restrict access to data stored in the persistent store. Overview of the Persistent Store defines many of the WebLogic services and subsystems that can create connections to the persistent store. This list includes: – JMS Messages. JMS Messages are described in Understanding the Messaging Models in *Programming WebLogic JMS*. – Web Services. Web Services are described in Using Reliable SOAP Messaging in *WebLogic Web Services: Advanced Programming*. – EJB Timer services. EJB Timer services are described in Understanding Enterprise JavaBeans in *Programming WebLogic Enterprise JavaBeans*. – Store-and-Forward services. Store-and-Forward services are described in Understanding the Store-and-Forward Service in *Configuring and Managing WebLogic Store-and-Forward*. – JTA Transaction Log (TLOG). The JTA Transaction Log is described in Managing Transactions in *Programming WebLogic JTA*. The default persistent store maintains its data in a `data\store\default` directory inside the `servername` subdirectory of a domain's root directory.

Table 3-1 Securing the WebLogic Server Host

Security Action	Description
Limit the number of user accounts on the host machine.	Avoid creating more user accounts than you need on WebLogic Server hosts, and limit the file access privileges granted to each account. Ideally, the host machine would have two user accounts with system administrator privileges on operating systems that allow more than one system administrator user and another user with sufficient privileges to run WebLogic Server. Having two system administrator users provides a back up at all times. The WebLogic Server user should be a restricted user, not a system administrator user. One of the system administrator users can always create a new WebLogic Server user if needed. Review active user accounts regularly and when personnel leave. *Background Information*: Some WebLogic Server configuration data and some URL (Web) resources, including Java Server Pages (JSPs) and HTML pages, are stored in clear text on the file system. A sophisticated user or intruder with read access to files and directories might be able to defeat any security mechanisms you establish with WebLogic Server authentication and authorization schemes.
For your system administrator user accounts, choose names that are not obvious.	For additional security, avoid choosing an obvious name such as "system", "admin", or "administrator" for your system administrator user accounts.
Safeguard passwords.	The passwords for user accounts on production machines should be difficult to guess and should be guarded carefully. Set a policy to expire passwords periodically. Never code passwords in client applications.

Table 3-1 Securing the WebLogic Server Host

Security Action	Description
On each host computer, give only one user account access to WebLogic resources in addition to the two system administrator users who also have access privileges.	On each WebLogic Server host computer, use the operating system to establish a special user account (for example, `wls_owner`) specifically to run WebLogic Server. Grant to this operating-system (OS) user account the following privileges: • Access privileges only to the BEA Home directory, the WebLogic Server product directory tree, and your domain directories. The **BEA Home directory** is a repository for common files that are used by multiple BEA products installed on the same machine. The WebLogic Server **product installation directory** contains all the WebLogic Server software components that you choose to install on your system, including program files. A **domain directory** contains the configuration files, security files, log files, Java EE applications, and other Java EE resources for a single WebLogic domain. If you install multiple domains on a WebLogic Server host computer, each domain directory must be protected. By default, the BEA installation program places all BEA files and your domain directories in a single directory tree, whose top directory is named `bea`. All WebLogic Server files are a subdirectory of this directory tree (`bea\wlserver_10.0`), and your domain files are in other subdirectories (`bea\user_projects\domains\`*`domain1`*, `bea\user_projects\domains\`*`domain2`*, ...). You can, however, locate the WebLogic Server product installation directory and your domain directories outside the BEA Home directory. For more information, refer to "Selecting Directories for the WebLogic Platform Installation" in the *Installation Guide*. • No other OS user should have read, write, or execute access to BEA files and your domain files. (The system administrator users have access privileges by default.) This protection limits the ability of other applications executing on the same machine as WebLogic Server to access BEA files and your domain files. Without this protection, some other application could gain write access and insert malicious, executable code in JSPs and other files that provide dynamic content. The code would be executed the next time the file was served to a client.

Table 3-1 Securing the WebLogic Server Host

Security Action	Description
On each host computer, give only one user account access to WebLogic resources (continued...).	Knowledgeable operating system users may be able to bypass WebLogic Server security if they are given write access, and in some cases read access, to the following files: • WebLogic Server Installation • JDK files (typically in the WebLogic Server installation, but can be configured to be separate) • Domain directory • JMS SAF files • File backed HTTP sessions Everything that uses the persistent store, such as JMS SAF files, has sensitive data that should be protected from read access as well as from write access. The persistent store supports persistence to a file-based store or to a JDBC-enabled database. If you use the file store to store files on WebLogic Server, the applications can be stored anywhere. You must remember the locations of all of the files in order to protect them from read and write access. If you use the JDBC store to store applications, make sure to properly secure the database by protecting it from read and write access. For more information on using the persistent store, see "Using the WebLogic Persistent Store" in *Configuring WebLogic Server Environments*.

Table 3-1 Securing the WebLogic Server Host

Security Action	Description
To bind to protected ports on UNIX, configure WebLogic Server to switch user IDs or use Network Address Translation (NAT) software.	On UNIX systems, only processes that run under a privileged user account (in most cases, root) can bind to ports lower than 1024. UNIX systems allow only one system administrator (root) user. However, long-running processes like WebLogic Server should not run under these privileged accounts. Instead, you can do either of the following: • For each WebLogic Server instance that needs access to privileged ports, configure the server to start under the privileged user account, bind to privileged ports, and change its user ID to a non-privileged account. If you use Node Manager to start the server instance, configure Node Manager to accept requests only on a secure port and only from a single, known host. Note that Node Manager needs to be started under a privileged user account. See Bind to protected ports on UNIX in the *Administration Console Online Help*. • Start WebLogic Server instances from a non-privileged account and configure your firewall to use Network Address Translation (NAT) software to map protected ports to unprotected ones. BEA does not provide NAT software.
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 development and sample software on a production machine.	Do not install development tools on production machines. Keeping development tools off the production machine reduces the leverage intruders have should they get partial access to a WebLogic Server production machine. Do not install the WebLogic Server sample applications on a production machine. When the BEA installation program asks whether you want a Typical Installation or Custom Installation: 1. Choose Custom Installation. Then click Next. 2. On the Choose Components page, remove the check mark from the Server Examples check box. Then click Next. Complete the remaining pages of the BEA installation program.

Table 3-1 Securing the WebLogic Server Host

Security Action	Description
Enable security auditing.	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. Administrators should ensure that sufficient disk space is available for the audit log.
Consider using additional software to secure your operating system.	Most operating systems can run additional software to secure a production environment. For example, an Intrusion Detection System (IDS) can detect attempts to modify the production environment. Refer to the vendor of your operating system for information about available software.
Apply operation-system service packs and security patches.	Refer to the vendor of your operating system for a list of recommended service packs and security-related patches.
Apply the latest BEA service packs and implement the latest security advisories.	If you are responsible for security related issues at your site, register on the BEA Advisories & Notifications page at http://dev2dev.bea.com/advisoriesnotifications to receive notifications of newly available security advisories. Remedies recommended in the security advisories are posted on the Advisories & Notifications page. In addition, you are advised to apply each service pack as it is 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. You can download service packs from http://commerce.bea.com/downloads. Report possible security issues in BEA products to `secalert@bea.com`.
Do not run WebLogic Server in Development mode in a production environment.	Production mode sets the server to run with settings that are more secure and appropriate for a production environment.
Do not run web servers (Apache/Sun/IIS) as root.	If the web server is running as root, any script that the root executes must be protected from modification by non-root users.

Securing Network Connections

When designing network connections, you balance the need for a security solution that is easy to manage with the need to protect strategic WebLogic resources. The following table describes options for securing your network connections.

Table 3-2 Securing Network Connections

Security Action	Description
Use hardware and software to create firewalls.	A firewall limits traffic between two networks. Firewalls can be a combination of software and hardware, including routers and dedicated gateway machines. They employ filters that allow or disallow traffic to pass based on the protocol, the service requested, routing information, packet content, and the origin and destination hosts or networks. They can also limit access to authenticated users only. The WebLogic Security Service supports the use of third-party Identity Assertion providers, which perform perimeter-based authentication (Web server, firewall, VPN) and handle multiple security token types/protocols (SOAP, IIOP-CSIv2). For more information, refer to Perimeter Authentication in *Understanding WebLogic Security*. For more information about using firewalls with WebLogic Server, refer to Security Options for Cluster Architectures in *Using WebLogic Server Clusters*.
Use WebLogic Server connection filters.	Instead of, or in addition to, using hardware and third-party software to create firewalls, consider using WebLogic Server connection filters to limit network traffic based on protocols, IP addresses, and DNS node names. Connection filters are most appropriate when the machines in a WebLogic Server domain can access each other without going through a firewall. For example, you might use a firewall to limit traffic from outside the network, and then use WebLogic Server connection filters to limit traffic behind the firewall. See "Using Connection Filters" in *Securing WebLogic Server*.

Table 3-2 Securing Network Connections

Security Action	Description
Use a domain-wide Administration Port for administrative traffic.	An Administration Port limits all administrative traffic between server instances in a WebLogic Server domain to a single port. When the server is run without an Adminstrative Port, an application can inadvertently transmit confidential server configuration on the wire in clear-text. Running the server with an Administration Port significantly reduces the chances of this happening. Furthermore, having an Administrative Port configured is helpful should a denial-of-service attack occur because the resources for handling requests for, and the limitations on Administration Port requests are separate from those of the rest of the server. When used in conjunction with a connection filter, you can specify that a WebLogic Server instance accepts administrative requests only from a known set of machines or subnets and only on a single port. Enabling the Administration Port requires clients to interact with the Administration Console using SSL which protects sensitive data from being sniffed on the wire by an attacker and protects against some cross site scripting attacks. See "Configure the domain-wide administration port" and "Enable Configuration Auditing" in the *Administration Console Online Help*.

Table 3-2 Securing Network Connections

Security Action	Description
Secure the embedded LDAP port.	To protect the embedded LDAP port against brute force attacks, close off the embedded LDAP listen port using a connection filter in a single server configuration. While this does not protect the embedded LDAP port in a multiple server configuration, the default connection filter implementation supports filtering based on the source IP address which should be used to allow access only from servers that are part of the domain. As a result, only the machines in the domain can access the LDAP port. For more information on using connection filters, see "Using Network Connection Filters" in *Programming WebLogic Security*.
Do not enable remote access to the JVM platform MBean server.	As of JDK 1.5, the JDK provides an MBean server (the platform MBean server) and a set of MBeans that contain monitoring information about the JVM. You can configure the WebLogic Server Runtime MBean Server to run as the platform MBean server, which enables JMX clients to access the JVM MBeans and WebLogic Server MBeans from a single MBean server connection. Remote access to the platform MBean server can be secured only by standard JDK 1.5 security features (see http://java.sun.com/j2se/1.5.0/docs/guide/management/agent.html#remote). If you have configured the WebLogic Server Runtime MBean Server to be the platform MBean server, enabling remote access to the platform MBean server creates an access path to WebLogic Server MBeans that is not secured through the WebLogic Server security framework. If it is essential that remote JMX clients have access to the JVM MBeans, BEA recommends that you access them through the WebLogic Server Runtime MBean Server. See Registering MBeans in the JVM Platform MBean Server in *Developing Manageable Applications with JMX*.

Securing Your Database

Most Web applications use a database to store their data. Common databases used with WebLogic Server are Oracle, Microsoft 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, such as encrypting sensitive data before writing it to the database. For example, leave all customer data in the database in plain text except for the encrypted credit card information.

Securing the WebLogic Security Service

The WebLogic Security Service provides a powerful and flexible set of software tools for securing the subsystems and applications that run on a server instance. The following table

provides a checklist of essential features that BEA recommends you use to secure your production environment.

Table 3-3 Securing the WebLogic Security Service

Security Action	Description
Deploy production-ready security providers to the security realm.	The WebLogic Security Service uses a pluggable architecture in which you can deploy multiple security providers, each of which handles a specific aspect of security. By default WebLogic Server includes its own security providers that provide a complete security solution. If you have purchased or written your own security providers: • Make sure that you have deployed and configured them properly. You can verify which security providers are currently deployed in the Administration Console. In the left pane, select Console, select Security Realms, then click on the name of the realm and select the Providers tab. • Make sure that the realm in which you deployed your security providers is the default (active) realm. For instructions on how to set the default security realm in the Administration Console, see Change the default security realm in the *Administration Console Online Help*. • Refer to Customizing the Default Security Configuration in *Securing WebLogic Server*.
Use SSL, but do not use the demonstration digital certificates in a production environment.	To prevent sensitive data from being compromised, secure data transfers by using HTTPS. WebLogic Server includes a set of demonstration private keys, digital certificates, and trusted certificate authorities that are for development only. Everyone who downloads WebLogic Server has the private keys for these digital certificates. Do not use the demonstration identity and trust. Refer to Configuring Keystores in the *Administration Console Online Help* and Configuring SSL in *Securing WebLogic Server*.

Table 3-3 Securing the WebLogic Security Service

Security Action	Description
Make sure that WebLogic Server enforces security constraints on digital certificates.	When communicating by SSL, by default WebLogic Server rejects any digital certificates in a certificate chain that do not have the Basic Constraint extension defined by the Certificate Authority. This level of enforcement protects your Web site from the spoofing of digital certificates. Make sure that no server startup command includes the following option, which disables this enforcement: `-Dweblogic.security.SSL.enforceConstraints=false` For more information about this option, see "SSL Certificate Validation" in *Securing WebLogic Server*. In your development environment, you might have disabled the enforcement of security constraints to work around incompatibilities with demonstration digital certificates that WebLogic Server provided in releases prior to 7.0 Service Pack 2. Make sure you enable this feature in your production environment.
Verify that host name verification is enabled to avoid man-in-the-middle attacks.	By default, the WebLogic SSL implementation validates that the host to which a connection is made is the intended or authorized party. However, during the implementation of WebLogic Server at your site, you might have disabled host name verification. To enable host name verification, see Configure a custom host name verifier in the *Administration Console Online Help*. Refer to Using Host Name Verification in *Securing WebLogic Server*. *Background Information:* 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. The WebLogic Server HostName Verifier protects SSL connections from man-in-the-middle attacks.

Table 3-3 Securing the WebLogic Security Service

Security Action	Description
Restrict the size and the time limit of requests on external channels to prevent denial-of-service attacks.	To prevent some denial-of-service attacks, WebLogic Server can restrict the size of a message as well as the maximum time it takes a message to arrive. The default setting for message size is 10 megabytes and 480 seconds for the complete message timeout. BEA recommends that you: • Set the size limit of requests on internal channels so that a Managed Server can to accept messages from the Administration Server. • Restrict the size and time limits of requests on external channels. • Configure internal channels so that they are only accessible internally and not externally. To configure these settings for the HTTP, T3, and IIOP protocols refer to the following tasks in the *Administration Console Online Help*: • Configure HTTP protocol • Configure T3 Protocol • Enable and Configure IIOP *Background Information:* A denial-of-service attack leaves a Web site running but unusable. Hackers deplete or delete one or more critical resources of the Web site. To perpetrate a denial-of-service attack on a WebLogic Server instance, an intruder bombards the server with many requests that are very large, are slow to complete, or never complete so that the client stops sending data before completing the request.
Set the number of sockets allowed to a server to prevent denial-of-service attacks.	To prevent some denial-of-service attacks, limit the number of sockets allowed for a server so that there are fewer than the number of sockets allowed to the entire process. This ensures that the number of file descriptors allowed by the operating system limits is not exceeded. Even after the server's limit is exceeded, administrators can access the server through the Administration Port. You can configure this setting using the MaxOpenSockCount flag. In the *Administration Console Online Help*, see Servers: Configuration: Tuning.

Table 3-3 Securing the WebLogic Security Service

Security Action	Description
Configure WebLogic Server to avoid overload conditions.	Configure WebLogic Server to avoid overload conditions in order to allow WebLogic Server sufficient processing power so that an administrator can connect to it and attempt to correct the problem in case the server comes under heavy load. Because communication over administration channels is not prevented when the system is overloaded, administrators can always connect regardless of any current overload condition. In case of heavy load, the administrator should bring the server into the admin state, locate the offending user, and then prevent that user from overloading the server with requests. To configure WebLogic Server to avoid overload conditions, set the shared capacity attribute in the overload protection MBean. The setting you choose for this attribute is the threshold after which no more non-administrator requests are accepted by WebLogic Server. For more information on overload conditions, see Avoiding and Managing Overload in *Configuring WebLogic Server Environments*.
Configure user lockouts and login time limits to prevent attacks on user accounts.	By default, the WebLogic Security Service provides security against dictionary and brute force attacks of user accounts. If during development you changed the settings for the number of invalid login attempts required before locking the account, the time period in which invalid login attempts have to take place before locking the account, or the amount of time the user account is locked, review the settings and verify that they are adequate for your production environment. See Protect user accounts in the *Administration Console Online Help*. *Background Information:* In a dictionary/brute force attack, a hacker sets up a script to attempt logins using passwords out of a "dictionary." The WebLogic Server user lockout and login settings can protect user accounts from dictionary/brute force attacks.
If you use multiple Authentication providers, be sure to set the JAAS control flag correctly.	If a security realm has multiple Authentication providers configured, configure the order and precedence of **each** provider by setting the JAAS control flags. See Set the JAAS control flag in the *Administration Console Online Help*.

Table 3-3 Securing the WebLogic Security Service

Security Action	Description
Enable security auditing.	Auditing is the process of recording key security events in your WebLogic Server environment. When the Auditing provider that the WebLogic Security Service provides is enabled, it logs events in *DomainName*\DefaultAuditRecorder.log You enable an Auditing provider in the Administration Console on the Security Realms → *RealmName* → Providers → Auditing page. See Configure Auditing Providers in the *Administration Console Online Help*. **Note:** Using an Auditing provider might adversely affect the performance of WebLogic Server even if only a few events are logged. Review the auditing records periodically to detect security breaches and attempted breaches. Noting repeated failed logon attempts or a surprising pattern of security events can prevent serious problems. If you develop your own custom Auditing provider and would like more information on posting audit events from a provider's Mbean, refer to the Best Practice: Posting Audit Events from a Provider's MBean section in *Developing Security Providers for WebLogic Server*.
Ensure that you have correctly assigned users and groups to the default WebLogic Server security roles.	By default, all WebLogic resources are protected by security policies that are based on a default set of security roles. Make sure you have assigned the desired set of users and groups to these default security roles. Refer to Users, Groups, And Security Roles in *Securing WebLogic Resources Using Roles and Policies*.
Create no fewer than two user accounts with system administrator privileges.	One of the system administrator users should be created when the domain is created. Create other user(s) and assign them the Admin security role. When creating system administrator users give them unique names that cannot be easily guessed. Having at least two system administrator user accounts helps to ensure that one user maintains account access in case another user becomes locked out by a dictionary/brute force attack.

Securing Applications

Although much of the responsibility for securing the WebLogic resources in a WebLogic Server domain fall within the scope of the server, some security responsibilities lie within the scope of individual applications. For some security options, the WebLogic Security Service enables you to determine whether the server or individual applications are responsible. For each application that you deploy in a production environment, review the items in the following table to verify that you have secured its resources.

Table 3-4 Securing Applications

Security Action	Description
Determine which deployment model secures your Web applications and EJBs.	By default, each Web application and EJB uses deployment descriptors (XML files) to declare its secured resources and the security roles that can access the secured resources. Instead of declaring security in Web application and EJB deployment descriptors, you can use the Administration Console to set security policies that secure access to Web applications and EJBs. This technique provides a single, centralized location from which to manage security for all Web applications and EJBs. You can combine these two techniques and configure WebLogic Server to copy security configurations from existing deployment descriptors upon the initial deployment of a URL (Web) or EJB resource. Once these security configurations are copied, the Administration Console can be used for subsequent updates. Refer to Options for Securing Web Application and EJB Resources in *Securing WebLogic Resources Using Roles and Policies*.
Use JSP comment tags instead of HTML comment tags.	Comments in JSP files that might contain sensitive data and or other comments that are not intended for the end user should use the JSP syntax of `<%/* ... */%>` instead of the HTML syntax `<!-- ... -->`. The JSP comments, unlike the HTML comments, are deleted when the JSP is compiled and therefore cannot be viewed in the browser.
Do not install uncompiled JSPs and other source code on the production machine.	Always keep source code off of the production machine. Getting access to your source code allows an intruder to find security holes. Consider precompiling JSPs and installing only the compiled JSPs on the production machine. For information about precompiling JSPs, refer to "Precompiling JSPs" in *Developing Web Applications, Servlets, and JSPs in WebLogic Server*.

Table 3-4 Securing Applications

Security Action	Description
Configure your applications to use SSL.	Set the `transport-guarantee` to `CONFIDENTIAL` in the `user-data-constraint` element of the `web.xml` file whenever appropriate. Refer to security-constraint in *Developing Web Applications, Servlets, and JSPs in WebLogic Server*.
Do not use the `Servlet` servlet.	BEA does not recommend using the `Servlet` servlet in a production environment. Instead, map servlets to URIs explicitly. Remove all existing mappings between WebLogic servlets and the `Servlet` servlet from all Web applications before using the applications in a production environment. For information on mapping servlets, refer to Configuring Servlets in *Developing Web Applications, Servlets, and JSPs in WebLogic Server*.
Do not leave `FileServlet` as the default servlet in a production environment.	BEA does not recommend using the `FileServlet` servlet as the default servlet a production environment. For information on setting up a default servlet, refer to Setting Up a Default Servlet in *Developing Web Applications, Servlets, and JSPs in WebLogic Server*.
Verify all WebLogic security policies.	In WebLogic Server 7.0, security policies replace ACLs and answer the question "who has access" to a WebLogic resource. Make sure that you have not removed security policies from WebLogic resources, and make sure that your security role assignments provide users the kind of access that you intend. Refer to Securing WebLogic Resources Using Roles and Policies.
Examine applications for security.	There are instances where an application can lead to a security vulnerability. Many of these instances are defined by third-party organizations such as Open Web Application Security Project (see http://www.owasp.org/documentation/topten for a list of common problems). Of particular concern is code that uses Java native interface (JNI) because Java positions native code outside of the scope of Java security. If Java native code behaves errantly, it is only constrained by the operating system. That is, the Java native code can do anything WebLogic Server itself can do. This potential vulnerability is further complicated by the fact that buffer overflow errors are common in native code and can be used to run arbitrary code.

Table 3-4 Securing Applications

Security Action	Description
If your applications contain untrusted code, enable the Java security manager.	The Java security manager defines and enforces permissions for classes that run within a JVM. In many cases, where the threat model does not include malicious code being run in the JVM, the Java security manager is unnecessary. However, when third parties use WebLogic Server and untrusted classes are being run, the Java security manager may be useful. To enable the Java security manager for a server instance, use the following Java options when starting the server: `-Djava.security.manager` `-Djava.security.policy[=]=`*`filename`* Refer to "Using the Java Security Manager to Protect WebLogic Resources" in *Programming WebLogic Security*.
Replace HTML special characters when servlets or JSPs return user-supplied data.	The ability to return user-supplied data can present a security vulnerability called **cross-site scripting**, which can be exploited to steal a user's security authorization. For a detailed description of cross-site scripting, refer to "Understanding Malicious Content Mitigation for Web Developers" (a CERT security advisory) at http://www.cert.org/tech_tips/malicious_code_mitigation.html. To remove the security vulnerability, before you return data that a user has supplied, scan the data for HTML special characters. If you find any such characters, replace them with their HTML entity or character reference. Replacing the characters prevents the browser from executing the user-supplied data as HTML. See "Securing User-Supplied Data in JSPs" in *Developing Web Applications, Servlets, and JSPs in WebLogic Server* and "Securing Client Input in Servlets" in *Developing Web Applications, Servlets, and JSPs in WebLogic Server*.
Ensure security checks on performed on JMS resources.	Set the `weblogic.jms.securityCheckInterval` attribute to zero to ensure that an authorization check is performed for every Send, Receive, and `getEnumeration` action on a JMS resource.