
ORACLE

JavaOne™

ORACLE

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.

Security with Java Deployment

Chris Bensen
Principal Member of Technical Staff

Java Client Deployment and Performance
September 29, 2014

JavaOne ORACLE

Program Agenda

1. Overview
2. Security Changes
3. New Features
4. Best Practices
5. Packager
5. Future

JavaOne ORACLE

Overview

Overview

- Security Levels
- Java Control Panel (JCP) option to disable Java in the browser
- Expiring JRE
- Security Baseline
- Local applets blocked
- Applet Sandboxing with Safari 6.1 and 7
- JNLP auto-download support removed
- Deployment Toolkit (DT) auto-download for secure Java versions

Security Changes

Security Changes

- Security Levels
  - Medium Security Level has been removed JDK 8u20

Security Changes

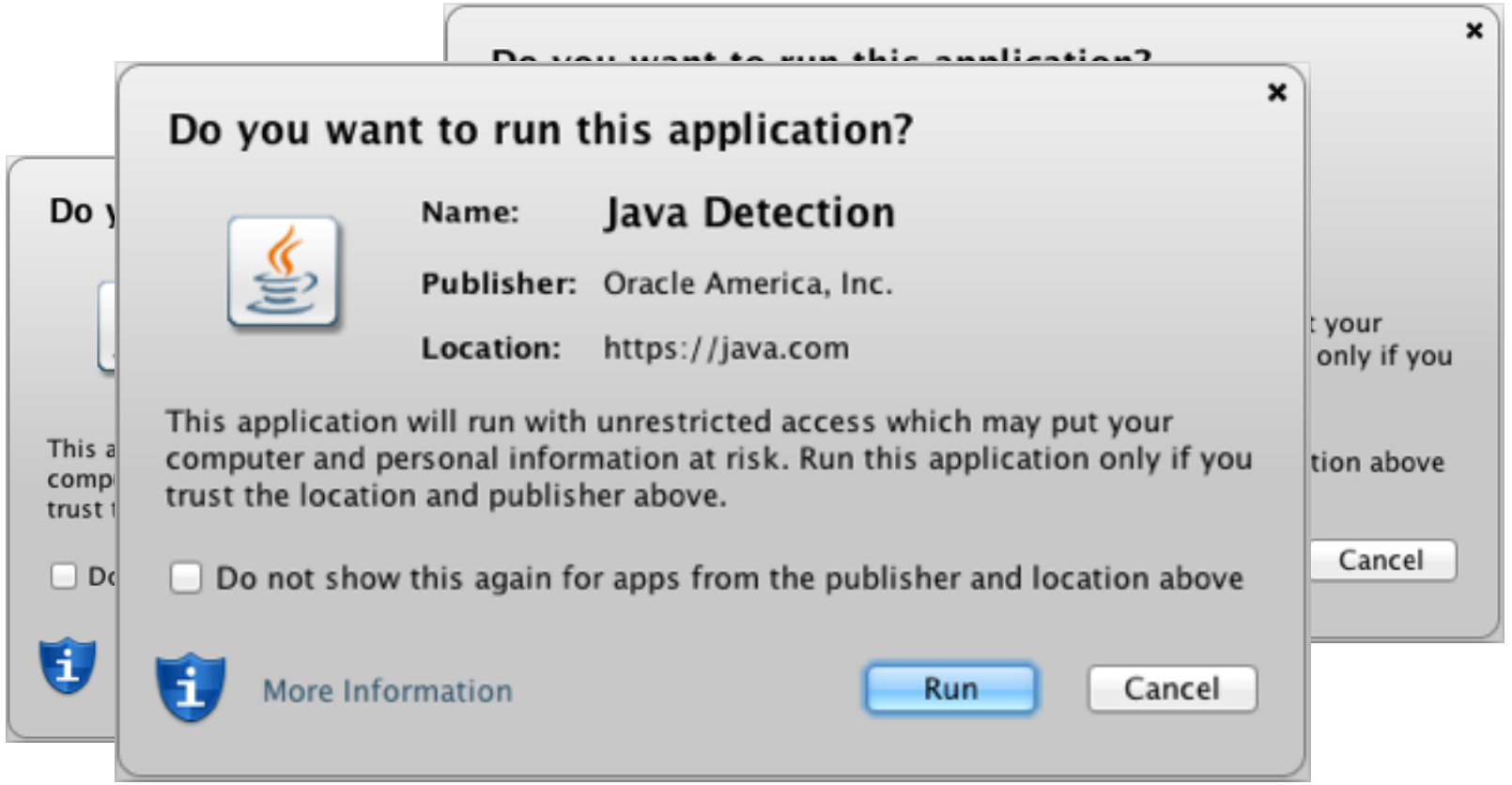
- Security Levels
  - High
    - All JARs must be signed
    - Main JAR must have Permission Attribute
    - Only exceptions are when app is covered by DRS or ESL
  - Very High
    - All features of High
    - Revocation server cannot be reached, Very High will block

Security Changes

- Restore Security Prompts
  - Added in JDK 7u45/8
  - Prompted by Installer

Security Changes

- Reduced Dialog Frequency JDK 7u55/8
  - We heard you, now with fewer dialogs



Security Changes

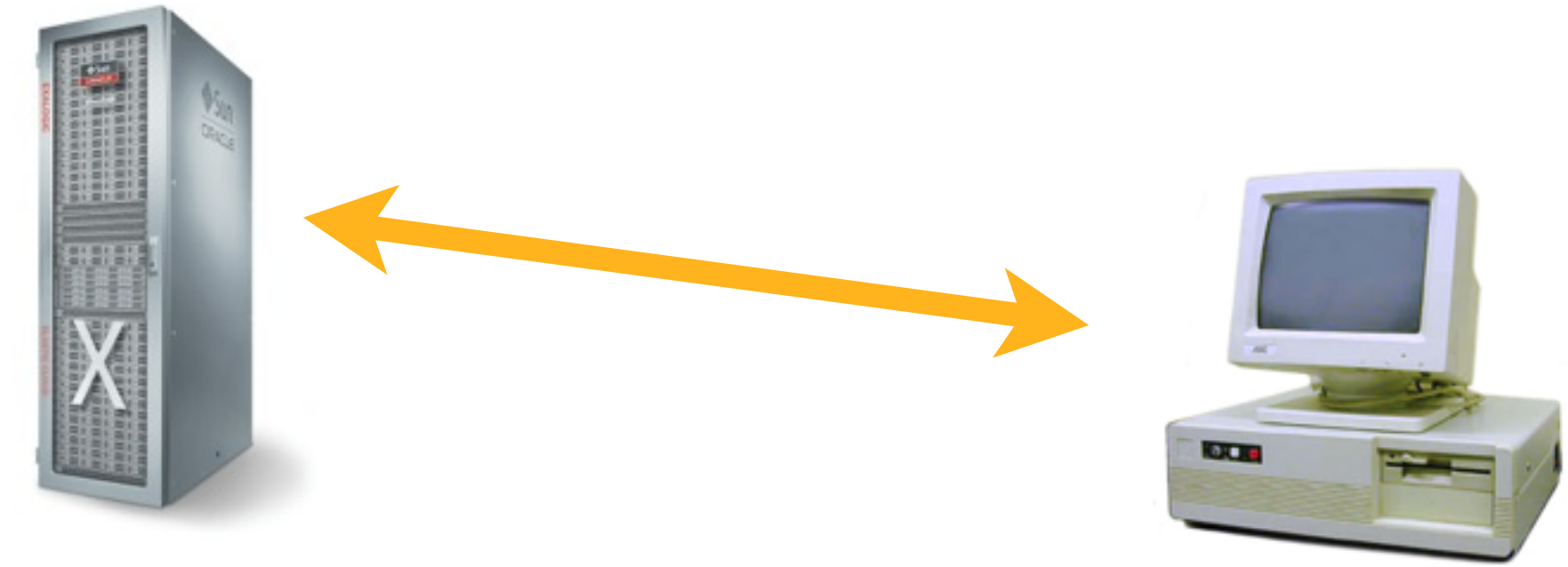
- JavaScript to Java (Liveconnect)
  - Can be blocked or allowed by:
    - Caller-Allowable-Codebase (CAC)
      - DRS rule cannot override Caller-Allowable-Codebase attributes
    - Deployment Rule Set (DRS)
    - Exception Site List (ESL)
    - If not blocked or allowed by above user can allow or block
  - If App and HTML are on different hosts
    - Specific rule is required

Security Changes

- Dynamic Blacklist Changes
  - Blacklist was only used for signing certificates
  - Secure Sockets Layer (SSL) Certificates
  - Transport Layer Security (TLS) Certificates
  - Blacklist automatically updates from java.com
  - Oracle can add any SSL/TLS Certificate
  - Preemptive for Heartbleed exploit

Security Changes

- JDK Same Origin Policy (SOP)



Security Changes

- JDK Same Origin Policy (SOP)

Security Changes

- JDK Same Origin Policy (SOP)

Security Changes

- JDK Same Origin Policy (SOP)

Security Changes

- JDK Same Origin Policy (SOP)
  - Was using IP address
  - Now uses hostname
  - Virtual hosting
  - Cloud services
  - JDK 8 only

New Features

New Features

- Manifest Attributes
  - NEW! Entry-point
  - NEW! Caller-allowable-codebase
  - NEW! Application-library-allowable-codebase
  - Main-Class
  - Permissions
  - Codebase
  - Application-Name
  - Application-Library-Allowable-Codebase
  - Trusted-Library
  - Trusted-Only

New Features

- NEW! Entry-Point
  - Manage public/private state of main classes in JAR
  - Entry-Point is an optional attribute
  - Manifest Example:

```
Main-Class: package.Example
Entry-Point: package.Example package.Example2
```

New Features

- NEW! Caller-allowable-codebase
  - Prohibit a trusted JAR from being reused on other domains
  - Example 1
    - Caller-Allowable-Codebase: *
  - Example 2
    - Caller-Allowable-Codebase: *.com *.org
  - Example 3
    - Caller-Allowable-Codebase: https://www.example.com
  - Example 4
    - Caller-Allowable-Codebase: https://*.example.com:443

New Features

- NEW! Application-library-allowable-codebase
  - List of hosts that match the specified patterns
    - protocol
    - host
    - port
  - Any jnlp extensions (or non-main JARS in html applets) must come from listed places

New Features

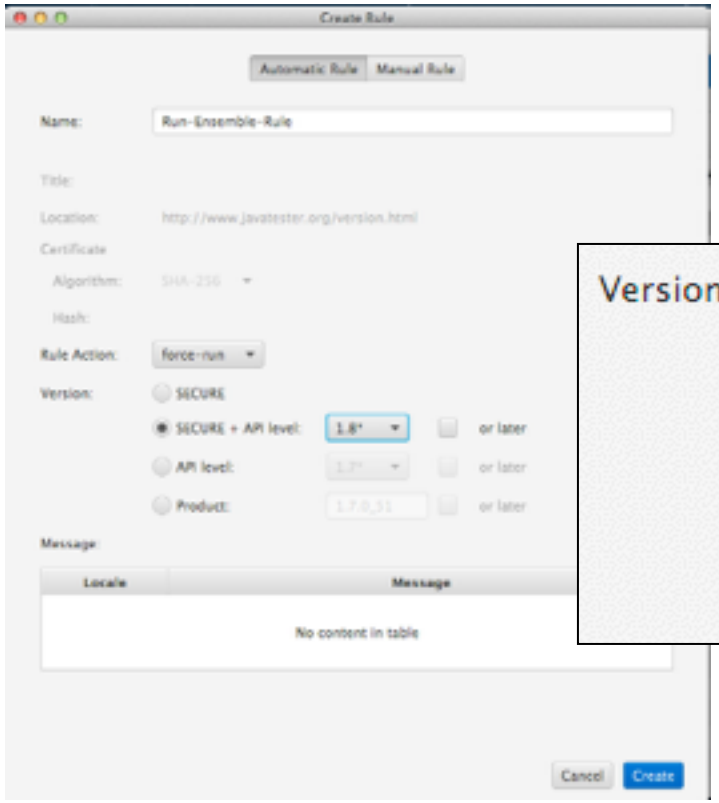
- NEW! Application-library-allowable-codebase
  - Example 1
    - Application-Library-Allowable-Codebase: *.example.com
  - Example 2
    - Application-Library-Allowable-Codebase: https://www.example.com
  - Example 3
  - Application-Library-Allowable-Codebase: *.example.com:1080

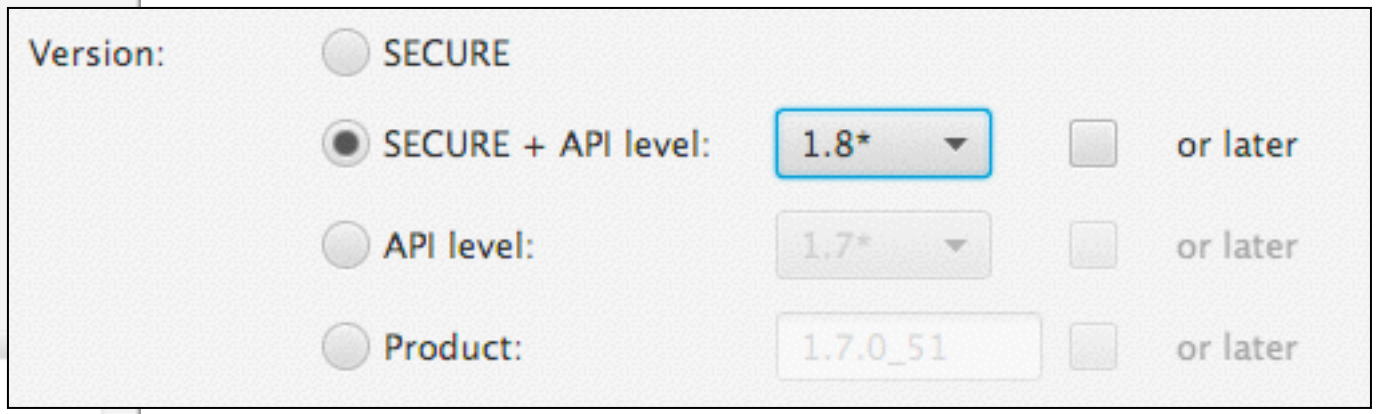
New Features

- Advanced Management Console (AMC)
  - Deployment Rule Set (DRS)
    - force attribute
    - Specify a specific version of the JRE to use
  - AMC exposes force attribute as force-run feature

New Features

- Advanced Management Console (AMC)





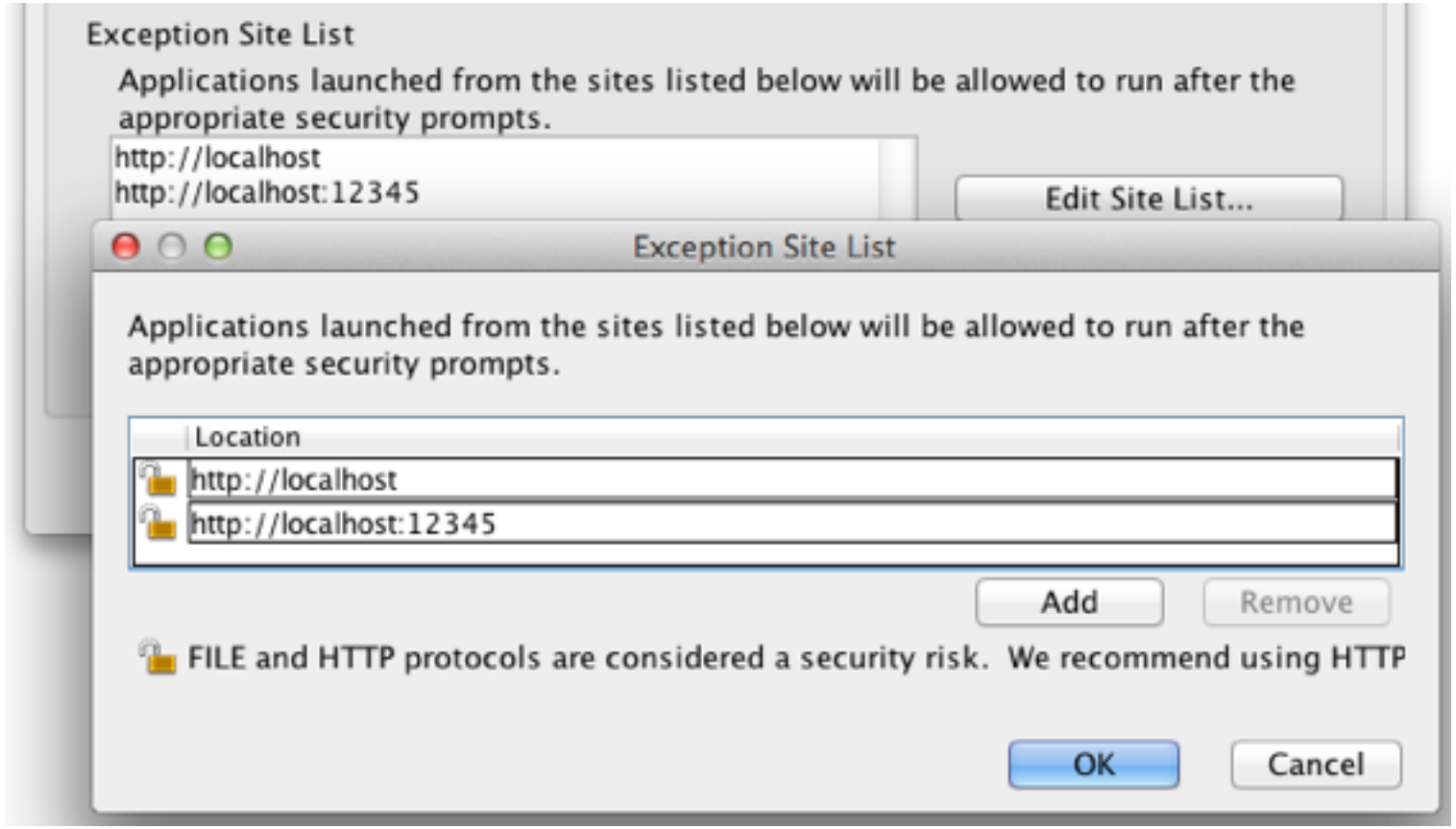
New Features

- Advanced Management Console (AMC)
  - ruleset.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ruleset version="1.1+">
    <rule>
        <id location="http://www.example.com/test.html"/>
        <action permission="block"/>
    </rule>
    <rule>
        <id location="http://www.javatester.org/version.html"/>
        <action force="true" permission="run" version="1.8*"/>
    </rule>
</ruleset>
```

New Features

- Exception Site List (ESL)



Best Practices

Best Practices

Deployment Choices

- Java Web Start
- Browser Applet
- Self-Contained Application (Packager)
  - http://docs.oracle.com/javase/8/docs/technotes/guides/deploy/part_packaging.html#HGBFGCBI

Best Practices

Evaluate your application

- Deployment Considerations
  - all-permissions or sandbox
  - liveconnect, JS->Java, Java->JS, roundtrip
  - JVM options
  - Java System properties
  - JRE Version
  - Location of artifacts - html/jnlp/JARs
    - Location of hosting (same host, different hosts)
    - Lazy resources
    - 3rd party resources
  - Signed JARs

Best Practices

Read Secure Coding Guidelines

- Secure Coding Guidelines http://www.oracle.com/technetwork/java/seccodeguide-139067.html
- Good recommendations regarding coding securely.
- It has some discussion on subjects directly related to deploying Java securely on the web such as how to use doPrivileged blocks

Best Practices

doPrivleged Example

- Example:

```
public class LibClass {
  // System property used by library,
  //  does not contain sensitive information
  private static final String OPTIONS = "xx.lib.options";

  public static String getOptions() {
    return AccessController.doPrivileged(
      new PrivilegedAction<String>() {
        public String run() {
          // this is checked by SecurityManager
          return System.getProperty(OPTIONS);
        }
      }
    );
  }
}
```

Best Practices

Signed JNLP

- The JNLP file is duplicated in your main signed JAR
- Copy of JNLP in signed JAR must match web server copy
  - JNLP-INF/APPLICATION.JNLP
  - JNLP-INF/APPLICATION_TEMPLATE.JNLP
- Template specifies values that can be substituted by the external JNLP

Best Practices

JVM Options and Java system properties

- Secure properties and JVM arguments
  - Considered safe for use by anyone
  - Available to all applications
  - Prefixes such as “jnlp.*” and “javaws.*” available for application defined properties
- Insecure properties and arguments
  - All other properties and arguments
  - Requires use of signed JNLP
- List of secure properties found at http://docs.oracle.com/javase/7/docs/technotes/guides/javaws/developersguide/syntax.html

Best Practices

JRE Version

- Target to family
  - Use either jnlp “java” element, “java_version” applet parameter or deployment toolkit.
  - JNLP example:
    - <java version=”1.7+” />
    - Run with version of 1.7 and above, i.e., will run with 1.8
    - Split JRE versions requiring different params into separate “java” elements, ordered by priority
  - HTML example (applet/object/embed tag):
    - <param name=”java_version” value=”1.7*”/>
    - Run with version in 1.7, i.e. will NOT run with 1.8

Best Practices

JRE Version

- Target to family
  - DT example:
```
<script src="http://java.com/js/dtjava.js">
        dtjava.embed( { url: "app.jnlp }, { jvm: "1.7*" } );
</script>
```
  - Run with version in 1.7, i.e. will NOT run with 1.8
- Target specific version or versions or families
  - 1.7.0_40 Run with 7u40 specifically, not recommended
  - 1.7.0_40+ Run with 7u40 or later
  - 1.6* 1.7* Run in the latest of either 6 or 7

Best Practices

JRE Version

- Example 1: Runs only with Java version 1.7 update 11

```
<param name="java_version" value="1.7.0_11">
```

- Example 2: Runs with only Java version 1.7 (not with 1.6 or 1.8)

```
<param name="java_version" value="1.7*">
```

- Example 3: Runs with all versions of Java 1.7 or higher

```
<param name="java_version" value="1.7+">
```

Best Practices

Prevent your JARs from being repurposed

- Repurposing is when an attacker uses some of your JARs, with some combination of other JARs, JNLP files, applet tags or javascript using liveconnect
- Set optional “Codebase” manifest attribute to indicate origin
  - You can specify multiple locations
  - Prevents attacker copying JAR to another server
  - Examples:
    - Codebase: oracle.com
    - Codebase: https://*.oracle.com *.java.net

Best Practices

Prevent your JARs from being repurposed

- Set “Permissions” manifest attribute to indicate required permission
  - “all-permissions”
  - “sandbox”
- Set permissions level in html/jnlp.
  - Must match “Permissions” manifest attribute
- Mixed code, i.e., both sandbox/all-permissions jars
  - Should not use HTML applet tag for mixed code
  - Use one JNLP to reference JARs with same permissions
  - Reference extension JNLP with different permissions
  - Should sign all JARs with same certificate
  - Extension JARs can be signed with different certificate

Best Practices

Prevent your JARs from being repurposed

- Signed JNLP prevents
  - Your application being run with different application arguments, JVM arguments and system properties
  - Replacing your JARs with other extensions
  - Using your JARs in a non JNLP application
- Recommend using HTTPS
  - Prevents repurposing with man-in-the-middle attack
- Preventing applications from being repurposed http://docs.oracle.com/javase/8/docs/technotes/guides/deploy/manifest.html#A1213309

Best Practices

Signing JARs

- Moved to a model where jars must be signed (required as of Jan 2014)
- More friendly user experience with valid CA signed jars
- Always sign with a valid CA certificate, sandbox and all-permissions
- Timestamp JAR signatures (more robust future validation)
- Revoke your certificate if it is compromised
- Import certificate into key store during development
  - Import from the Java Control Panel
  - Import from command line
  - Self-signed or local CA certificate treated as valid CA certificate

Best Practices

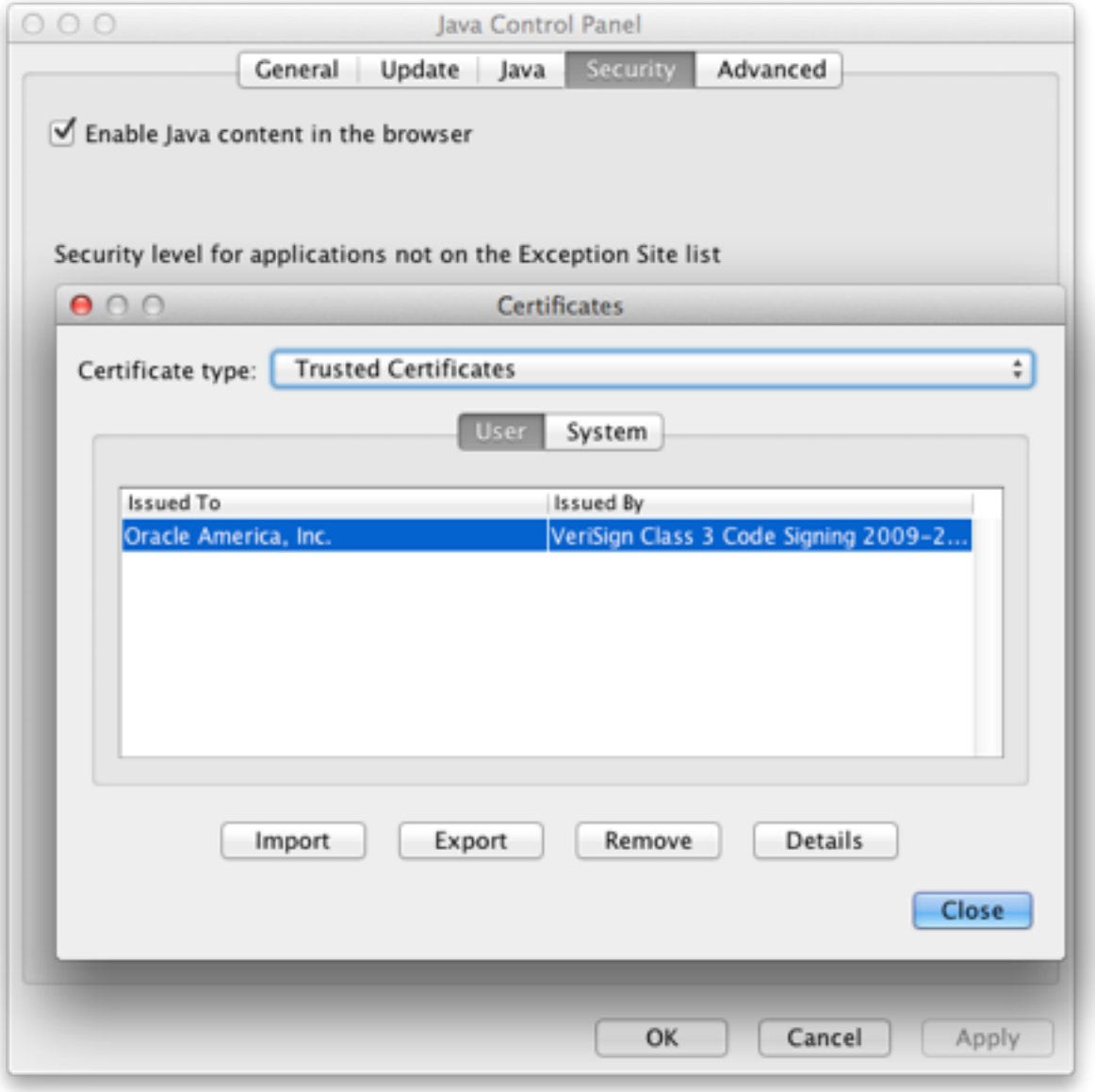
Managing certificates for development

- keytool:
  - keytool -selfcert -alias <name> -dname “CN=..., OU=..., ...”
  - keytool -import -alias <name> -file mycert.cer
  - keytool -export -alias <name> -file outcert.cer

Best Practices

Managing certificates for development

- Java Control Panel:



Best Practices

Keeping your application secure and up to date

- Monitor upcoming changes to Java security and use the latest security features:
- Java Platform Group, Product Management blog (focuses on developers/IT)
  - News about Java from the PM staff
    https://blogs.oracle.com/java-platform-group/
- Security Assurance (all audiences)
  - GPS blog covers Java security news where it impacts Oracle/Java
    https://blogs.oracle.com/security/
- Java Security (for consumers)
  - Related to Java browser users
    http://www.java.com/en/security/

Packager

Packager

- JDK 8u20
  - Renamed Java Packager
  - API for IDEs
    - Add external bundlers
  - Mac PKG Bundler
  - Mac App Store Ready
  - Signing of Mac Apps
  - Services/Daemons

Packager

- JDK 8u40
  - Single Source Launcher
  - Multiple Launchers (Windows, Linux)
  - File Associations
    - PKG, APP, MSI, RPM, DEB
  - Default Application Arguments
  - Simple DMG
  - JVM User Overrides
    - Java API to read/write in Packager.jar

Future

All items discussed are subject to change, bla bla bla

Future
(All items discussed are subject to change)

- Continue to improve security
- Windows Low Integrity
- Java Control Panel Rewrite
  - Security Levels will be a checkbox rather than slider
- Advanced Management Console (AMC) 2.0
  - Web based with REST API
  - Push DRS ruleset.xml to desktop
  - More Instrumenting

Future
(All items discussed are subject to change)

- Packager
  - Better Platform Support
  - Auto Update
  - Launcher metadata
  - Splash Screen
  - Auto Memory

Summary

- Use Packager
- Use Deployment Rule Set:
  - Improve the user experience
  - Continue to use legacy applications in the enterprise
- Follow best practices
  - Use manifest attributes
  - Use HTTPS over HTTP
  - Sign all JARs
  - Follow secure coding guidelines
  - Update your JRE
- Keep up with evolving security practices

Deployment Blog

- https://blogs.oracle.com/talkingjavadeployment/

Packager Talks

- JavaFX Packager Tool Integration Deep Dive - BOF2248
  9/29/14 (Monday) 9:00 PM - Moscone South - 236
  https://oracleus.activeevents.com/2014/connect/sessionDetail.ww?SESSION_ID=2248
- Packaging Your JavaFX Apps for the Mac and the Mac App Store - CON2228
  10/1/14 (Wednesday) 10:00 AM - Hilton - Plaza A
  https://oracleus.activeevents.com/2014/connect/sessionDetail.ww?SESSION_ID=2228
- Packaging and Deploying Java Apps in Java 8u20 - CON2247
  10/2/14 (Thursday) 11:30 AM - Hilton - Plaza A
  https://oracleus.activeevents.com/2014/connect/sessionDetail.ww?SESSION_ID=2247

Questions & Answers

Thank you!

Hardware and Software
Engineered to Work Together

JavaOne™

ORACLE


ORACLE