JavaOne™

ORACLE®

Security Policy File Best Practices

For Your Java/JDBC Modules

Ilesh Garish, PMTS, Oracle
Douglas Surber, CMTS, Oracle
Kuassi Mensah, Director, PM, Oracle
Oct 02, 2017

JavaOne™

JavaYourNext
(Cloud)

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Program Agenda

1. Java Security Manager
2. Application Servers
3. Policy File
4. Detailed Example JDBC
5. Q&A

JavaOne ORACLE

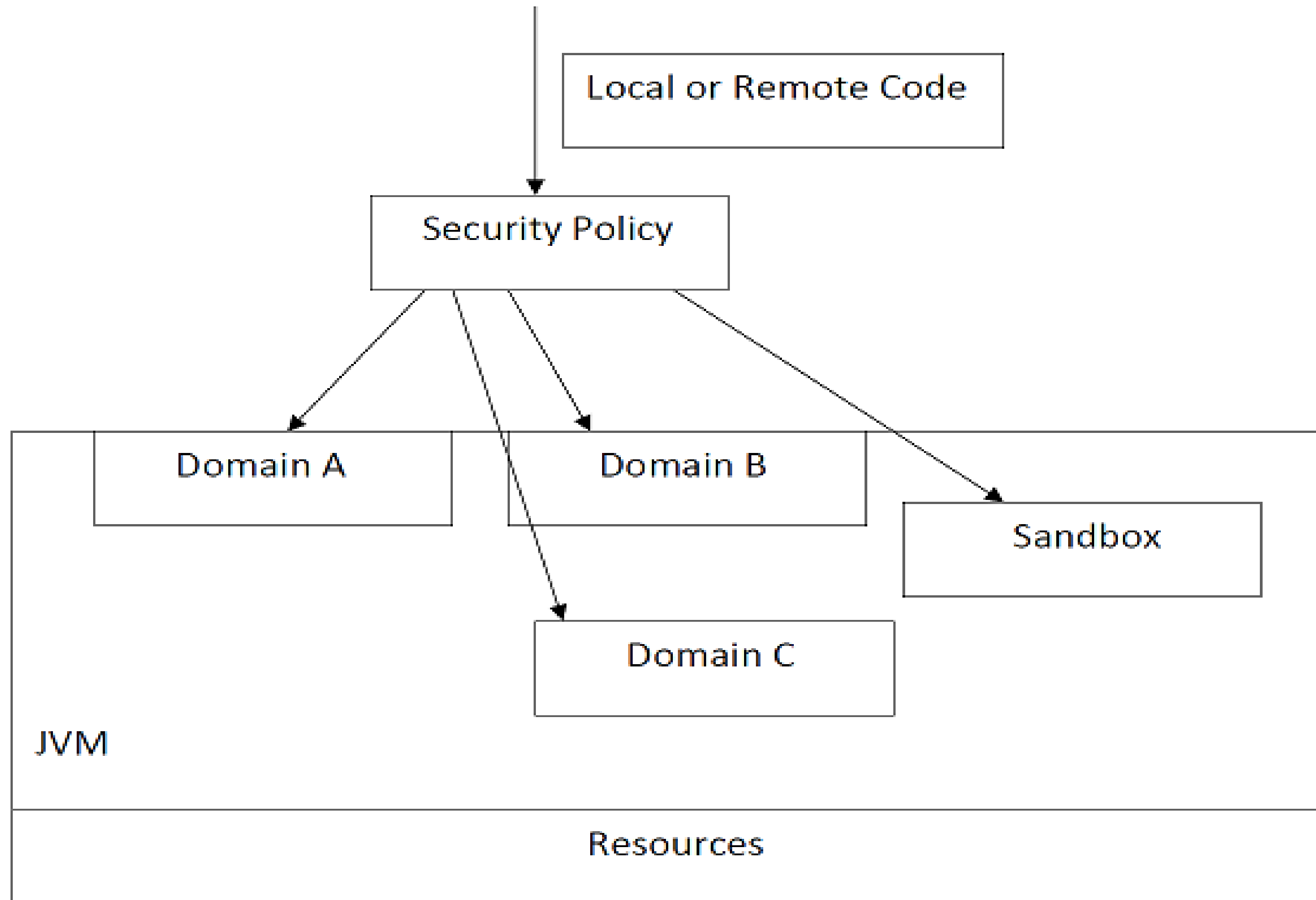
Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Java Security Manager

JavaOne ORACLE

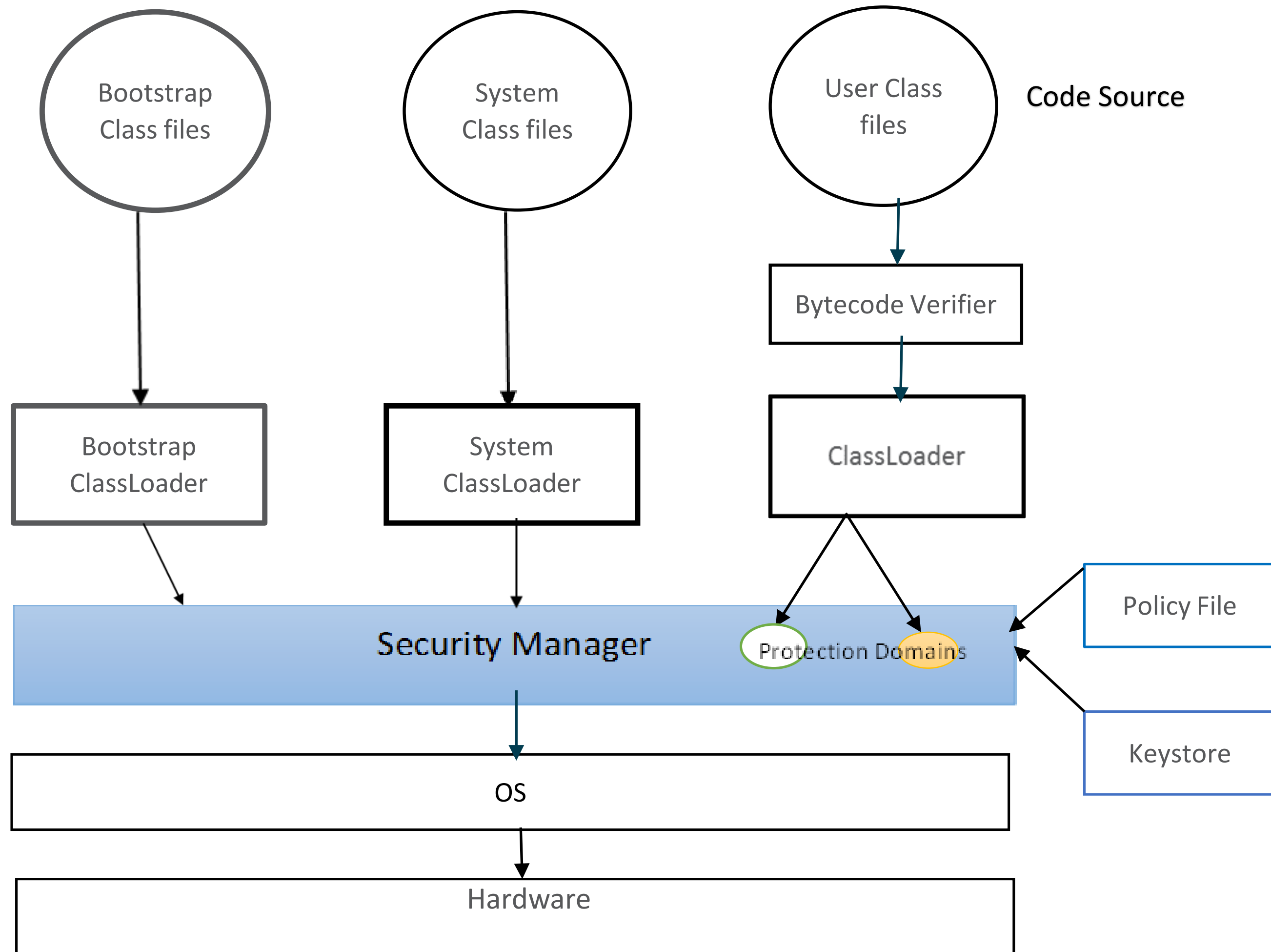
Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Security Model



Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Security Model



JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Java Security Manager

- Provides additional protection for resources running in a JVM.
- Prevents untrusted code from performing actions that are restricted by the Java security policy file.
- The Java Security Manager uses the Java security policy file to enforce a set of permissions granted to classes.
- The permissions allow specified classes to permit or not permit certain runtime operations.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Protection Domains

- Set of objects that are currently directly accessible by a principal.
- Principal is an entity in the computer system to which permissions are granted.
- Serves to group and to isolate between units of protection.

JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Application Servers

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Web Logic Server

ORACLE
FUSION MIDDLEWARE
WEBLOGIC SERVER

- Specify the `-Djava.security.policy` and `-Djava.security.manager` arguments when starting WebLogic Server.
- WLS provides a sample Java security policy file located at `$WL_HOME/wlserver/server/lib/weblogic.policy`
- If you do not specify a security policy file, the default security policies defined in the `java.policy` file in the `$JAVA_HOME/jre/lib/security` directory

JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Tomcat

- Configure the `catalina.policy` file for use with a Security Manager
- Start a Security Manager in place by using the `"-security"` option:

```
$CATALINA_HOME/bin/catalina.sh start -security   (Unix)
 %CATALINA_HOME%\bin\catalina start -security    (Windows)
```


Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

JBoss

- Specify the `-Djava.security.policy` and `-Djava.security.manager` arguments in `$JBOSS_HOME/server/$PROFILE/run.conf` when starting JBoss.
- JBoss provides a sample Java security policy file located at `$JBOSS_HOME/bin/server.policy.cert`
- You can overrirde default security policy file using == operator while setting `-Djava.security.policy`.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

WebSphere

- Open the WebSphere Application Server Administrative Console.
- Select and expand "Security" on the left navigation pane.
- Click on Global security.
- Check the checkbox for "Enforce Java 2 security".
- Click "Apply".
- Click "Save" to save your workspace changes to the master configuration.
  - The server may need to be restarted for the changes to take effect.
- java.policy – `app_server_root/java/jre/lib/security/java.policy`

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Policy File

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Policy File Location

- The system policy file `java.home/lib/security/java.policy`
- The user policy file `user.home/.java.policy`
- Policy file locations are specified in the security properties file

  `java.home/lib/security/java.security`

  Property Name:

  – policy.url.*n*=*URL*

  – policy.provider=sun.security.provider.PolicyFile
- -Djava.security.policy=mypolicy
  -Djava.security.policy==*someURL*
  policy.allowSystemProperty=false then it ignores. Default is true.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

JSM Assign Permissions

- Every class is associated with a *protection domain*, which consists of:
  - A link to the defining class loader of the class
  - A *code source* (a URL that identifies the code)
  - A *permission collection* (a set of assigned permissions)
  - An array of principals, specific to JAAS
- The protection domain permissions are built from the security policy
- Limited doPrivileged - run privileged actions with only a subset of your total possible permissions (Java 8)

JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

JSM Checking Permissions

- The security manager examines the current *access control context*
- This is the accumulation of the protection domains of classes from the call stack
  - Stopping only at the last call to AccessController.doPrivileged()
  - The access control context is inherited to newly created threads
- The permission check passes only if *all* the security domains in the access control context have granted the permission

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Setup Policy File

- A policy file is an ASCII text file and can be composed via
  - A text editor
  - The Graphical Policy Tool utility
- The Policy Tool saves you typing and eliminates the need for you to know the required syntax of policy files, thus reducing errors
- To start Policy Tool, simply type the following at the command line:
  - `policytool`

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Policy File Syntax

- A policy configuration file essentially contains a "keystore" entry, and contains zero or more "grant" entries

- A *keystore* is a database of private keys and their associated digital certificates such as X.509 certificate chains authenticating the corresponding public keys.

- The **keytool** utility is used to create and administer keystores.

- Look up the public keys of the signers specified in the grant entries of the file.
  - `keystore "some_keystore_url", "keystore_type", "keystore_provider";`
  - `keystorePasswordURL "some_password_url";`

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Policy File Syntax

- Grant Entries

- Each **grant entry** includes
  - one or more "permission entries“.
  - preceded by optional codeBase, signedBy, and principal name/value pairs

```
grant signedBy "signer_names", codeBase "URL",
principal principal_class_name "principal_name", ...
{permission permission_class_name "target_name", "action",
signedBy "signer_names“;... };
```

- A `signedBy` value indicates the alias for a certificate stored in the keystore.

JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Policy File Syntax

- The Principal Entries
- `principal` *`principal_class_name`* `"`*`principal_name`*`",`
- The `principal` set is associated with the executing code by way of a Subject.
- grant principal javax.security.auth.x500.X500Principal "cn=Mice" { permission java.io.FilePermission "/home/Mice", "read, write"; };

  This permits any code executing as the X500Principal, "cn=Mice", permission to read and write to "/home/Mice".

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Policy File Syntax

- The Permission Entries

  grant signedBy "*signer_names*", codeBase "*URL*", ... {
  permission *permission_class_name* "*target_name*",
  "*action*", signedBy "*signer_names*"; ... };

- The *permission_class_name* actually be a specific permission
  e.g. `java.io.FilePermission`, `java.lang.RuntimePermission`

- When you specify `java.io.FilePermission` then
  "`target_name`" is a file path

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Policy File Syntax

- The "*action*" is required for many permission types, such as `java.io.FilePermission`. e.g. “read”, “write” etc.
- The signedBy name/value pair for a permission entry is optional. If present, it indicates a signed permission.
  - e.g. suppose you have the following grant entry:

```
grant { permission Foo "foobar", signedBy "FooAbc"; };
```

Permission of type *Foo* is granted if the Foo.class permission was placed in a JAR file and the JAR file was signed by the certificate specified by the "FooAbc" alias.

JavaOne ORACLE
Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Policy File Syntax

- Items must appear in the specified order (permission, *permission_class_name*, `"`*`target_name`*`", "`*`action`*`", and signedBy "`*`signer_names`*`"`).
- An entry is terminated with a semicolon.
- Case is unimportant for the identifiers
  - e.g. permission, signedBy etc.
- Case is significant for the *permission_class_name*
- Property expansion is similar to expanding variables in a shell
  - `${some.property}`

JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Policy File Strategy

- Grant all permissions
  - permission `java.security.AllPermission`;
- Grant least permissions
  - Start with a blank policy file
  - Run the application
  - Check the thrown security exception
  - Add the smallest-grained permission possible in the policy file that fix exception
  - Repeat above 3 steps until application run without exception.
- Regularly review security policy files to accommodate any updates in application.

JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

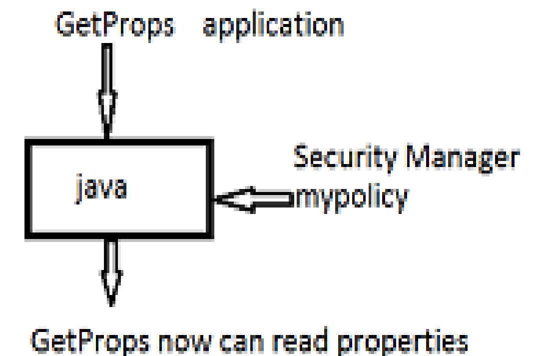
Review Policy Files

- Review all *.policy files
- Policy files precedence order
- Remove unused grants
- Add extra permissions to applications or modules that require them, not all applications.
- Document your changes
- Use –Djava.security.debug=all to troubleshoot security failures

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

JSM and Policy File

- The application program makes request for GetProps
- The Java API ask Security Manager if this is allowed.
- The Security Manager uses mypolicy file to check permissions.
- If disallows then Security Manager throws Security Exception, which Java API throws to Application.
- If allows then Java API completes operation and returns normally.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Permissions Outside of Policy File

- Virtually all of the permissions granted to code comes from policy files
- Advanced applications are allowed to grant additional permissions to code that they load
- Standard Java class loaders grant some additional permissions to every class that they load
  - File class loader always grant permission to read files in the directory
  - HTTP class loader always grant permission to establish a connection and to accept a connection from that host.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Security Policy API

- java.security.Policy implements
  - getPolicy() returns current policy.
    Needs SecurityPermission getPolicy

  - setPolicy() enables you to set a new system-wide security policy
    Needs SecurityPermission setPolicy

- New policy must extend the Policy class and implement the getPermissions() and refresh()

- Alternatively, the default policy implementation can be specified with a policy.provider property in the java.security file

  policy.provider=com.mycompany.security.GenericPolicy

JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Detailed Example

Oracle Database JDBC

JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Java SecurityManager

- Java does not enable a SecurityManager by default
  Java in the database always has a SecurityManager

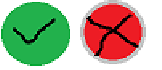
- Without a SecurityManager Java is not secure

- If you care about security, you must enable a SecurityManager:

  `-Djava.security.manager`

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Permissions



- If you enable a SecurityManager, you must grant Oracle JDBC certain permissions

```
-Djava.security.policy=ojdbc.policy
```

- If you don't, JDBC will not run at all

```
[java] java.security.AccessControlException: access denied
(javax.management.MBeanTrustPermission register)
```

- Must also grant user code some permissions

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Philosophy

- No privileged access to external resources
  - Black hat cannot use JDBC to bypass permissions required to access external resources
  - User code must have `java.net.socket permission` to open a Thin driver connection
  - Cannot use Thin driver for a DoS (denial-of-service) attack
- Protect critical internal resources
  - `oracle.jdbc.OracleConnection.abort()`
- “JDBC” stand for “Java Database Access”
  - no permissions required to access the database
- The database provides database security
  - user authentication

JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Default JDBC Policy File

- Default JDBC policy file is in the JDBC download

```
tar -xf demo.tar ojdbc.policy
```

- ojdbc.policy is parameterized. You must define certain system properties to use it.

```
-Doracle.jdbc.policy.JDBC_CODE_BASE=$ORACLE_HOME/lib/ojdbc8.jar
```

- Only define the system properties required by the JDBC features you use

JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Best Practice

- Create your own policy file by editing ojdbc.policy
  - Delete or comment out what you don't need
- Whether to use macros or hard code paths depends on your deployment environment
  - If it is easier to redefine system properties then use the macros
  - If it is easier to edit the policy file then hard code values.
  - Editing the policy file may be marginally less secure

JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

ojdbc.policy file

- In order to use this policy file as is, define the following system properties:

- Grant codebase
  - `oracle.jdbc.policy.JDBC_CODE_BASE=$ORACLE_HOM/lib/ojdbc8.jar`

- SocketPermission
  - `oracle.jdbc.policy.CLIENT_HOST=client.myco.com`
  - `oracle.jdbc.policy.DBMS_HOST=db.myco.com`
  - `oracle.jdbc.policy.DBMS_PORT=1521`

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

ojdbc.policy file

- Always needed PropertyPermissions
  - permission `java.util.PropertyPermission "user.name", "read";`
  - permission `java.util.PropertyPermission "oracle.jdbc.*", "read";`
  - permission `java.util.PropertyPermission "oracle.net.*", "read";`
  - permission `java.util.PropertyPermission "javax.net.ssl.*", "read";`

- Needed RuntimePermission for OCI driver
  - permission `java.lang.RuntimePermission "loadLibrary.ocijdbc12";`

- Needed only if you need orai18n.jar for NLS
  - permission `java.io.FilePermission "${oracle.jdbc.policy.ORAI18N}", "read";`

JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

ojdbc.policy file

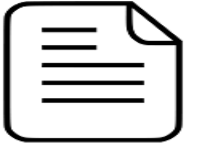
- Needed only if you use Fast Connection Failover
  - permission `oracle.ons.CreatePermission "ONSUser";`
  - permission `oracle.ons.SubscribePermission "ONSUser";`
  - permission `java.io.FilePermission "${oracle.jdbc.policy.OPMN_CONFIG}", "read";`
  - permission `java.util.PropertyPermission "oracle.ons.*", "read";`

- Remote ONS host:port
  - permission `java.net.SocketPermission "${oracle.jdbc.policy.REMOTE_ONS_HOST1}:${oracle.jdbc.policy.REMOTE_ONS_PORT1}", "connect,resolve";`
  - permission `java.netSocketPermission "localhost", "connect,resolve";`

- Many permissions based on feature in use such as DMS, XA, XDB, AQ, NLS etc.

JavaOne ORACLE
Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

ojdbc.policy



```
/*  Copyright Oracle 2007, 2017
  In order to use this policy file as is, define the following system properties

  -Doracle.jdbc.policy.CLIENT_HOST=client.myco.com
  -Doracle.jdbc.policy.DBMS_HOST=db.myco.com
  -Doracle.jdbc.policy.DBMS_PORT=1521
  -Doracle.jdbc.policy.DMS_CODE_BASE=$ORACLE_HOME/lib/dms.jar

  -Doracle.jdbc.policy.JDBC_CODE_BASE=$ORACLE_HOME/lib/ojdbc??.jar for the jar used

  -Doracle.jdbc.policy.ORAI18N=$ORACLE_HOME/lib/orai18n.jar
  -Doracle.jdbc.policy.USER_CODE_BASE=/home/myapp/lib/-
  -Doracle.jdbc.policy.ONS_CODE_BASE=$ORACLE_HOME/opmn/lib/ons.jar
  -Doracle.jdbc.policy.CONNECTION_POOL_CODE_BASE=myConnectionPool.jar
  -Doracle.jdbc.policy.OPMN_CONFIG=$ORACLE_HOME/opmn/conf/*
  -Doracle.jdbc.policy.REMOTE_ONS_HOST1=db.myco.com
  -Doracle.jdbc.policy.REMOTE_ONS_PORT1=4200

  Of course you can also edit this file and replace the macros (stuff enclosed in ${foo} )
  with the actual values. You can add multiple SocketPermissions to support multiple hosts
  and clients for the Thin driver or you can use wildcards (*). If you don't use DMS you don't
  need to define the DMS system properties.
*/

grant codeBase "file:${oracle.jdbc.policy.JDBC_CODE_BASE}" {
```

JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Performance

- May be small performance penalty
- It only applies when application attempts some activity that requires permission checks.
- Most operations that requires permission checks are expensive operations (IO, Network access etc.), so overhead of security checks will be pretty low percentage of total runtime.
- Benchmark your application

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Resources

- Policy Syntax File:
  - http://docs.oracle.com/javase/8/docs/technotes/guides/security/PolicyFiles.html#FileSyntax
- Permissions in JDK:
  - http://docs.oracle.com/javase/8/docs/technotes/guides/security/permissions.html
- JDBC Download:
  - http://www.oracle.com/technetwork/database/features/jdbc/jdbc-ucp-122-3110062.html

JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Q&A

JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

JavaOne™

ORACLE®