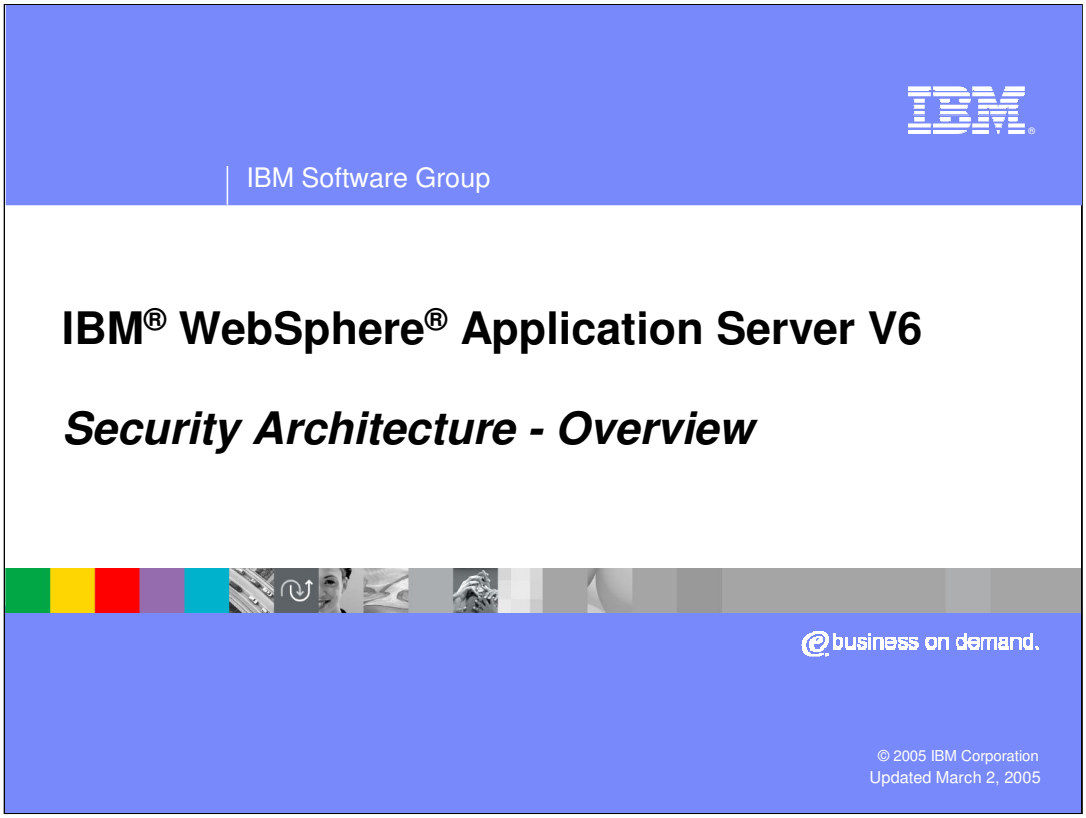


This presentation will focus on the Security Architecture at a high level.

Goals

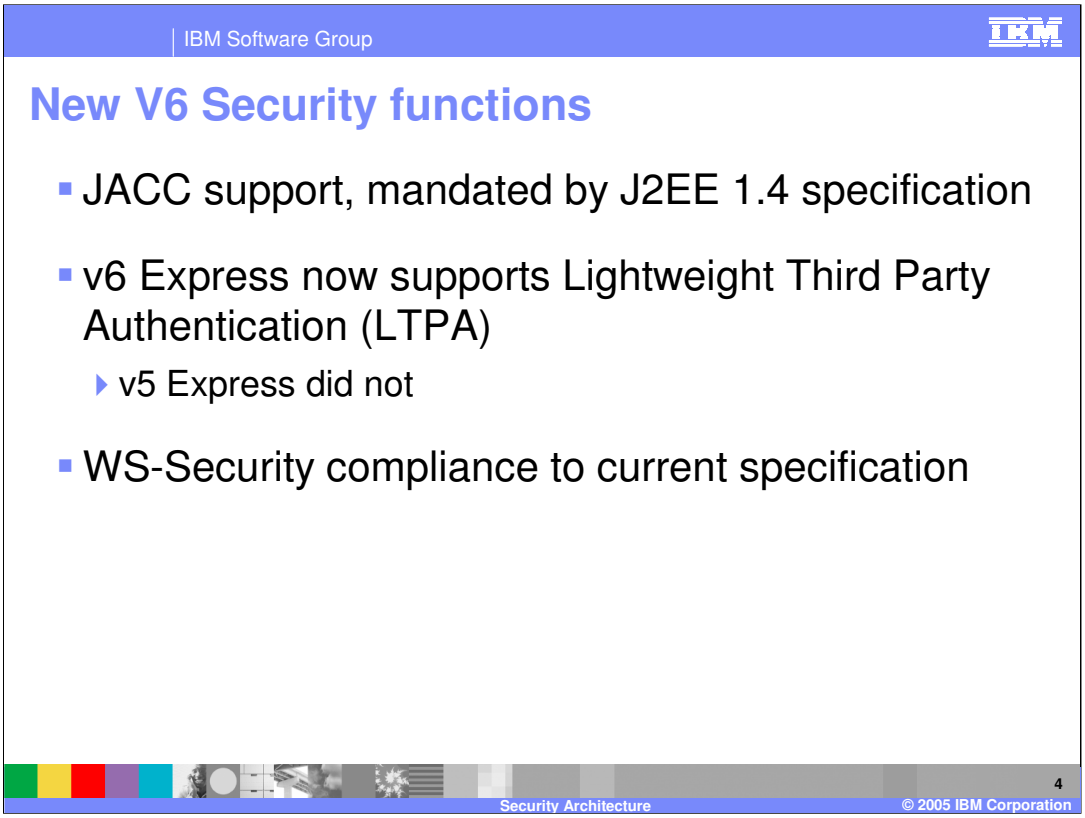
- Understand high level Security Architecture in WebSphere Application Server V6

The goals for this presentation are as listed on this page. There are many other Security related presentations that will go into the details of different Security components and functions.

Agenda

- WebSphere V6 Security - Basics
- Java™ 2 Enterprise Edition (J2EE) 1.4 Security Features
- Authentication Mechanism and User Registry
- Single Sign On Support
- Trust Association
- JACC support
- WS-Security support
- Summary

The agenda is as listed on this page.

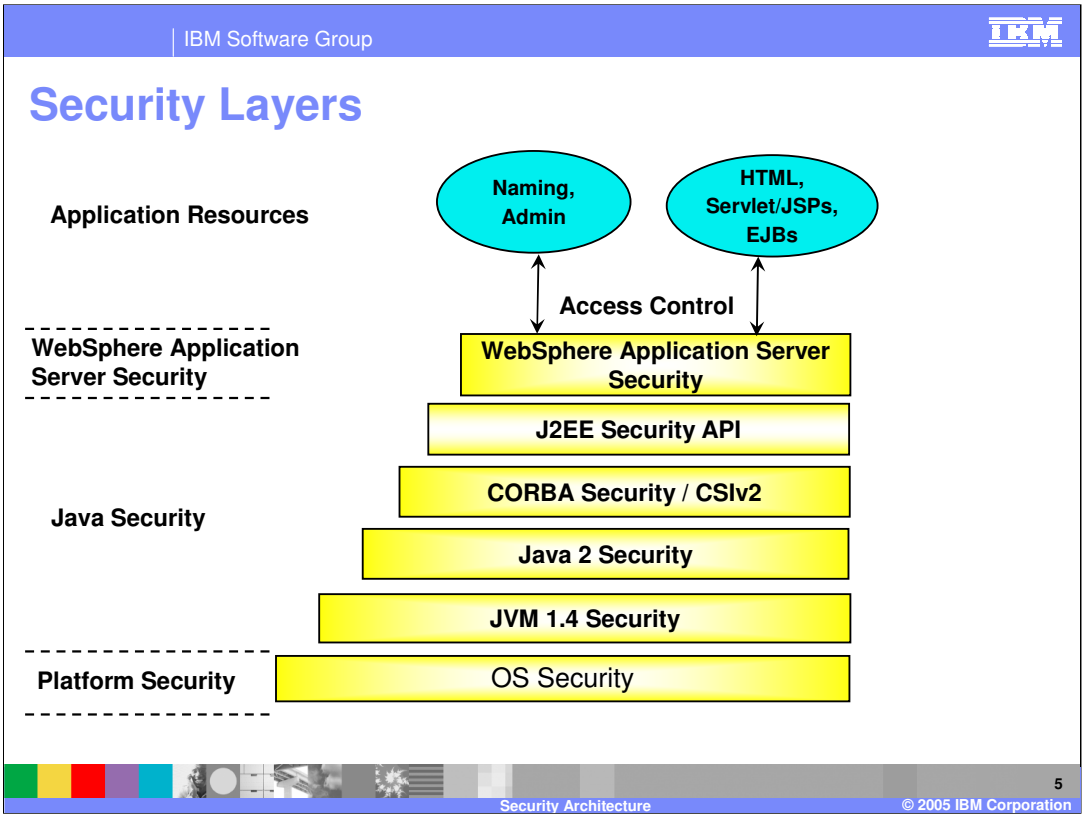


There have been few enhancements in version 6 Security functions. Most of the security related functions and administration are similar to version 5.

New to version 6 is the support for JACC, the Java Authorization Contract with Containers, as required by J2EE 1.4.

In addition, WS-Security specification has been made more current.

The Security in all version 6 packages have the same capabilities. This was not true in V5, where the Express package had limited Security functions and was missing the Local Third Party Authentication (LTPA) mechanism.



In WebSphere v4 there was security at many different levels. For WebSphere V5 and V6, the security options and capabilities at many of these levels has been enhanced. With WebSphere V5 on the zSeries platform, the WebSphere security layers are intended to work like those on any other platform. What is unique to each platform is the OS security.

•Operating System Security - The security infrastructure of the underlying operating system provides certain security services to the WebSphere Security Application. This includes the file system security support to secure sensitive files in WebSphere product installation. The WebSphere system administrator can configure the product to obtain authentication information directly from the operating system user registry.

•JVM 1.4 - The JVM security model provides a layer of security above the operating system layer.

•CORBA Security - Any calls made among secure ORBs are invoked using a IBM proprietary Secure Association Service (SAS) or J2EE standard CSIv2 authentication protocol that sets up the security context and the necessary quality of protection. After the session is established, the call is passed up to the enterprise bean layer.

•J2EE Security - The security collaborator enforces J2EE based security policies and support J2EE security APIs.

•WebSphere Security - WebSphere security enforces security policies and services in a unified manner on access to Web resources and enterprise beans. It consists of WebSphere security technologies and features to support the needs of a secure enterprise environment.

Security Basics

- The security configuration and setting is cell wide in a Network Deployment cell
  - DMgr, all Node Agents and all Servers have the same security configuration applied
  - Some local security override permitted on per Application Server
- Global security needs to be enabled

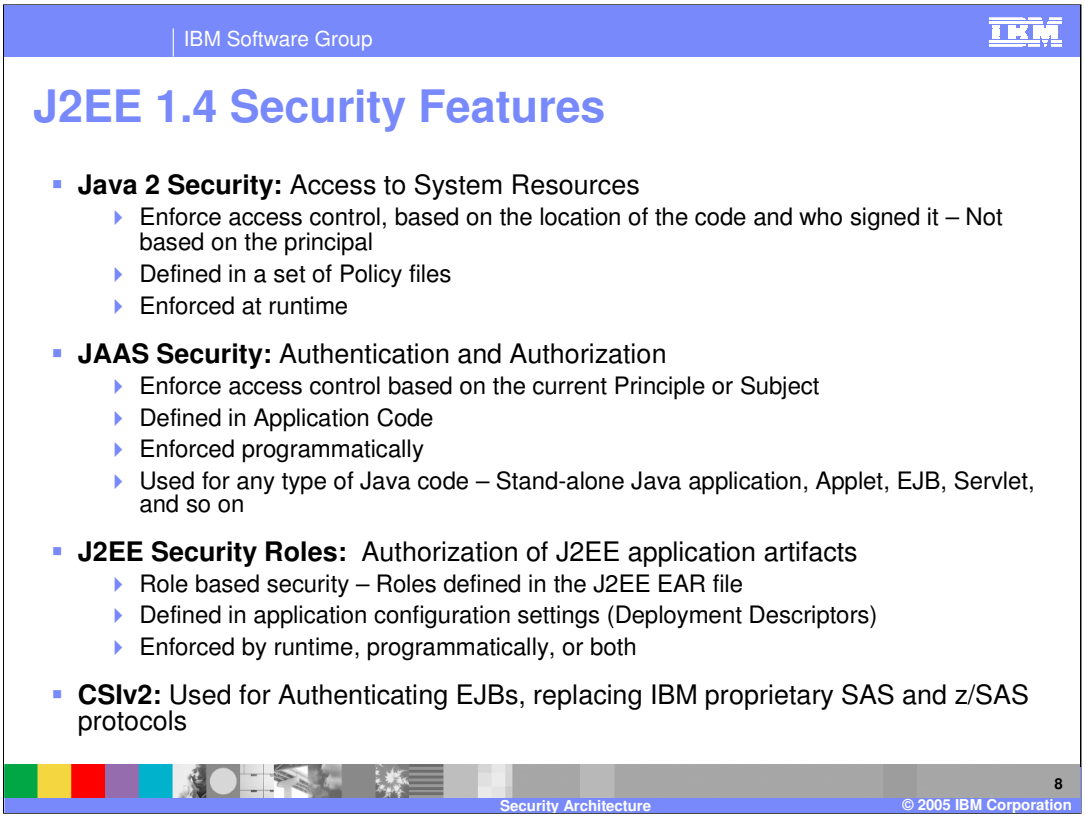
Global Security is either ON or OFF for the entire Cell in a Network Deployment package. The Authentication type and Registry applies to the entire cell also. You cannot have some Servers within the cell have different global security or different authentication mechanism than other servers within the same cell.

However, on per Application server within the cell, you might choose to turn off Java 2 security as well as J2EE Application Security, if you can trust the applications running on those servers.

Section

J2EE 1.4 Security Features

The next section will discuss the overview of J2EE 1.4 Security, which is the same as J2EE 1.3.



J2EE 1.4 Security defines the 4 specifications listed here.

The main difference between Java 2, JAAS, and J2EE Security is how the security is enforced.

Java 2 Security is enforced by the JVM and by the Server with the permissions specified in policy files.

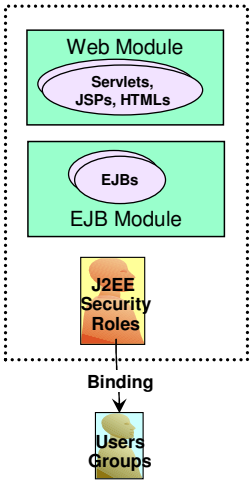
JAAS Security is enforced programmatically from within an application.

J2EE Security is enforceable by the J2EE application server or programmatically from within an application.

CSIv2 is the authentication protocol, used by EJB and EJB client over RMI-IIOP. WebSphere V6 continues to support IBM SAS protocols for backward compatibility, when connecting to and from EJBs running in version 4.

J2EE Security Roles: Application Authorization

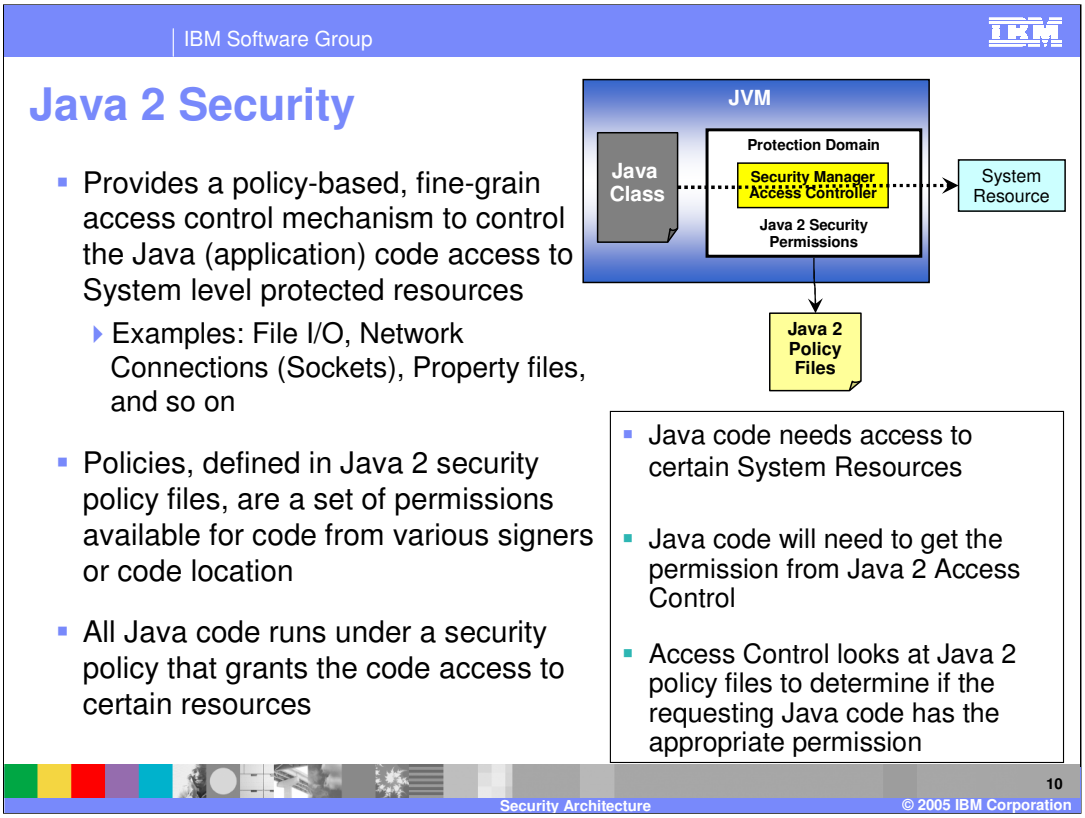
- Authorization is performed using J2EE Security Roles
  - Allows developer to specify security at an abstract level
- Security roles are applied to the Web and EJB application components
  - EJB methods or Web URIs
- Binding users and groups to J2EE security roles is usually done at application install time



The developer of the J2EE application can define Security roles within the application and assign permissions for those roles to access all or some of the EJB or Web artifacts. The defined Security roles apply to the entire Application (and all its modules). This allows the developer to provide method level permission by using abstract Roles, since they might not know where the application will be running and who the users or groups are defined for that Application Server on which the application runs.

The task of mapping the Security roles to the users or groups is normally performed by the System Administrator within the Application Server. This is called binding. With the support of JACC, you can use IBM® Tivoli® Access Manager or third party JACC provider to provide the binding information.

More details on this topic are available in the J2EE Security presentation.



Java 2 security is about protecting malicious code as part of an application, having access to the System resources (like files, ports, sockets, etc.)

So, if you need to allow the code to access System resources, you will need to give necessary permission. These are defined in policy files.

WebSphere supports a hierarchical definition of policy files at various levels - at application level, server level, etc. for different code running within WebSphere.

Before giving permission to the code to access a System resource, the access control mechanism within the JVM will look at the policy files to determine if the correct permission is provided. If not, then no access would be given.

More details on this topic is available in the Java 2 Security presentation.

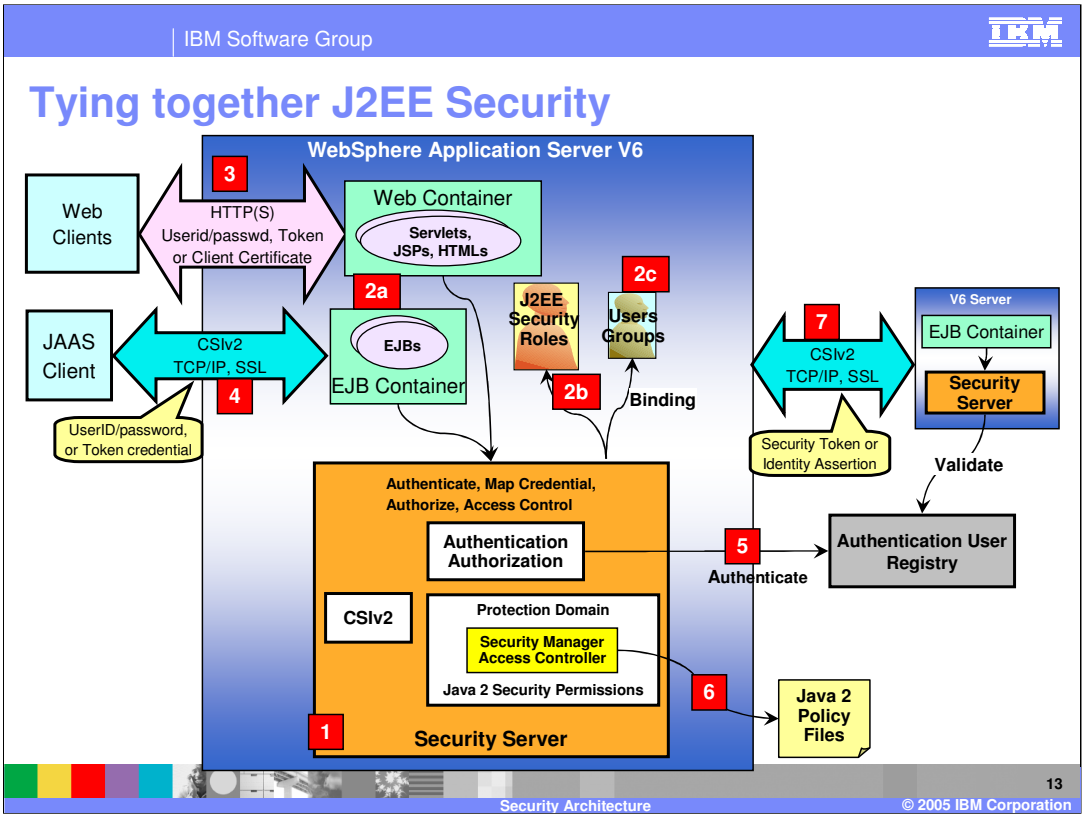
Java Authentication & Authorization Service-JAAS

- Set of Java 2 Security APIs used to establish identity and perform authorization:
  - Authentication determines who is currently executing the code, regardless of where the code is running
  - Authorization of the users, based on JAAS security policies to specify what access rights are granted to executing code
- JAAS authentication is based on Pluggable Authentication Module
  - Allows applications to remain independent from underlying authentication technologies

JAAS is a set of APIs to allow authentication and authorization in a pluggable fashion, in that, you can replace the authentication technologies without need to make any change to the application. JAAS configuration (specified in an external file for external clients, or within WebSphere for artifacts running within WebSphere) provides the Pluggable Authentication mechanism that makes it very flexible.

CSIv2: Authentication protocol for EJB Security

- Common Secure Interoperability Specification, Version 2 (CSIv2)
  - Defines the Security Attribute Service (SAS) that enables interoperable authentication, delegation and privileges
  - Defined by Object Management Group (OMG) standard to provide open, secure interoperability common framework across J2EE servers
- CSIv2 Protocol facilitates interoperability by serving as the higher-level protocol under which secure transports (SSL/TLS) can be unified
- CSIv2 configuration is integrated within WebSphere Application Server Security Administration



This picture ties the J2EE Security together:

(1) Security Server that performs all security authentication, authorization and other security related functions, like checking for Java 2 permissions, etc.

(2) Web and EJB applications (2a) with the J2EE Security Roles (2b) defined for the applications. J2EE Security Role to user/group binding (2c). This allows the Security Server to know which J2EE Security Roles the user/group belongs.

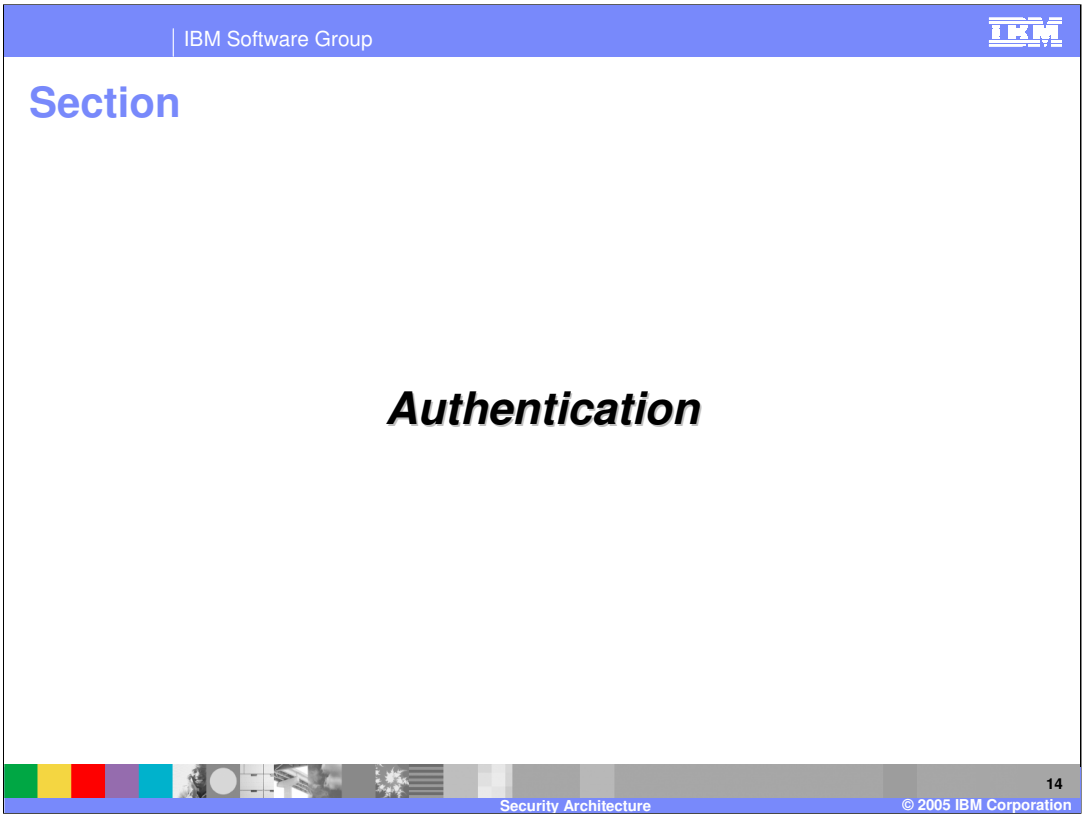
(3) Web client requests directly or through a Web Server using HTTP or HTTPs. Once authenticated by the Security Server, authorization is checked based on J2EE Security Roles and permissions

(4) JAAS client calling EJBs using CSIv2 authentication protocol, making RMI-IIOP calls with or without SSL. Once authenticated, authorization is checked based on J2EE Security Roles and permission

(5) All authentication is checked against the Supported User Registry

(6) Java 2 Security policy files used by the JVM Access Controller to check if the executing Java code has the necessary permissions to access the requested System resource

(7) Application within WebSphere acting as an EJB client to an EJB in another server and using CSIv2 authentication protocol



The next section will discuss authentication in WebSphere Application Server V6.

Authentication

- Authentication is the process of establishing whether a client is valid in a particular context
  - Client can be either a user, a machine, or an application
- An *authentication mechanism* defines rules about security information and how security information is stored in both credentials and tokens
  - WebSphere V6 on distributed platform supports Lightweight Third Party Authentication (LTPA) and Simple WebSphere Authentication Mechanism (SWAM)
- Authentication Mechanism uses User (Authentication) Registry, (where user ID, password, and other attributes are stored) to check the client authentication
  - WebSphere supports several User Registries - Local OS, LDAP and Custom Registry
- Allows for customization of authentication process via JAAS pluggable authentication

As indicated before, the selected Authentication mechanism and User Registry applies to the entire Network Deployment cell.

Authentication (cont.)

- In addition, WebSphere Application Server supports Single Sign On (SSO), Trust association, and Security Context Propagation
  - SSO - authenticate only one time, when accessing Web resources across multiple WebSphere Application Servers
  - Trust association allows 3rd party Reverse Proxy Security servers (RPSS) to act as a front-end authentication server for Web HTTP Requests into WebSphere Application Server
  - Enhanced trust association allows 3rd party RPSS to assert security attributes in addition to security identity
  - Security context propagation – propagates client security identity and security attributes from front-end authentication server to back-end WebSphere Application Servers in a multiple-hop, chain of requests

Section

Secure Sockets Layer (SSL) Support

The next section will discuss Secure Socket Layer support in WebSphere Application Server V6.

Secure Sockets Layer (SSL) in WebSphere V6

- SSL provides transport layer security with authenticity, integrity, and confidentiality, for a secure connection between a client and server in WebSphere V6
- SSL is used by multiple components within the Application Server
  - Built-in HTTP server within the Web Container for HTTP over SSL
  - ORB component using IIOP over SSL
  - Security LDAP client using LDAP over SSL to connect to LDAP user registry
- Administration:
  - You configure SSL settings using SSL repertoire
  - Once defined, you can use the SSL definitions where needed

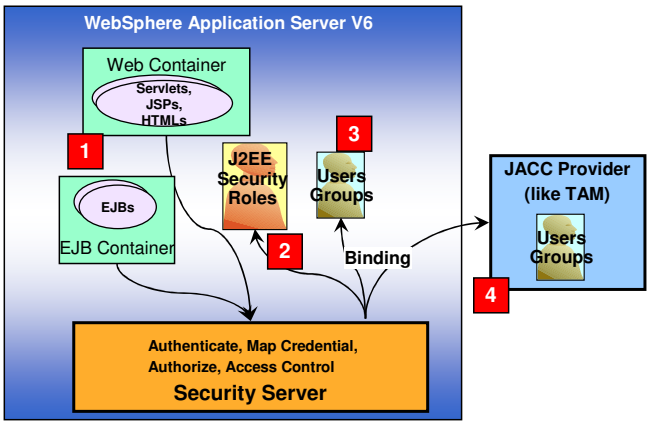
Section

Java Authorization Contract for Containers (JACC)

The next section will discuss Java Authorization Contract for Containers.

JACC: Overview

- JACC allows applications servers to interact with third party authorization providers using standard interfaces to make authorization decisions
  - The security policy, as well as the user or group bindings to the Security Role, are maintained by the JACC provider



Support of JACC based Authorization provider is in addition to the support of the Default Authorization binding, using the IBM Binding file for authorization information.
If using JACC, WebSphere Application Server V6 interacts with the JACC provider to update the binding during application install or modification of the binding information, post-install.

IBM's solution for a JACC provider is the IBM Tivoli Access Manager (TAM).
More details on JACC are provided in a separate presentation.

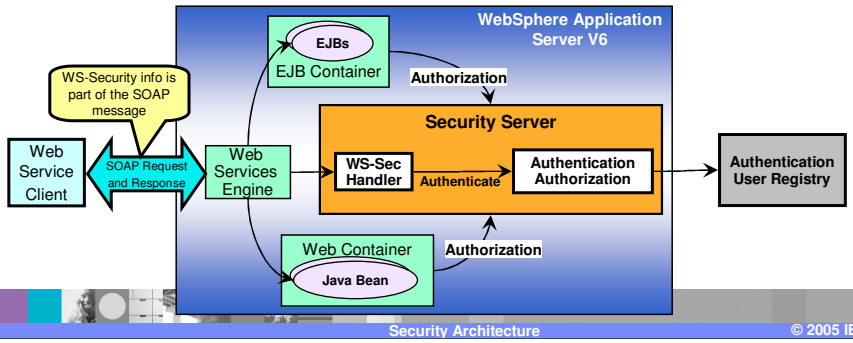
Section

Web Services Security (WS-Security)

The next section will discuss Web Services Security.

WS-Security: Overview

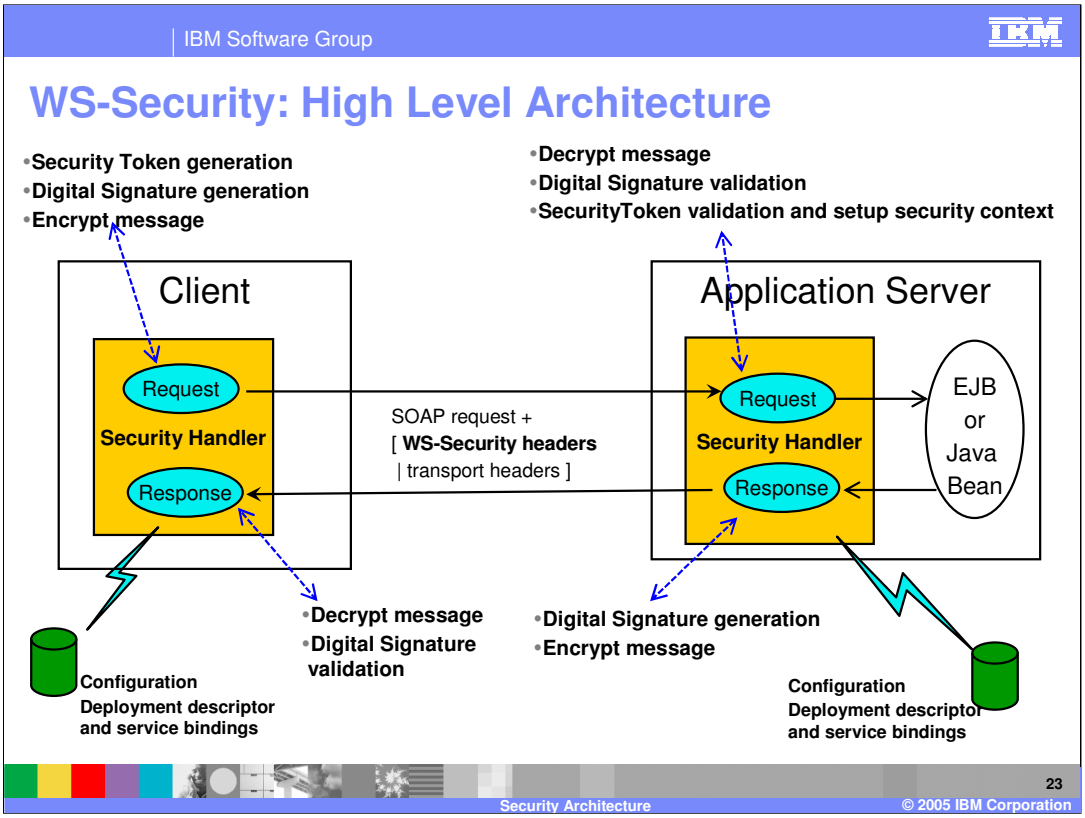
- WS-Security is a message level standard that defines how to secure SOAP messages, using
  - XML Digital Signature:
    - Digitally sign the SOAP XML document, providing integrity, authenticity, and signer authentication – JSR 105 to address this programmatically
  - XML Encryption:
    - Process for encrypting data and representing the result in XML providing confidentiality – JSR 106 to address this programmatically
  - XML Canonicalization:
    - Provides normalized XML document that can be digitally signed and verified
- Credential propagation through security tokens



WS-Security standardized security information gets embedded in the SOAP request and response message. The security information relates to authentication, integrity, confidentiality, timestamp and other security related information.

The picture shows the flow of the message.

When a request comes in the Web Services Engine working with the special system provided WS-Security handler to authenticate the message, decrypt and check for integrity, if required. Once the SOAP request is authenticated, the authorization is done using the J2EE Security roles for the Java bean or the EJB. In addition, the authentication uses the same Authentication mechanism and the user registry that has been defined for the Application Server.



The picture shows the high level WS-Security high level architecture on how WebSphere applies WS-Security information to the SOAP message.

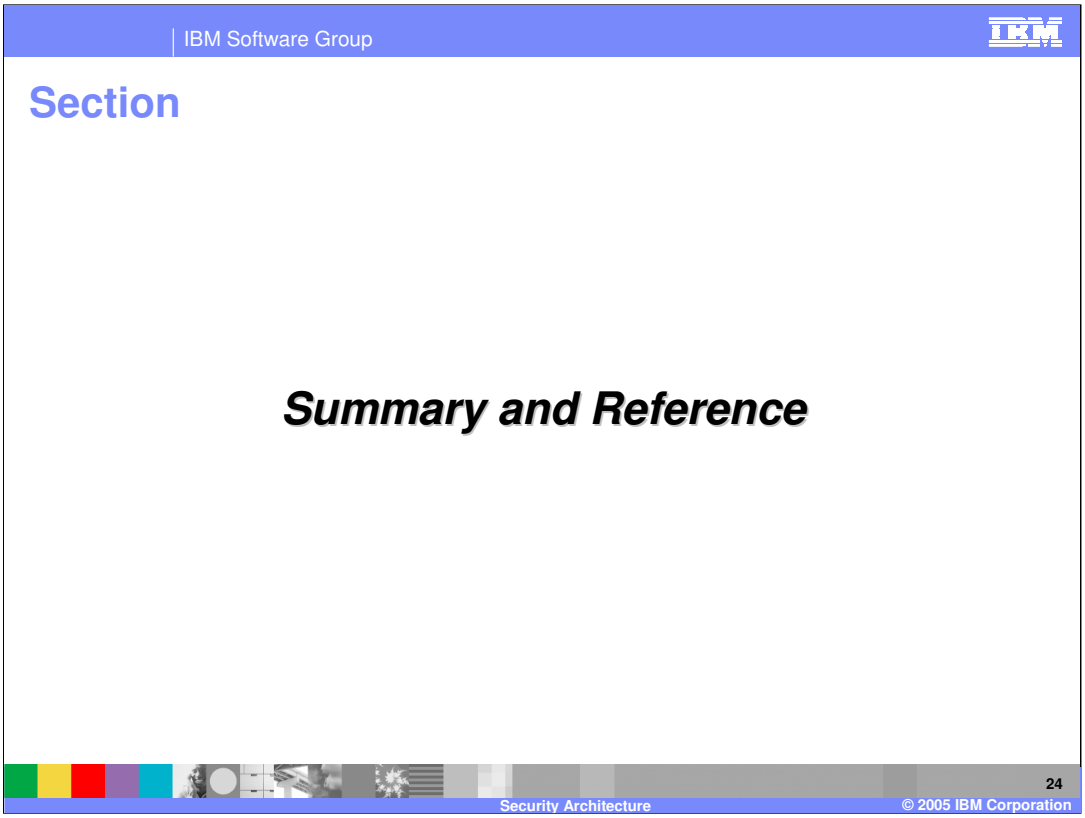
•WS-Security is designed and implemented as message level handlers of the Web Services engine, as a system handler. WS-Security handler is a “system handler” (called Security Handler in the foil) and is registered to the Web Service runtime

•On the client side (assuming WebSphere J2EE client), the WS-Security handler is invoked to generate the required security headers in the SOAP message before the message is sent out to the wire. The security handler generates the security constraint defined in the deployment descriptor and package the security information (digital signature, encrypted data and security tokens) in the SOAP message

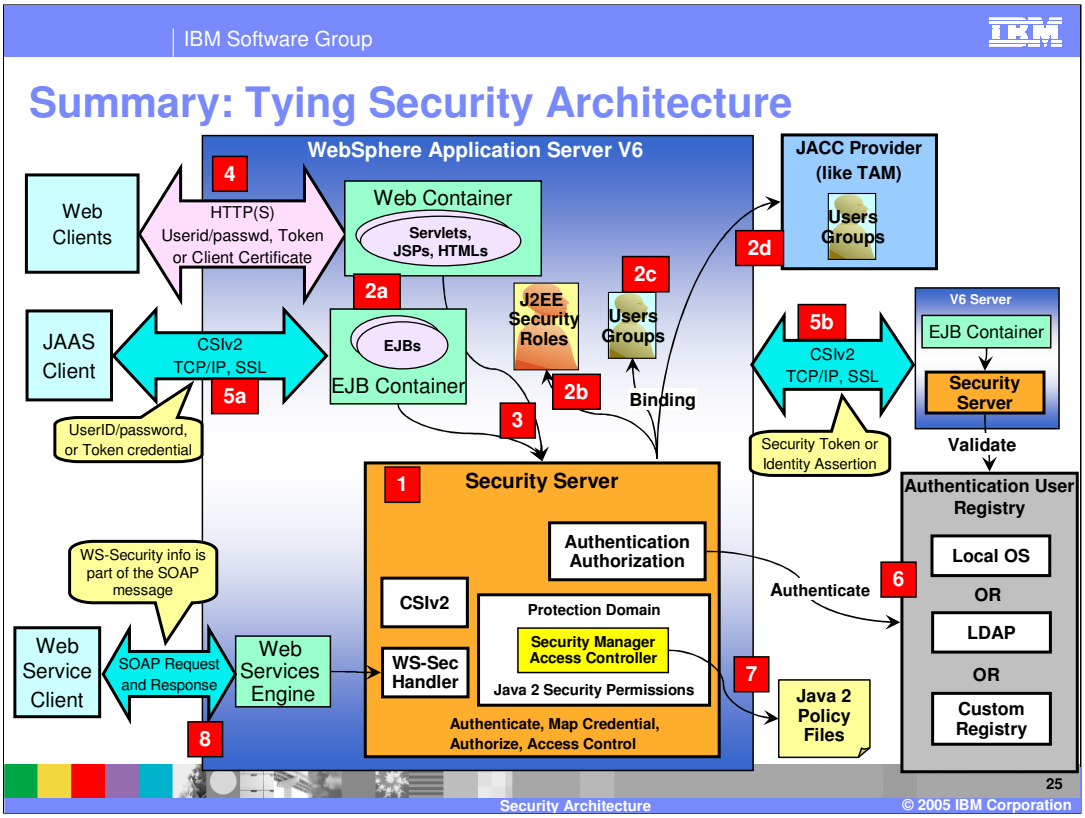
•On the server side, the WS-Security handler is called to enforce the declared security constraint in the deployment descriptor prior to dispatching the request to the Web Service EJB or Java Beans implementation.

•This is similar to security interceptors in the CSIv2 or SAS.

Note: The security constraints of request sender and request receiver must match. Also, the security constraints of the response sender and response receiver must match. For example, if you specify integrity as a constraint in the request receiver, then you must configure the request sender to have integrity applied to the SOAP message. Otherwise, the request is denied because the SOAP message does not include the integrity specified in the request constraint.



The next section will discuss a summary and reference to the aforementioned security methodologies.



This big Security picture ties all the components discussed so far in this presentation:

(1) Security Server that performs all security authentication, authorization and other security related functions like checking for Java 2 permissions

(2) Web and EJB applications (2a) with the J2EE Security Roles (2b) defined for the applications. J2EE Security Role to user/group binding, using the default binding within the Application Server (2c), or the 3rd party JACC provider (2d).

(3) Web Container and EJB container that interacts with the Security Server for authentication and authorization checks

(4) Web client requests directly or through a Web Server using HTTP or HTTPs

(5) External JAAS client (5b) calling EJBs within the Application Server, or EJB clients within the Application Server calling an EJB in another server (5b), using CSIv2 authentication protocol, making RMI-IIOP calls with or without SS

(6) Supported User Registry used for authentication

(7) Java 2 Security policy files used by the JVM Access Controller to check if the executing Java code has the necessary permissions to access the requested System resource

(8) Web Service client and the provider using WS-Security for the SOAP request and the response.

Summary and References

- This presentation covers Security components and functions at a high level
- Refer to other Security presentations for more details on many of the Security functions
- References:
  - WebSphere Application Server V6 Information Center
  - J2EE articles on the internet
    - Search for JAAS, CSIv2, J2EE Security

In summary, this presentation has focused on Authorization, Secure Socket Layer ,Java Authorization Contract for Containers, and Web Services Security. There are many other presentations that cover the details of Java 2 Security, J2EE Security, CSIv2, and so on.

Trademarks, Copyrights, and Disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	CICS	IMS	MQSeries	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

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

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

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

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

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.