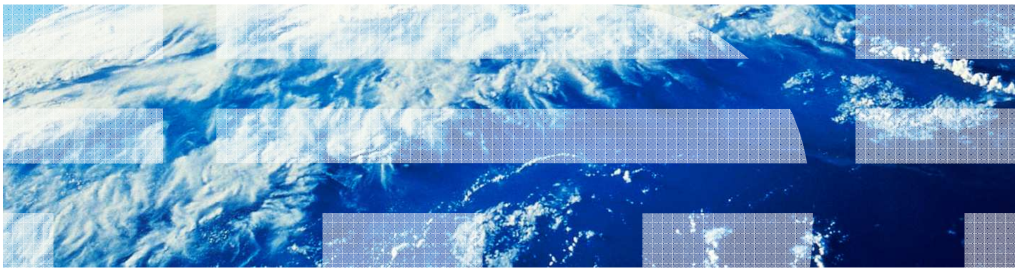
WebSphere security configuration problems



WebSphere software

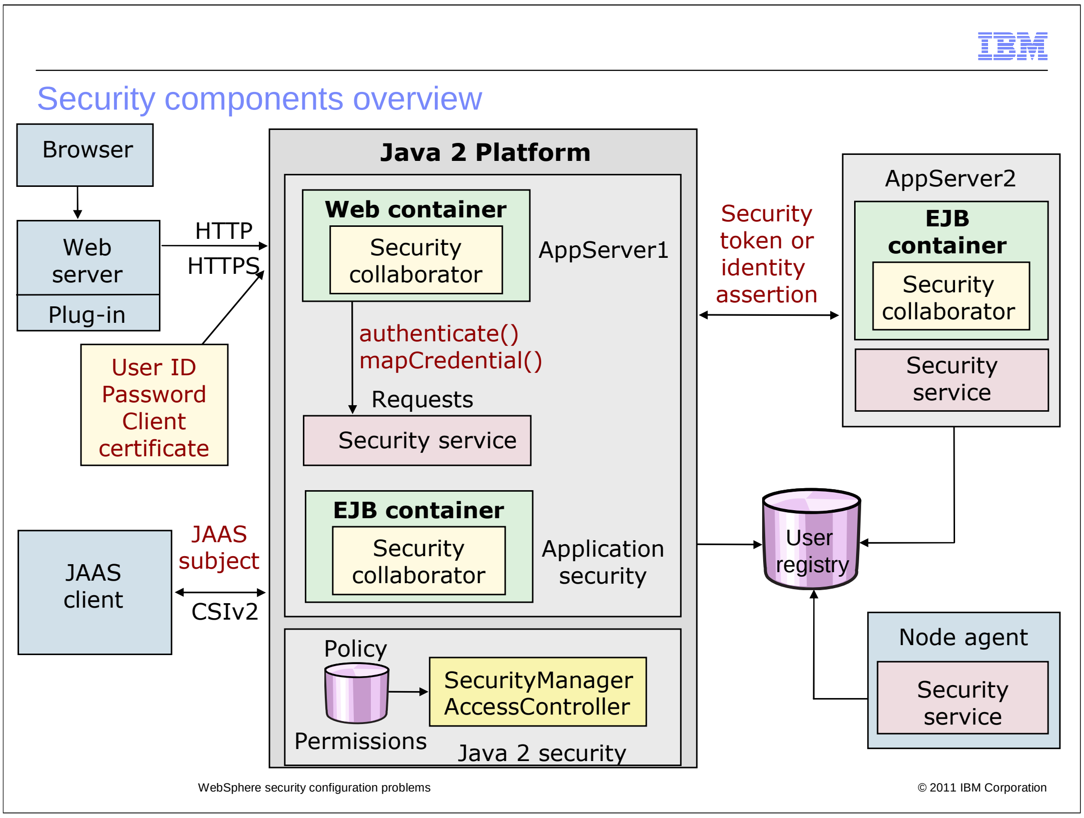
This unit describes how to detect and troubleshoot security related problems.

Unit objectives

- After completing this unit, you should be able to:
- Describe common problems with WebSphere security
- Recognize symptoms of common security-related problems
- Analyze relevant log files for security messages
- Enable server tracing on relevant security components
- Analyze and interpret trace information
- Recognize symptoms of common Secure Sockets Layer (SSL) configuration problems
- Recognize symptoms of common Java 2 security problems
- Locate the security configuration files
- Use tools to validate the security configuration files
- Use wsadmin securityoff to disable security

WebSphere security configuration problems

After completing this unit, you should be able to describe common problems with WebSphere security, recognize symptoms of common security-related problems, analyze relevant log files for security messages, enable server tracing on relevant security components, analyze and interpret trace information, locate the security configuration files, and use tools to validate the security configuration files, as well as use wsadmin command to disable security.

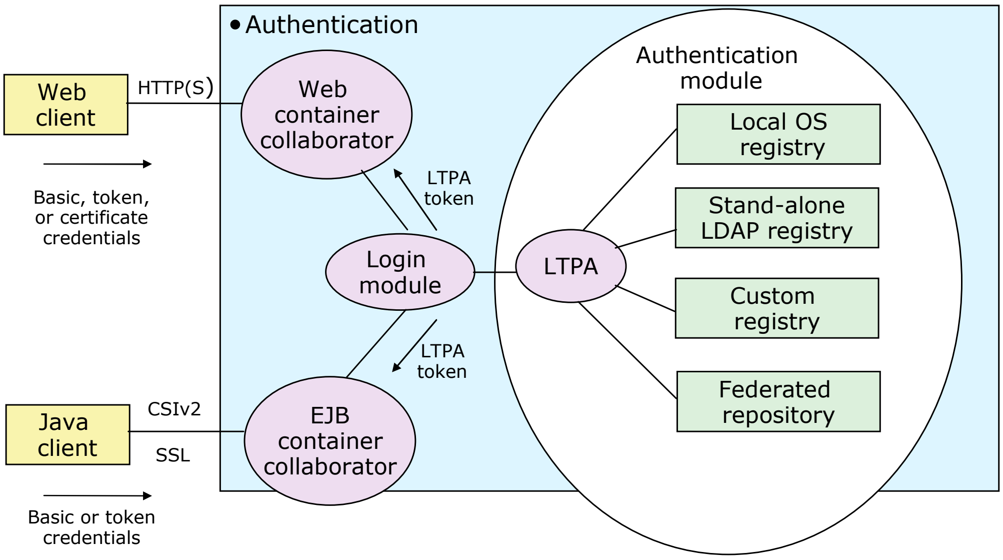


Security is not a single component of the WebSphere Application Server run-time. It is composed of many subsystems working together to provide authentication and authorization services. The primary component for managing the security aspects of the server is the security server component. The security server works with components called security collaborators which are located in server's containers such as the web container and the EJB container. Security collaborators use access manager implementations to do role-based access control. Underneath the WebSphere security layer, the Java 2 security implementation is available to enforce JDK-level security with security policies.

Security flows: Web browser communication

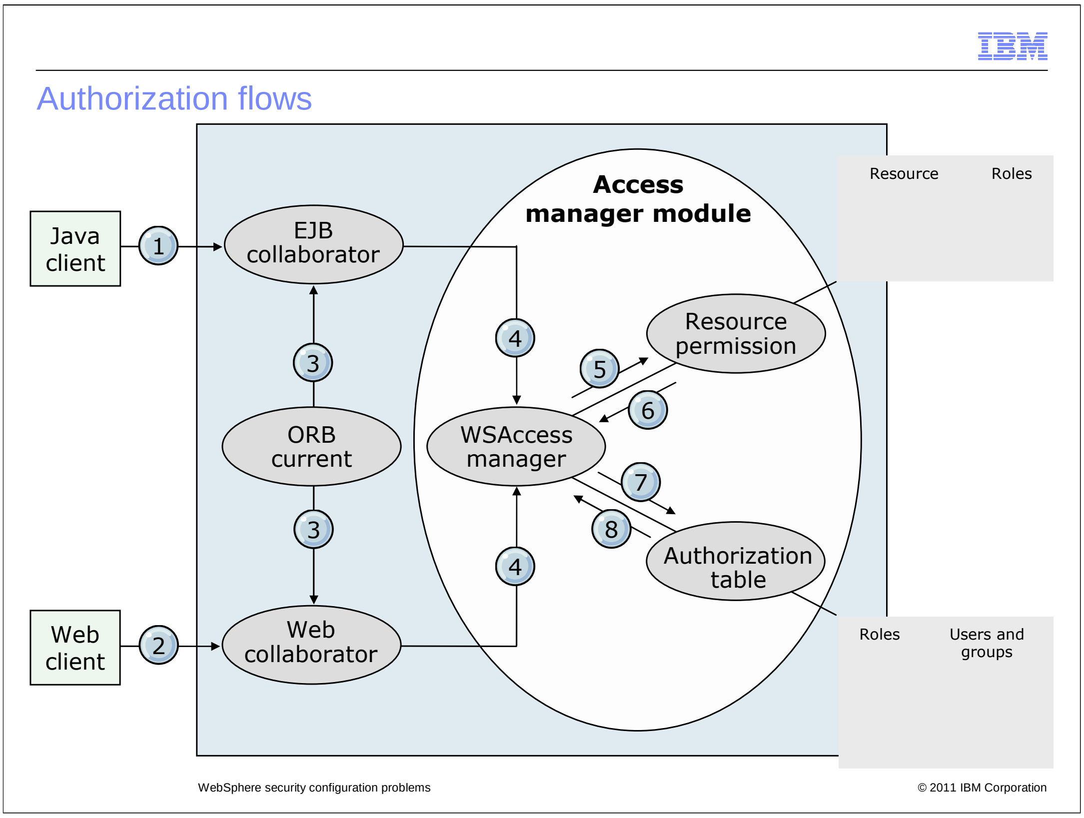
- When a Web browser sends a request to a WebSphere application, the following security interactions occur:
  1. A user requests a Web resource that is protected.
  2. The Web server plug-in receives the request and recognizes that the resource is on the application server.
  3. Plug-in redirects the request to the Web container security collaborator, which calls the JAAS login configuration if necessary.
  4. If authentication is successful, request reaches the Web container.
  5. The Web security collaborator passes the following to the security server for authorization:
     - User credentials
     - Security information contained in the deployment descriptor from EAR file
  6. If the Web application subsequently calls an EJB
     - User credentials are extracted from the established security context.
     - Authorization is performed by the EJB collaborator.

Authentication flows



WebSphere security configuration problems

This diagram describes the process of authentication and authorization for a Web client attempting to access a secured. Note that the Web container security collaborator works in conjunction with a WebSphere logic module to authenticate incoming requests for service using the Lightweight Third-Party Authentication, or LTPA, implementation. The LTPA subsystem can communicate with several types of user repositories such as local OS registries and LDAP directories. Upon a successful authentication by the web client, an LTPA token is given to the client to mark the client as previously authenticated so that subsequent requests to protected resources are not challenged each time.



Web security collaborators and EJB security collaborators both enforce role-based access control by using an access manager implementation. The access manager makes authorization decisions based on the security policy derived from the deployment descriptor.

When the security policy is specified for a Web resource, the Web container performs access control when the resource is requested by a Web client. The Web container challenges the Web client for authentication data if none is present according to the specified authentication method, ensures that the data constraints are met, and determines whether the authenticated user has the required security role. An authenticated user principal can access the requested servlet or JSP file if the user principal has one of the required security roles.

Servlets and JSP files can use the HttpServletRequest methods, isUserInRole and getUserPrincipal.

The EJB security collaborator the authenticated user principal can access the requested EJB method if it has one of the required security roles. EJB code can use the EJBContext methods isCallerInRole and getCallerPrincipal. EJB code also can use the JAAS programming model to perform JAAS login and WSSubject doAs and doAsPrivileged methods.

Note that when security is enabled, the EJB container enforces access control on EJB method invocation. The authentication takes place regardless of whether a method permission is defined for the specific EJB method.

Security flows: Java client communication

- When a Java client interacts with a WebSphere application, the following occurs:
  1. A Java client performs a JAAS login prior to a business request.
  2. The CSIv2 or IBM SAS interceptor performs authentication on the server side on behalf of the ORB, and sets the security context.
  3. Client makes a business request that reaches the server side ORB.
  4. The server side ORB passes the request to the EJB container.
  5. If the request is for a protected EJB method, the EJB container passes the request to the EJB collaborator.
  6. The EJB collaborator reads the deployment descriptor from the EAR file and reads the user credentials from the security context.
  7. Credentials and security information are passed to the security server, which validates user access rights and passes this information back to the collaborator.
  8. After receiving a response from the security server, the EJB collaborator authorizes the client or denies access.

Federated repositories

- Federated repositories provide:
  - Federation capabilities using the virtual member manager (VMM) component
  - Administrative utilities to manage existing users and groups through
    - The administrative console
    - Command line utilities
    - Public APIs
- Integrated with WebSphere Application Server security
  - As a user registry option
    - Federated Repositories option
- Ability to use multiple repositories simultaneously for user registry
  - File based (default out-of-box security)
  - Multiple LDAP directories
  - Databases

Federated repositories are a feature provided with WebSphere Application Server V7.0. Federated repositories allow WebSphere to be configured to use multiple user repositories simultaneously. This integration allows for scenarios when multiple LDAP directories to be used or a combination of database and file-based user registries. Federated repositories can be configured via the WebSphere administrative console, command line utilities, and public Java APIs.

What can go wrong? The short list (1 of 2)

- Errors trying to enable administrative security
  - Invalid user IDs
  - Problems accessing the user registry
- Errors after security is enabled
  - Authentication failures
  - Authorization errors accessing a Web page
- Access and login problems after security is enabled
  - Problems trying to log in to the administrative console
  - Access exceptions if applications are not prepared for Java 2 security
  - Remote user registry inaccessible
  - Node synchronization problems

Some common places were errors can be seen include invalid user ID exceptions, authorization errors accessing a Web page, and problems trying to log in to the administrative console.

What can go wrong? The short list (2 of 2)

- Errors with the SSL configuration
  - Problems accessing resources with HTTPS URLs
  - SSL handshake exceptions
- Single signon configuration problems
  - Authentication failures, mismatching LTPA keys
- User authorization issues
  - Problems with user/group role mappings
- Server fails to start

Additionally, you may experience SSL configuration problems such as problems accessing resources with HTTPS URLs, single sign-on configuration problems leading to authentication failures and mismatching LTPA keys, problems with user and group role mappings, and server startup failures related to security configuration issues.

Approach to troubleshooting security-related issues

- **Does the problem occur when security is disabled?**
  - A good test to determine that a problem is security related.
  - Just because a problem only occurs when security is enabled does not always make it a security problem.
  - More troubleshooting is necessary to ensure the problem is really security-related.
  - Does the problem go away when Java 2 security is disabled?
- **Did security seem to initialize properly?**
  - A lot of security code executes during server initialization.
  - Examine the SystemOut.log and SystemErr.log files to check for warnings and exceptions that are security related.

Troubleshooting security related issues involves analyzing whether the problem occurs when security is disabled. Just because a problem only occurs when security is enabled does not always make it a security problem. You should also ensure that security initializes properly. Additionally, unexpected problems can arise when Java 2 security is enabled. Examine the SystemOut.log and SystemErr.log files to check for warnings and exceptions is highly advisable when troubleshooting security issues.

Normal security initialization messages

AuditServiceI A SECJ6004I: Security Auditing is disabled.

distSecurityC I SECJ0309I: Java 2 Security is disabled.

Configuration A SECJ0215I: Successfully set JAAS login provider… distSecurityC I SECJ0212I: WCCM JAAS configuration information… distSecurityC I SECJ0240I: Security service initialization completed…

SASRas A JSAS0006I: Security connection interceptor initialized.

SASRas A JSAS0001I: Security configuration initialized.

SASRas A JSAS0003I: Authentication mechanism: LTPA

SASRas A JSAS0004I: Principal name:defaultWIMFileBasedRealm/

SASRas A JSAS0007I: Client request interceptor registered.

SASRas A JSAS0008I: Server request interceptor registered.

SecurityCompo A JSAS0009I: IOR interceptor registered.

UserRegistryI A SECJ0136I: Custom Registry… has been initialized

distSecurityC I SECJ0243I: Security service started successfully

distSecurityC I SECJ0210I: Security enabled true

WebSphere security configuration problems

This slide is a screen shot of security messages that were generated in the SystemOut.log during a successful security initialization by the application server.

Authentication or authorization problem

- Many security problems fall under one of these two categories.
- Authentication is the process of determining who the caller is.
  - When authentication fails, this is typically because:
    - Authentication data is not what was expected (wrong ID, wrong password)
    - User being authenticated is not in the registry or password is no longer valid
    - The registry is misconfigured or not accessible
- Authorization is the process of validating that the caller has the proper authority to invoke the requested method.
  - When authorization fails, this is usually related to:
    - Application bindings from assembly and deployment
    - Identity of the caller who is accessing the method
    - Roles that are required by the method

WebSphere security configuration problems

Security problems can be classified into two categories: authentication and authorization. Authentication is the process of determining who the caller is and verifying that they are who they say they are, and,authorization is the process of validating that the caller has the proper authority to invoke a certain method such as appropriate roles.

Problems related to Secure Sockets Layer (SSL)

- SSL is a distinct layer of security.
  - Problems are usually separate from authentication and authorization.
- SSL problems are usually first-time setup problems because the configuration can be difficult.
- Handshake exceptions are common.
  - Each client must contain a valid signer certificate for the server.
  - During mutual authentication, each server must contain valid client certificates.

WebSphere security configuration problems

SSL is a distinct layer of security. Problems in SSL are typically separate from authentication and authorization. Most SSL problems are related to first-time setup and configuration, including the use of IBM's Key Management tool. Other issues include protocol differences resulting in SSL handshaking problems.

Stack trace in the system log file

- A single stack trace tells a lot about the problem.
  - What code initiated the code that failed
  - What component is failing
  - Which class the failure actually came from
- Sometimes the stack trace is all that is needed to solve the problem.
  - It may pinpoint the root cause.
- Other times, it only gives a clue, and may be misleading.
- When product support analyzes a stack trace and it is not clear what the problem is:
  - They may request that you gather additional trace data.
  - You may need to trace several security components with different levels of detail.

The SystemOut and System.err logs contain exception stack trace information that is useful when performing problem analysis. You will be able to see what code initiated the code that failed, what component is failing, and which class the failure actually came from. Sometime the stack trace is not enough to pinpoint the root cause of the failure. You may need to enable several different trace stings and gather additional data to further debug the issue.

Example: SystemOut.log stack trace

- Symptom: The deployment manager appears to start successfully. However, an examination of the SystemOut.log file of the Dmgr shows many exceptions and stack traces. The beginning of the first stack trace looks like:

[7/2/09 15:46:03:849 EDT] 0000000a LdapRegistryI E SECJ0352E: Could not get the users matching the pattern wsbind because of the following exception javax.naming.CommunicationException: DM01:389 [Root exception is java.net.ConnectException: Connection refused: connect]

at com.sun.jndi.ldap.Connection.<init>(Connection.java:222)

at com.sun.jndi.ldap.LdapClient.<init>(LdapClient.java:133)

Here is an example of a security exception stack trace from the deployment manager SystemOut.log. This authentication failure message displays when an external user account repository is corrupted or unavailable, and WebSphere Application Server is unable to authenticate the user name in the repository.

Example: SystemOut.log exceptions

- An attempt to log into the administrative console fails.
  - The following error messages are found in the SystemOut.log:

LdapRegistryI E SECJ0336E: Authentication failed for user wasadmin because of the following exception com.ibm.websphere.security.CustomRegistryException: java.net.ConnectException: Connection refused: connect

LTPAServerObj E SECJ0369E: Authentication failed when using LTPA. The exception is javax.naming.CommunicationException: DM01:389 [Root exception is java.net.ConnectException: Connection refused: connect].

FormLoginExte E SECJ0118E: Authentication error during authentication for user wasadmin

Here is an example of a security exception from the SystemOut.log, after an attempt to log into the administrative console fails.

Tracing security components

- Classes that implement WebSphere security:
  - **com.ibm.ws.security.***
    - **com.ibm.ws.security.audit.***
    - **com.ibm.ws.security.auth.***
    - **com.ibm.ws.security.ejb.***
    - **com.ibm.ws.security.policy.***
    - **com.ibm.ws.security.registry.***
    - others
  - **com.ibm.websphere.security.***
    - **com.ibm.websphere.security.WSSecurityHelper**
    - **com.ibm.websphere.security.WebSphereSecurityPermission**
  - **SASRas**
  - **VMM**
    - **com.ibm.ws.wim.***
    - **com.ibm.websphere.wim.***
    - **com.ibm.wsspi.wim.***

WebSphere security configuration problems

When you need gather additional data to further debug a security issue, you may need to enable several different components. This slide demonstrates a few different trace specifications that can be employed to troubleshoot problems with specific security components.

MustGather tracing requirements

- For specific security problems, the MustGather documents require that certain components be traced.
  - Global security problems
    - **`com.ibm.ws.security.*=all`**
    - **`SASRas=all:com.ibm.ws.security.*=all:ORBRas=all`** (when EBJ authentication is involved)
  - Java 2 security problems
    - **`*=info:com.ibm.ws.security.policy.*=all:com.ibm.ws.security.core.SecurityManager=all`**
  - Federated repository problems
    - **`com.ibm.ws.security.*=all:com.ibm.websphere.wim.*=all:com.ibm.wsspi.wim.*=all:com.ibm.ws.wim.*=all`**

- Typically you should increase the **Maximum Number of Historical Files** from 1 to 10 in the Diagnostic Trace Service

WebSphere security configuration problems

This slide covers examples of trace specifications required to troubleshoot global security problems, Java 2 security problems, and isses arising from the use of federated repositories.

Security tracing can be quite verbose so it may be necessary to increase the maximum number of historical traces from one to ten for the server(s) in which the tracing is being done.

Result from security components trace

- Portion of trace.log file

```
LdapRegistryI > getUniqueUserId Entry wsbind
LdapRegistryI > getUsers Entry wsbind
LdapRegistryI > search Entry
LdapRegistryI 3 DN: dc=ibm,dc=com
LdapRegistryI 3 Search scope: 2
LdapRegistryI 3 Filter: (&(uid=wsbind)(objectclass=inetOrgPerson))
LdapRegistryI 3 Time limit: 3
LdapRegistryI 3 Attr[0]: 1.1
LdapRegistryI > getDirContext Entry
LdapRegistryI 3 try connect to ldap://DM01:389
LdapRegistryI 3 enterJNDI:P=231764:O=0:CT
LdapRegistryI 3 exitJNDI:P=231764:O=0:CT
LdapRegistryI 3 javax.naming.CommunicationException: DM01:389 [Root exception is java.net.ConnectException: Connection refused: connect]
LdapRegistryI A SECJ0418I: Cannot connect to the LDAP server ldap://DM01:389.
        at java.net.PlainSocketImpl.socketConnect(Native Method)
```

The log shows the events leading up to the Communication Exception.

WebSphere security configuration problems

As a result of enabling trace on certain components, you will see more detailed information about your exception and also events leading up to the exception.

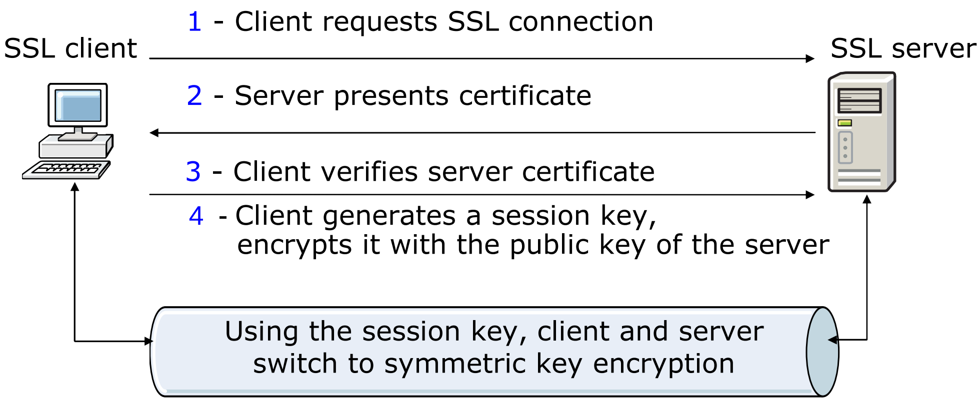
SSL use in WebSphere

- SSL used by WebSphere to provide data encryption and authentication between a client and server.
  - This includes connections to resources outside of WebSphere like LDAP and databases.
- WebSphere uses JSSE as the SSL implementation that is provided by the IBM JRE.
  - JSSE handles the SSL handshake and protection provided by SSL.
- SSL can be configured between many different endpoints in WebSphere.
  - Browser to Web server
  - Plug-in to the application servers
  - Application servers to LDAP servers
  - Application servers to databases

SSL communication is used by WebSphere Application Server to provide secure communication between itself and resources outside of WebSphere such as directory servers, database servers, and messaging servers. WebSphere uses the Java Secure Socket Extension (JSSE) to implement SSL for secure Internet communications. JSSE moderates the SSL handshaking done between two parties wishing to communicate securely. SSL can be setup between many different components in a multi-tiered WebSphere topology such as between Web browser and Web server or between the WebSphere HTTP Plug-In and back-end WebSphere Application Server instances.

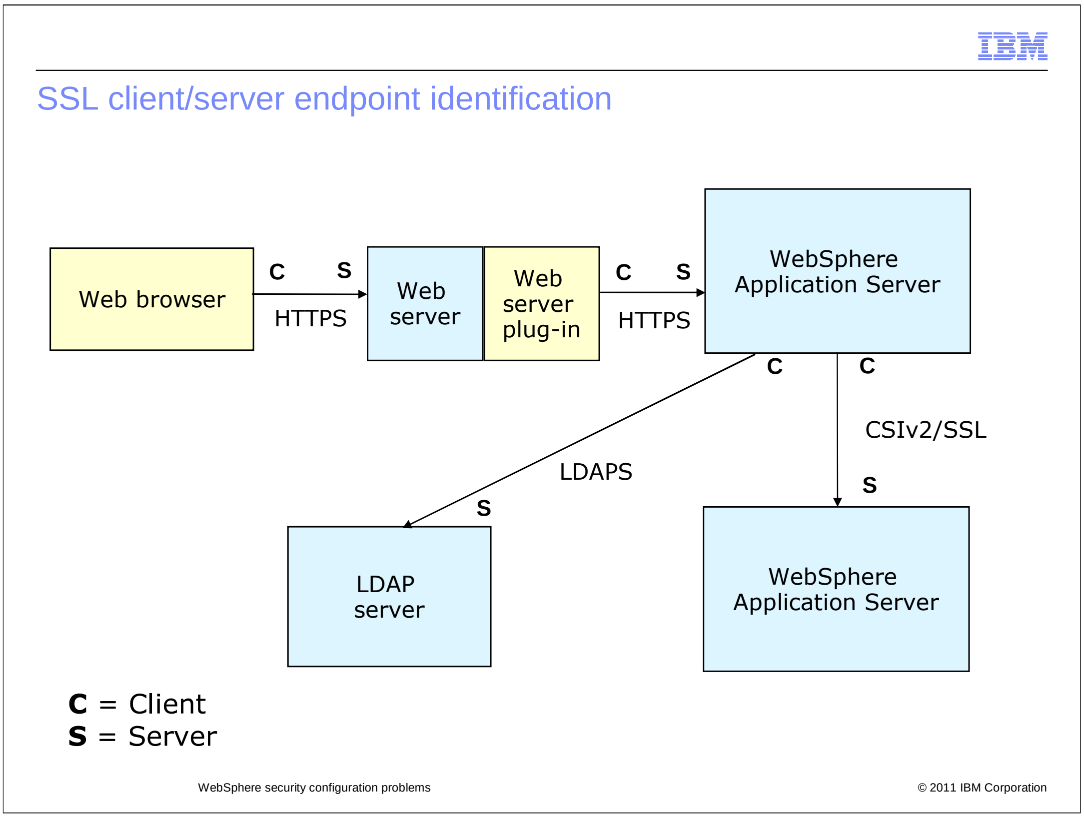
How does SSL work? The "handshake"

- SSL uses a combination of asymmetric and symmetric encryption to create a session between the client and server.
  - Asymmetric encryption is used to negotiate a session key (shared secret).
    - Asymmetric encryption is slow but does not require a shared secret.
  - Symmetric encryption is used to transfer data between the client and server.
    - Symmetric encryption is fast but requires a shared secret.



WebSphere security configuration problems

SSL uses a combination of asymmetric and symmetric encryption methods to create a secured channel for communication between a client and a server. Because the client chooses its own session key, nobody else knows it. It can securely send that session key to the server by first encrypting the it using the public key of the server. Now, nobody but the client and server know the session key. The session key is then used as a “shared secret” to switch to the much more efficient symmetric key encryption. A certificate (or signing certificate) contains information about the server, including the public key of the server, and is digitally signed by the Certificate Authority. Using a certificate, a client or server can assert the authenticity of its identity to the other party. The diagram depicted on this screen details the back-and-forth process of establishing, or building up, the secure SSL communication.



This diagram illustrates the various points in SSL communication. Understanding who is the client and server during in an SSL connection allows the administrator to properly configure the keystores, truststores and other configuration elements required to communicate securely. Understand that “client,” in this context, can refer to a server process. A client is the initiator of an SSL connection, so the Web server plug-in is a client to the application server, and one application server can be a client to another application server. Also, during mutual authentication, the server’s keystore must contain the signer certificate of the client. SSL mutual authentication is merely a statement of trust. It does not look at distinguished names or any other cert content, it just verifies that the certificate was signed by a trusted signer. This is different to what WebSphere will do for an end-user certificate used for authentication, where it will verify that the user is actually in the registry.

SSL problems — handshake failures (1 of 3)

- **javax.net.ssl.SSLHandshakeException: unknown certificate**
  - Some possible causes are:
    - Not having the public key for the target server in the client truststore file
    - The application is setting the following System Properties with WebSphere Security enabled
      - javax.net.ssl.keystore
      - javax.net.ssl.truststore
  - To correct these problems:
    - Export the public key of the server from the keystore file and import it as a signer certificate into the client truststore
    - The recommended solution is to use socket factories, which define their own keystore/truststore, without using the System properties.
    - For an example, go to:
      - **http://www.ibm.com/developerworks/java/library/j-customssl**

WebSphere security configuration problems

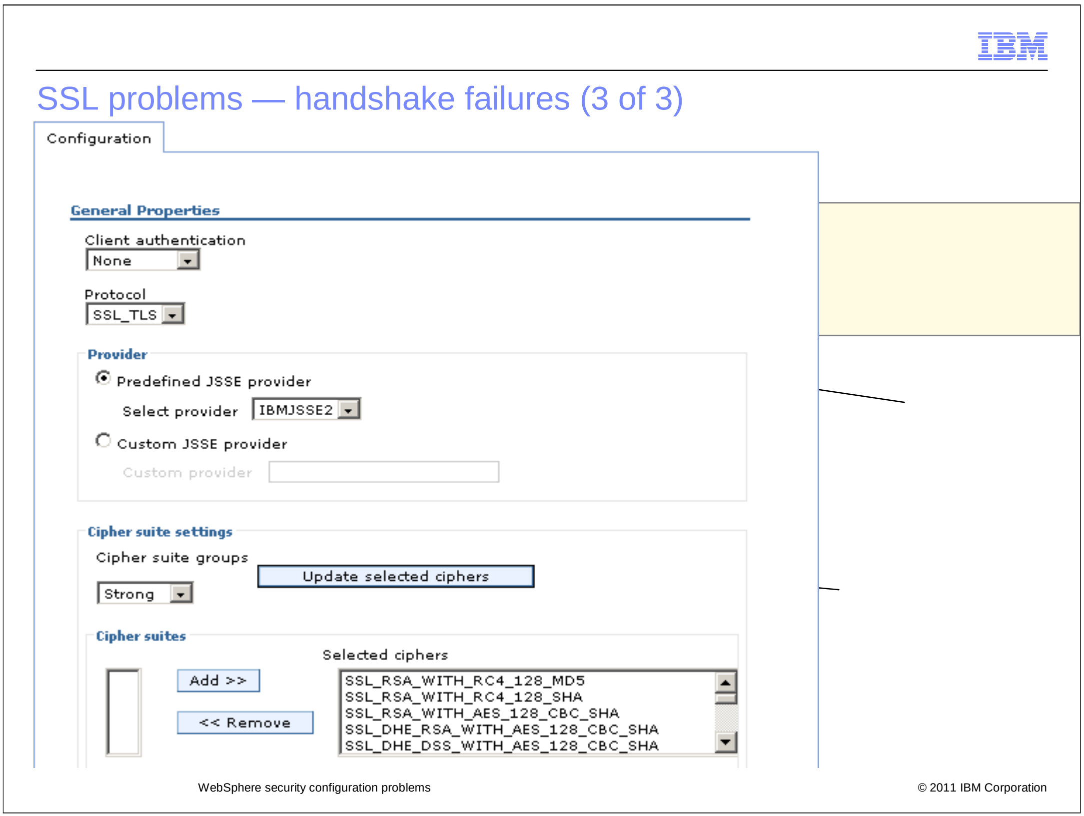
One possible cause of an SSL handshake exception is that the public key for the target server that the client is trying to communicate with, is not present in the client truststore file. To correct this type of issue, one must export the public key of the target server and import it into the truststore used by the client. Configuration of the trust and keystores should be verified as well; if the stores are not defined correctly, the handshake will fail as well.

Typically, the best solution to preventing handshake failures is to utilize socket factories in the client due to the fact that socket factories define their own keystore and truststores explicitly without using the system properties general used to define these values.

SSL problems — handshake failures (2 of 3)

- **javax.net.ssl.SSLHandshakeException —** The client and server could not negotiate the desired level of security. Reason: handshake failure
  - Some possible causes are:
    - Not having common ciphers between the client and server
    - Not specifying the correct protocol
  - To correct these problems:
    - Review the SSL settings in the administrative console
    - Check the property that is specified in the Protocol box matches the client/server
    - Check the cipher suites that are in the Selected Ciphers text box
      - May need to add more cipher suites to the list
    - Correct the protocol or cipher problem by using a different client or server protocol and cipher selection
      - Typical protocols are SSL, SSLv3, TLS

The SSLHandshakeException shown on this slide describes a problem where the client-side and server-side entities of the transaction can not decide on a mutually suitable level of SSL security. Some possible causes of this error are not having common encryption ciphers installed on both the client and the server and potentially specifying the incorrect protocol. To rectify these problems, review the SSL settings in the WebSphere configuration, checking for correct protocol settings and cipher suites. Typical protocols for SSL communications are SSL, SSLv3, and TLS.



The screen capture shown here depicts the configuration panel used to verify and set cipher suites used in SSL communications. Additionally, an administrator can set what JSSE provider WebSphere will use to handle SSL communications and the protocol that will be used to negotiate the conversations. The default protocol is SSL_TLS, which supports all handshake protocols except for SSLv2 on the server side. When United States Federal Information Processing standard or FIPS option is enabled, Transport Layer Security (TLS) is automatically used regardless of this setting. Cipher suites are used to negotiate with the remote side of the connection during the handshake. A common cipher needs to be selected or the handshake fails.

SSL error messages

- The Java Cryptographic Extension (JCE) files were not found
  – Error when launching IKeyman

- **CWPKI0033E: The keystore located at "C:/WebSphere/AppServer/profiles/profile1/etc/trust.p12" failed to load due to the following error: Unable to verify MAC.**

WebSphere security configuration problems

Encountering an error that states that the Java Cryptographic Extension (JCE) files aren't being found by the JVM when trying to start the IBM Key Management Tool (Ikeyman) indicates that there may be a problem with the gskikm.jar file. To resolve this problem set the JAVA_HOME parameter so that it points to the Java Developer Kit that is shipped with WebSphere Application Server, then rename the file gskikm.jar file located in the installation_directory/java/jre/lib/ext to gskikm.jar.org.

The second example error indicates the wrong keystore password was used to unlock and open the keystore file. To resolve this problem change the password field that references this keystore by using the correct password. The default password is WebAS.

Common SSL connection error message

- SSL handshake failure
  - Error when no trusted certificate is found

CWPKI0022E: SSL HANDSHAKE FAILURE: A signer with SubjectDN "CN=192.168.1.204, O=IBM, C=US" was sent from target host:port "192.168.1.12:9403".

The signer may need to be added to local trust store "c:\WASV61\profiles\Dmgr01\etc\trust.p12" located in SSL configuration alias "DefaultSSLSettings" loaded from SSL configuration file:c:\WAS\App\profiles\Dmgr01\properties\ssl.client.props".

The extended error message from the SSL handshake exception is: "No trusted certificate found".

WebSphere security configuration problems

The message above shows an example of one of the most common errors that occurs when attempting to establish an SSL connection from a client to a server. In this case, the server has sent a certificate that is not recognized by the client; that is, the trust store of the client does not contain the corresponding signer. The error message provides detailed information on this error, which should make it easier to correct. Notice that it indicates the host and port, the missing signer, the SSL configuration, and even the trust store that is being used. This tells the administrator precisely what needs to be done: In this case, the missing signer needs to be obtained and added to the trust.p12 trust store specified.

Steps to diagnose SSL handshake issues (1 of 3)

1. Identify the endpoints for the SSL communication.
   - Determine who is the SSL client (client initiating the SSL handshake) and who is the SSL server (the receiving party in the SSL handshake attempt).
   - Sometimes not obvious, as the SSL client can be
     - Java EE client or thin client
     - Web browser
     - WebSphere process
   - The SSL server is typically a WebSphere process.
2. SSL handshake issues occur between two endpoints, and the SSL handshake error usually surfaces in the error log of the client.
   - Determine the SSL configuration, keystore, and truststore for the SSL client and SSL server.

The SSL handshake is the process that occurs when a client opens a socket to a server. If anything goes wrong with the key exchange, cipher exchange, and so on, the handshake fails and the socket is not valid.

Key step to debugging SSL handshake issues, is identifying the endpoints for the SSL communication. Determine who is the SSL client (client initiating the SSL handshake) and who is the SSL server (the receiving party in the SSL handshake attempt).

Note that sometimes finding the endpoints is not obvious, as the SSL client can be a Java EE client or thin client and a WebSphere process.

Steps to diagnose SSL handshake issues (2 of 3)

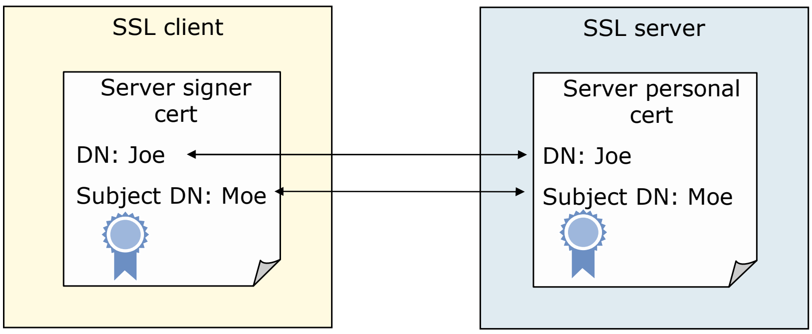
1. Knowledge of the certificates can be used to identify where the setup can be incorrect.
   – Display the certificate information for the signer certificate and personal certificate (if there is one) using one of these tools:
      - Administrative console SSL management
      - iKeyman utility

2. Check the validity period for the signer certificate on the SSL client side and SSL server side.
   – Verify that the current date falls within the start date and end date for the certificates.
   – The certificate's start date should precede the current date and not be expired.

Knowledge of the certificates can be used to identify where the setup can be incorrect. From the SSL certificate and key management page in administrative console you can manage endpoint security configurations and manage certificate expiration. You can Manage your certificates expiration by enabling certificate monitoring from AdminConsole. When the monitor runs, it visits all the key stores and checks to see if they are within certificate expiration range. You can also setup to automatically replace expiring self-signed and chained certificates, if enabled, which a new self-signed or chained certificate is generated.

Steps to diagnose SSL handshake issues (3 of 3)

1. Confirm that the SSL client’s truststore contains the signer certificate of the server.
   - Verify that the issuer’s distinguished name and subject distinguished name for the server’s signer certificate on the SSL client side matches that of the server personal certificate on the SSL server side.



You will also want to confirm that the SSL client’s truststore contains the signer certificate of the server. SSL ensures that the administrator has the CA signer certificate available that is used to sign the personal certificate, and that it is stored in both the client and or the server trusted store. SSL client certificate authentication takes place during the connection handshake by using SSL certificates.

General SSL problem determination steps

- Check the SystemOut.log and SystemErr.log files to determine which SSL problem you are facing.
- Use the search facility within IBM Support Assistant or through the WebSphere Support site to look for common solutions.
- If a common solution is not found, gather a JSSE trace.
  - Set these generic JVM arguments.
    - **`-Djavax.net.debug=true -Djava.security.auth.debug=all`**
    - Restart the application server
  - Recreate the problem and view trace output in the SystemOut.log
    - Look for exceptions and stack traces and work back up the thread to find the lines preceding the exception.
    - The inputs like client key, server key, expiration dates, and ciphers are listed in the trace output

WebSphere security configuration problems

Problem determination for SSL problems is carried out much the way as other component troubleshooting. Check the logs first then use the search facility inside the IBM Support Assistant tool to review the WebSphere support site to look for common solutions. If a solution is not found, set these system property on the client and server processes: -Djavax.net.debug=true. For the server, add the system property to the Generic JVM arguments property of the Java virtual machine settings page. This setting enables detailed tracing that is useful for debugging SSL socket communication.

Recreate the problem and look for exceptions and stack traces in the SystemOut.log, and work back up the thread to find the lines preceding the exception.

Java 2 security problems (2 of 3)

- Java 2 security access control exceptions at run time can result if:
  - An application is not prepared for Java 2 security
  - The application provider does not provide a *was.policy* file as part of the application
  - The application provider does not communicate the expected permissions

- Gather diagnostic data from the SystemOut.log file.
  - Set the **`com.ibm.websphere.java2secman.norethrow`** property for the server.
  - The AccessControl exception contains the
    - Permission violation that causes the exception
    - Exception call stack
    - Permissions granted to each stack frame

Java 2 security access control exceptions at run time can happen if an application is not prepared for Java 2 security or application provider does not provide a was.policy file as part of the application.. You can set the com.ibm.websphere.java2secman.norethrow trace to gather more detailed data from the SystemOut.log file.

Java 2 security problems (3 of 3)

- Handling Java 2 security access exceptions
  - Disable Java 2 security — easy, but will organization security policies allow it?
  - Leave Java 2 security enabled, but then grant
    - Either just enough additional permissions
    - or
    - All permissions to just the problematic application
- Use the PolicyTool (**<*WAS_ROOT*>/java/jre/bin/policytool**) to edit policy files
  - Grant the **java.security.AllPermission** permission in the **was.policy** file that is embedded in the application EAR file
- For example:

```
grant codeBase "file:${application}" { permission
java.security.AllPermission; };
```

WebSphere security configuration problems

One of the easiest ways to debug Java 2 security exceptions is to disable Java 2 security to see if you are still experiencing the problem, but sometimes this is not allowed. Java Development Kit provides the PolicyTool to edit policy files and grant permissions required to resolve AccessControl issues.

Security Association Service messages

- JSAS messages are from Security Association Service.
  - Examples

JSAS0201E: [{0}] Invocation credential realm does not match target's realm: {0}. If using the SWAM authentication mechanism, you should switch to using LTPA instead for remote IIOP invocations.
Explanation: Attempting a remote invocation over IIOP using the SWAM authentication mechanism is not supported.
User Response: Retry with the LTPA authentication mechanism configured in Global Security

JSAS0202E: [{0}] Credential token expired. {1}
Explanation: The credential token associated with the user credential has expired. This typically occurs with LTPA.
User Response: Close the client and login again.

Here are some examples of security authentication exceptions you might see in the SystemOut or SystemErr logs, which typically will start with JSAS in the error code message.

WebSphere Security messages

- SECJ messages are from WebSphere Security.
  - Examples

SECJ0007E: Error during security initialization. The exception is {0}.
Explanation: An unexpected error occurred during security initialization.
User Response: This is a general error. Look for previous messages that may be related to the failure or a configuration problem. Enabling security debug trace for components com.ibm.ws.security.* and com.ibm.ejs.security.* may yield additional information.

SECJ0056E: Authentication failed for reason {0}
Explanation: Authentication failed with the specified reason.
User Response: Verify that the user id and password are entered correctly. Consult with the administrator of the user registry if the problem persist.

Here are some examples of WebSphere security exceptions you might see in the SystemOut or SystemErr logs, which typically will start with SECJ in the error code message.

Web UI Security Center messages

- SECG messages are from web UI Security Center.
  - Examples

SECG0005E: An exception occurred when exporting Lightweight Third Party Authentication (LTPA) keys: The exception is {0}.
Explanation: Unable to get the Lightweight Third Party Authentication (LTPA) keys from the server.
User Response: Regenerate the keys and try the operation again

SECG0027E: The Ignore case option is required for the Lightweight Directory Access Protocol (LDAP) directory type {0}.
Explanation: Select the Ignore case option for the LDAP directory type selected.
User Response: Enable the Ignore case option in the administrative console. Expand Security > User Registries. Click LDAP and select the Ignore case option.

Here are some examples of WebSphere UI security center exceptions you might see in the SystemOut or SystemErr logs, which typically will start with SECG in the error code message.

Web services security (WS-Security) messages

- WSEC messages are from web services security.
  - Examples

WSEC0001E: Error trying to find Security Server. The exception is {0}.
Explanation: This exception is unexpected. The cause is not immediately known.
User Response: If the problem persists, see problem determination information on the WebSphere Application Server Support…

WSEC0007W: Server level Web Services Security configuration file {0} is not found.
Explanation: The server level Web Services Security configuration document might be corrupted or missing. The file provides the default binding configuration for Web Services Security.
User Response: If you would like to use the default bindings information, please copy ws-security.xml from the ${USER_INSTALL_ROOT}/config/templates directory.

Here are some examples of WebSphere Services security exceptions you might see in the SystemOut or SystemErr logs, which typically will start with WSEC in the error code message.

Virtual member manager (VMM) messages

- CWWIM messages are from the virtual member manager.
  - Examples

CWWIM0000E: The virtual member manager configuration XML file 'file_name' was not found.
Explanation: Virtual member manager cannot run without the configuration information. Virtual member manager cannot continue without this file. The name of this file is wimconfig.xml.The path of this file is VMM_HOME/config...
Programmer Response: Ensure that this file exists.

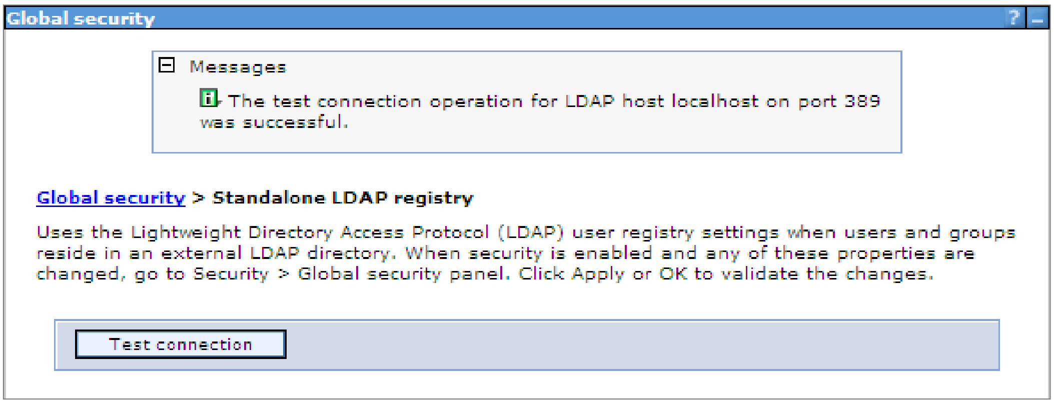
CWWIM0500E: The 'realm_name' realm name specified is not valid.
Explanation: The realm name is not defined in the virtual member manager configuration file.
Programmer Response: Ensure that the realm name exists and that it is spelled correctly. If it does not exist, you must define the realm or specify a different, valid realm.

WebSphere security configuration problems

Here are some examples of Virtual member manager exceptions you might see in the SystemOut or SystemErr logs, which typically will start with CWWIM in the error code message.

Administrative console security PD tools (1 of 3)

- The **Test connection** button attempts to connect to the LDAP server from the deployment manager using the LDAP server host name and port.

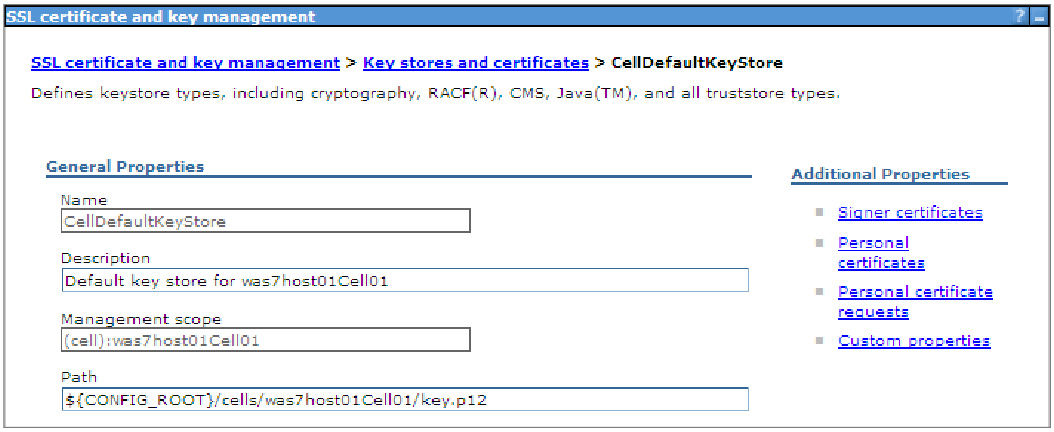


WebSphere security configuration problems

The administrative console provides some facilities to assist in security-relates problem determination efforts. For example, a “test connection” button is provided for testing the connection to the configured LDAP server from the deployment manager using the LDAP server host name and port, and returns a result message.

Administrative console security PD tools (2 of 3)

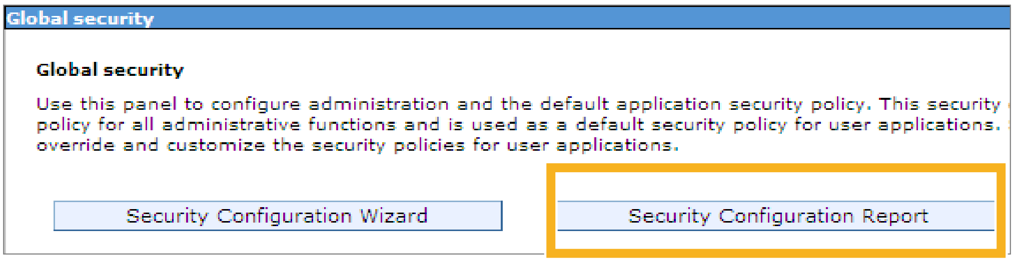
- SSL certificate and key management
  - Certificate expiration, endpoint SSL configuration, key managers, trust managers
  - Keystores and certificates
    - Based on iKeyman functionality from previous WebSphere versions



The administrative console also provides SSL certification and key management to assist in certificate expiration, endpoint SSL configuration, and other security related problem determination efforts.

Administrative console security PD tools (3 of 3)

- Security Configuration Report button
  - Launches a security configuration report that displays the core security settings of the application server
    - Also displays the administrative users and groups and the CORBA naming roles



The security configuration report gathers and displays the current security settings of the application server. Information is gathered about core security settings, administrative users and groups, CORBA naming roles, and cookie protection. When multiple security domains are configured, each security domain has it's own report with a subset of the sections shown in the global security report that apply to the domain.

Security configuration files (1 of 3)

- security.xml
  - Each profile has a copy of security.xml of the deployment manager located at **<*WAS_HOME*>\profiles\<*Profile_Name*>\config\cells\<*Cell_Name*>**
  - Contains all of the security configuration information and status.
    - User registry
    - Authentication mechanism
    - Many more
  - Each server has its own copy of security.xml that can override the cell-wide configuration.
    - Enable or disable application security
    - Enable or disable Java 2 security
- ws-security.xml
  - Each application server has a copy of the ws-security.xml file, which defines the default binding information for web services security.

The next few slides will talk about some of the important security configuration files.

Security.xml file contains all of the security configuration information on user registry, authentication mechanism, and much more. Also, each application server can have its own security.xml file to override the cell-wide configuration.

ws-security.xml file defines the default binding information for web services security.

Security configuration files (2 of 3)

- sas.client.props and soap.client.props
  - Each profile has a copy of sas.clients.props in its properties directory.
  - Soap.client.props is used to store administrator ID and password.
- app.policy and was.policy
  - Contain policy information for Java 2 Security.
  - Each profile has a copy of app.policy in its properties directory.
  - Each application has a copy of was.policy in its EAR file.
- wimconfig.xml
  - Each profile has a copy of wimconfig.xml located at **<*WAS_HOME*>\profiles\<*Profile_Name*>\config\cells\<*Cell_Name*>/wim/config**
  - Contains all of the virtual member manager configuration information
- **<*WAS_HOME*>\etc\wim**
  - This directory contains the virtual member manager setup and migration files

WebSphere security configuration problems

Here are more of the important security configuration files.

The sas.client.props file configures the Secure Sockets Layer (SSL) client authentication.

soap.client.props file is used to store administrator ID and password.

Java 2 security uses several policy files to determine the granted permission for each Java program.

app.policy file is a default policy file shared by all of the WebSphere Application Server applications.

was.policy file is an application-specific policy file for WebSphere Application Server applications, it is also embedded in the enterprise archive (EAR) file (META-INF/was.policy).

The wimconfig.xml file is the main file that controls the behavior of the Virtual Member Manager component (VMM).

Security configuration files (3 of 3)

- fileRegistry.xml
  - If file-based user registry is configured, each profile has a copy of fileRegistry.xml located at **<*WAS_HOME*>\profiles\<*Profile_Name*>\config\cells\<*Cell_Name*>**
  - This is the file repository and it contains:
    - User and group identifiers
    - Encrypted passwords for users

The fileRegistry.xml file contains user and group identifiers, including the encrypted passwords for the user entries.

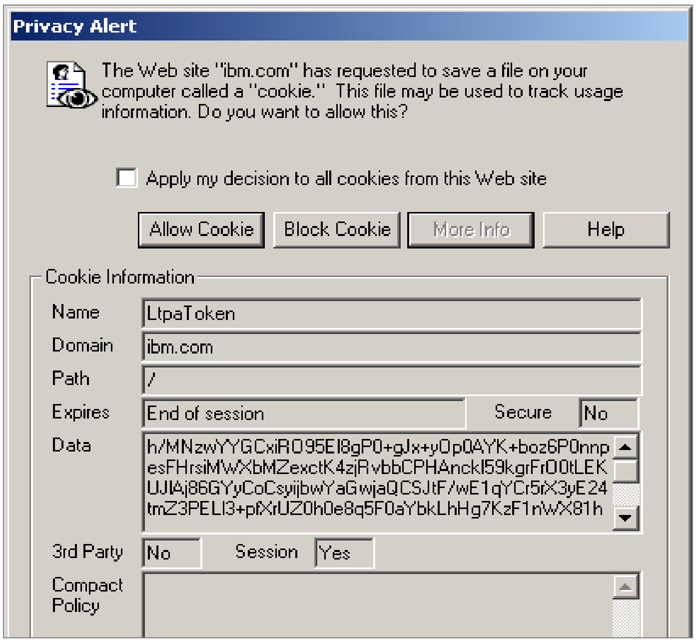
Tools to validate the security configuration

- Various tools can be downloaded from the IBM Support site for checking and validating security configuration.
- These tools include:
  - ACert
    - A command-line tool that checks expiration dates of all SSL certificates defined in WebSphere Application Server SSL repositories
    - The expiration dates of each certificate are displayed
    - Similar functionality built into administrative console as of V6.1
  - WebSphere Security Scanning Tool
    - A command line tool that scans static WebSphere Application Server security configuration files to look for potential vulnerabilities
- LDAP browsers and editors
  - Very useful for working with LDAP user registries

There are various tools available from IBM Support site which check and validate security configuration. ACert is a command-line tool that checks expiration dates of all SSL certificates. The WebSphere Security Scanning Tool scans static security configuration files to look for potential vulnerabilities. Additionally, having an LDAP Browser is useful for working directly with LDAP user registries and verifying the structure of the directory.

Tracking LTPA tokens

- To track LTPA tokens, configure your web browser to warn about cookies
- If you do not get a warning after a successful authentication you can have problems with
  - Domain suffix for the LTPA SSO configuration
  - Proxy server configuration



At times, it is useful to determine how WebSphere Application Server is managing the LTPA cookie or token. To gain insight into this process you should enable your web browser to warn about cookies. Once you do this, your browser will inform you when WebSphere Application Server sends back an LTPA token.

Disabling administrative security

- Sometimes it is necessary to disable administrative security in order to troubleshoot security-related problems.
- If the application server or deployment manager is running, use the administrative console.
  1. Select **Security > Secure administration**
  2. Clear **Enable administrative security**
  3. Save changes to the master repository
- If the application server cannot be started, for example
  - Password of the server user ID in the user registry is expired
  - The user registry cannot be reached for authentication
    - Disable administrative security using the command line
      1. **<WAS_HOME>\bin\wsadmin.bat -conntype NONE**
      2. **wsadmin>securityoff**
      3. **wsadmin>quit**
      4. **Start server1 or dmgr**
    - **Note:** Only changes the security.xml file of the local node

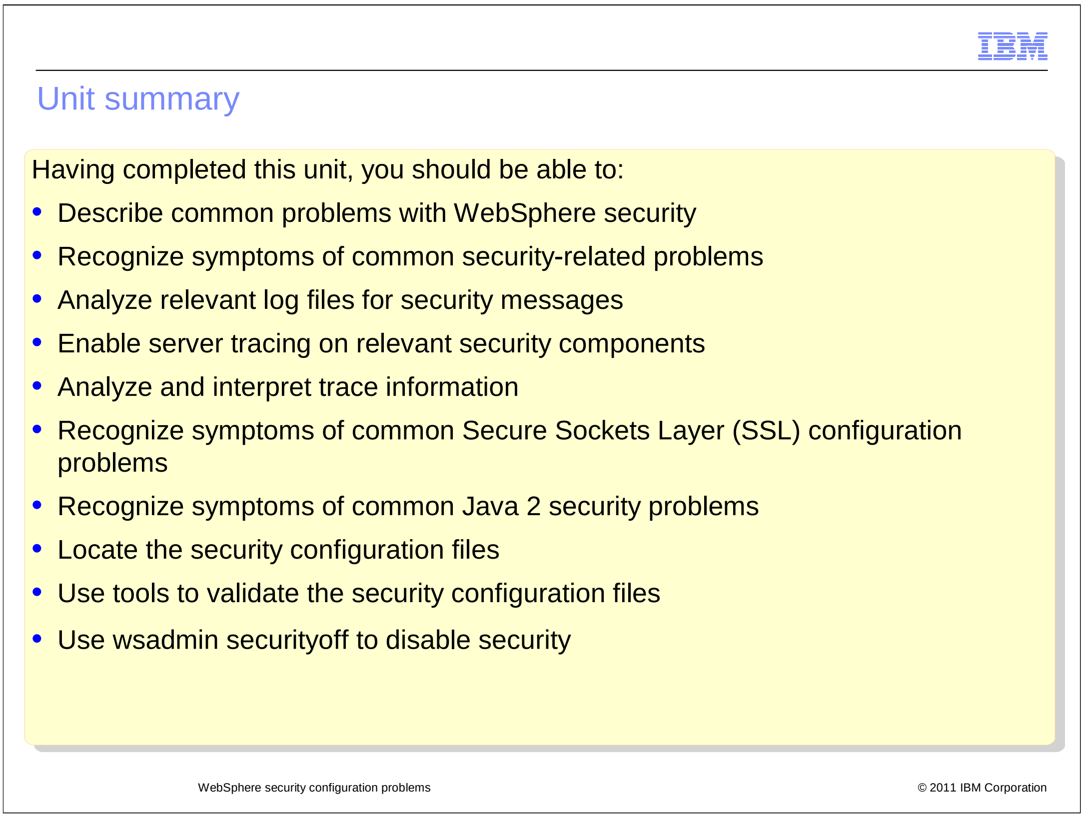
WebSphere security configuration problems

•Sometimes it is necessary to disable administrative security in order to troubleshoot security-related problems.

•If the application server or deployment manager is running, use the administrative console to navigate to the secure administration page, and clear the Enable administrative security indicator.

•If the application server cannot be started, because the password of the server user ID in the user registry is expired, the user registry cannot be reached for authentication, disable administrative security using the command line, as shown here.

•**Note that o**nly changes the security.xml file of the local node



Now that you have completed this unit, you should be able to, describe common problems with WebSphere security, recognize symptoms of common security-related problems, analyze relevant log files for security messages, enable server tracing on relevant security components, analyze and interpret trace information, locate the security configuration files, and use tools to validate the security configuration files.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WA5721G11_Security_Config_Probs_edited.PPT

This module is also available in PDF format at: ../WA5721G11_Security_Config_Probs_edited.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, disclaimer, and copyright information

IBM, the IBM logo, ibm.com, Approach, System p, and WebSphere are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.
Other company, product, or service names may be trademarks or service marks of others.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.