



IBM Software Group

IBM® WebSphere® Application Server V6

Java™ 2 Security



@business on demand.

© 2005 IBM Corporation
Updated May 11, 2005

This presentation will focus on the Java 2 Security aspect of the J2EE Security architecture.

Goals

- Understand Java 2 Security in V6
- Understand IBM extensions for hierarchical policy files



The goal of this presentation is to help you understand the Java 2 Security component and IBM extensions. An understanding of the WebSphere Application Server V6 Security Architecture, which is covered in a separate presentation, will help you better understand this module.

Section

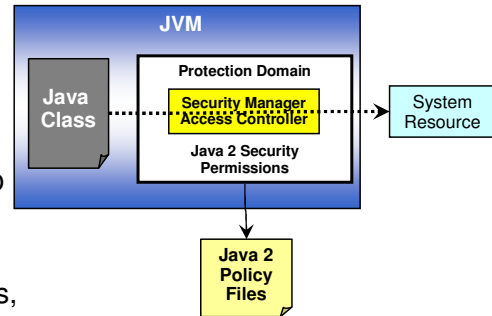
Java 2 Security



This section will cover Java 2 Security.

Java 2 Security

- Provides a policy-based, fine-grain access control mechanism to control the Java (application) code access to System level protected resources
 - ▶ Examples: File I/O, Network Connections (Sockets), Property files, etc.
- Policies, defined in policy files, are a set of permissions available for code from various signers or code location
- All Java code runs under a security policy that grants the code access to certain resources



- Java code needs access to certain System Resources
- Java code will need to get the permission from Java 2 Access Control
- Access Control looks at the Java 2 policy file(s) to determine if the requesting Java code has the appropriate permission



The purpose of Java 2 security is to prevent application code from gaining access to System resources such as files, ports, and sockets in a malicious or unintended manner.

If your application code must access System resources, you have to grant the necessary permission by means of a policy file. WebSphere Application Server supports a hierarchical definition of policy files at various levels, including application and server level, for code running within the Application Server.

Before allowing code to access a System resource, the access control mechanism of the JVM will read the policy files to determine if the correct permission is provided. If it is not, then no access will be given.

Example: Permissions Protected by Java 2 Security

File Access

- ▶ canRead(), FileInputStream(), RandomAccessFile(), isDirectory(), isFile(), length(),
- ▶ canWrite(), FileOutputStream(), mkdir(), renameTo(), createTempFile()
- ▶ delete(), deleteOnExit()

Network Access

- ▶ send(), receive(), getLocalAddress(), getHostName(), getLocalHost(), getAllByName()

Java VM

- ▶ ClassLoader(), loadLibrary(), checkPermission(), checkLink(), checkExit()

Program Threads

- ▶ stop(), resume(), suspend(), interrupt(), setPriority(), setName(), setDaemon()

System Resources

- ▶ getPrintJob(), setProperty(), getProperty(), setDefault(), getFont(), getEventQueue()

Security Aspects

- ▶ getFields(), getMethods(), getConstructors(), setPublicKey(), addCertificate()



Here are some examples of System Resources that are protected by Java 2 Security. You must explicitly grant permission to the code or the user executing the code in order to access these resources. This list of System Resources is not comprehensive.

Policy File Syntax

Policy file syntax

```
keystore "URL" "keystore-type"
grant signedby "signer" codebase <Codebase URL>
  {permission "class_name", "target" "action" ;
  permission "class_name", "target" "action" ;
  ...
};
```

where:

- Keystore file is the public key used to identify the signer
- signedby – name of the signer
- codebase - granting permission for code located in URL
- permission to access "class_name" to allow perform "action" on the "target" "action"

▪ Example 1:

```
grant codeBase "file:/home/MyProgram/" {
  permission java.io.FilePermission "/tmp/log.txt", "read, write"; };
→ Grants Java code located in directory :/home/MyProgram I/O permission to read/write /tmp/log.txt file
```

▪ Example 2:

```
grant codebase "http://www.MyURL.com/" {
  permission java.net.SocketPermission "9.53.129.2", "connect, accept"; }
→ Grants Java code from http://www.MyURL.com to accept/connect sockets to 9.53.129.2
```



Java 2 Security permissions are granted through the use of policy files. Policy files must adhere to a strict format. Any formatting errors in the policy file will prevent it from being processed.

The syntax of the policy file is shown on this slide.

Permission can be granted by codebase, specifying the location of the Java code that will access the resources specified in the permissions, or by the signer of the code.

In the first example, I/O read and write permission on the log.txt file is granted in the policy file for all code running from the specified directory.

In the second example, permission is granted for code from the specified URL to connect and accept sockets to the IP address.

Without these permission in the policy file, attempts to access the resources by the Java code would fail with a missing Java 2 security Access control exception. Messages are written to the Application Server log files.

Policy Files in WebSphere

	Policy File	Default Location	Description
STATIC Policies	java.policy	<PROFILE_HOME>/java/jre/lib/security/java.policy	Default permissions granted to all classes
	server.policy	<PROFILE_HOME>/properties/server.policy	Default permissions granted to all the product servers.
	client.policy	<PROFILE_HOME>/properties/client.policy	Default permissions for all of the product client containers and applets on a node
DYNAMIC Policies	filter.policy	<PROFILE_HOME>/ config/cells/cell_name	Filters OUT policy that are specified in other policy files – this allows System administrator to provide protection globally, even if other policy files may give the permission
	spi.policy	<PROFILE_HOME>/ config/cells/cell_name /nodes/node_name/spi.policy	For the Service Provider Interface (SPI) or 3 rd party resources embedded in the product.
	library.policy	<PROFILE_HOME>/ config/cells/cell_name /nodes/node_name/library.policy	For the shared libraries (Java library classes) used by applications. Default is empty.
	app.policy	<PROFILE_HOME>/ config/cells/cell_name	Default permissions granted to all J2EE applications running on a specific node
	ra.xml	rar_file_name/META-INF/was.policy.RAR	Default permission for the specific Resource Adapter, embedded in the RAR file
	was.policy	Within each application – policies provided by developer	Describes the policy for that application – if none provided, a default one will be created

There are several hierarchical policy files that can be specified in the JVM and WebSphere Application Server that provide permissions for different code bases.

Both static and dynamic policy files are supported. Static policy files provide the default permissions, while dynamic policy files provide application permissions. You can use variables to define codebase locations in the Dynamic Policy files.

Valid Symbols used in Dynamic policy files

Valid symbols in Dynamic Policy file	Description
file:\${application}	Permissions apply to all resources within the application
file:\${jars}	Permissions apply to all utility Java archive (JAR) files within the application
file:\${ejbComponent}	Permissions apply to EJB resources within the application
file:\${webComponent}	Permissions apply to Web resources within the application
file:\${connectorComponent}	Permissions apply to connector resources within the application

Example

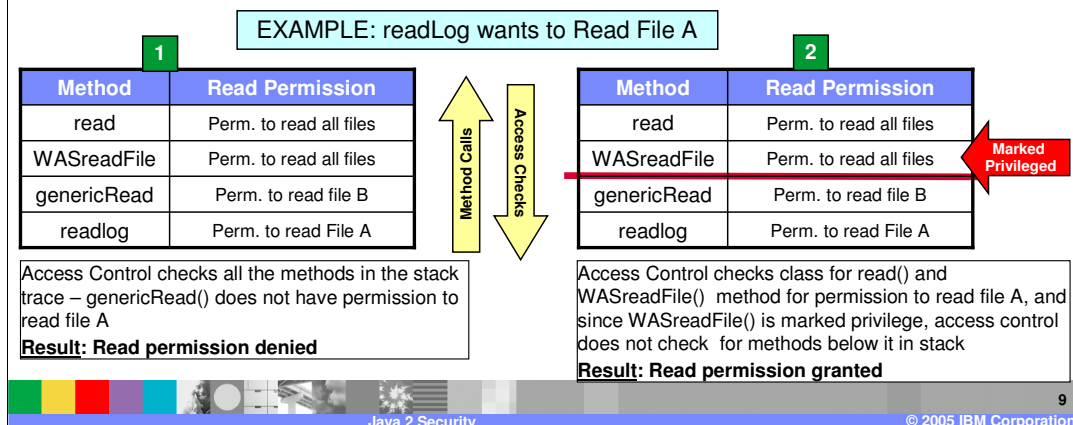
```
grant codeBase "file:${application}" { permission java.io.FilePermission
    "${user.install.root}${/}bin${/}DefaultDB${/}-", "read,write,delete"; }
```

→ Grants Java code in "IncCMP11.jar" file I/O permission to read/write files in WebSphere install root bin/DefaultDB directory

Shown here is a list of symbols that can be used in a dynamic policy file to grant permission based on resource type. The example demonstrates how these variables can be used in a policy file to grant permissions

Java 2 Security: Access Controller

- The access controller is the mechanism used by the security manager to enforce the protection
- The Access controller walks down the Call stack trace to check if right permission is given for the code in the call
- The walkthrough occurs until end of stack trace or caller is marked as privileged (with a doPrivileged)
- Marking as privileged allow callers to perform certain system functions without the need of granting them permissions



The two methods of providing access control are shown here.

In the first example, all the classes or callers in the call stack trace must have the appropriate Java 2 permission before any of them are allowed to access the resource.

In the second example, a caller is marked as *privileged*, in the call stack. The callers below the privileged caller in the call stack are not checked for permissions. Only the callers above the privileged caller would require the permission.

Now consider two scenarios, one with no caller marked as privileged, and one with one of the callers marked as privileged.

In the first scenario, before the read() method is allowed to read a file, the Access controller checks to see if the read() method has the appropriate permission. If it does, it moves to the next caller in the call stack trace. If it does not, it throws an exception for every call in the call stack trace.

In this scenario, the check starts with the read() method, followed by WASreadFile() and down the stack. When it comes to genericRead(), access will be denied because that method does not have permission to read file A.

In the second scenario, if a caller is marked as privileged and has the appropriate permission, no further checking of the callers lower in the call stack trace is done.

Java 2 Security: Filters

- Contains the list of permissions that require filtering from certain policy files in the cell - Enables selective disabling of Java 2 security
 - ▶ Provides sort of "filtering" for was.policy
 - ▶ Warn if custom defined permissions (not java.* or javax.*)
 - ▶ Warn if java.security.AllPermission
- This filtering mechanism only applies to the was.policy and app.policy files
- Default WebSphere policy file:

```
filterMask {  
  permission java.lang.RuntimePermission "exitVM";  
  permission java.lang.RuntimePermission "createSecurityManager";  
  permission java.lang.RuntimePermission "setSecurityManager";  
};
```
- **The effective Application policy becomes:
app.policy + was.policy + java.policy "-" filter.policy**



Filters ensure that application developers do not grant applications permission to do things that could be dangerous, such as exit the JVM or create their own Security Manager and implement their own security. Even if the application grants permission, permissions specified in the filter.policy file will be disabled.

Enabling Java 2 Security

- Java 2 Security enabled when Global Security is turned on
 - ▶ Java 2 Security can be manually disabled, while keeping Global Security on
 - ▶ Test existing applications before turning on Java 2 Security
- Adding Java 2 Security to installed applications
 - ▶ For individual applications, the policy file, "**was.policy**", is placed in the META-INF folder of the Enterprise Application
 - ▶ Default one created during install process, if one is not already present
- Adding/modifying was.policy
 - ▶ Add was.policy file to EAR using AST or Rational® tools and reinstall
 - Recommended and less error prone
 - ▶ Place the was.policy file in the appropriate location in the file system where the application was installed
 - <WAS_PROFILE_DIR>\config\application\<application>.ear\META-INF\
- Tool for creating/modify policy files: **policyTool** – Part of JDK, in java/jre/bin/policytool
 - ▶ Advised to use the tool rather than editing by hand



Java 2 Security is optional and can only be enabled if Global security is first enabled.

It is recommended that you use the policyTool to edit the policy files to ensure that the strict format is maintained. If a hand edit of the server.policy file introduces an error, it will prevent the server from starting.

Section

Summary and Reference

This section will provide a summary of this presentation.

Summary

- Java 2 Security provides mechanism to specify authorization for the Java code to access System resources
- This prevents application code with ill intent to access protected System resources
- Java 2 security is specified using policy files
- There are several different policy files that provide layered protection
- WebSphere Application Server V6 provides additional capability by allowing you to specify dynamic policy files



In summary, Java 2 security provides access control on System resources for your application code. WebSphere Application Server V6 adds additional capability with the dynamic policy files.

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.

© Copyright International Business Machines Corporation 2004. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.