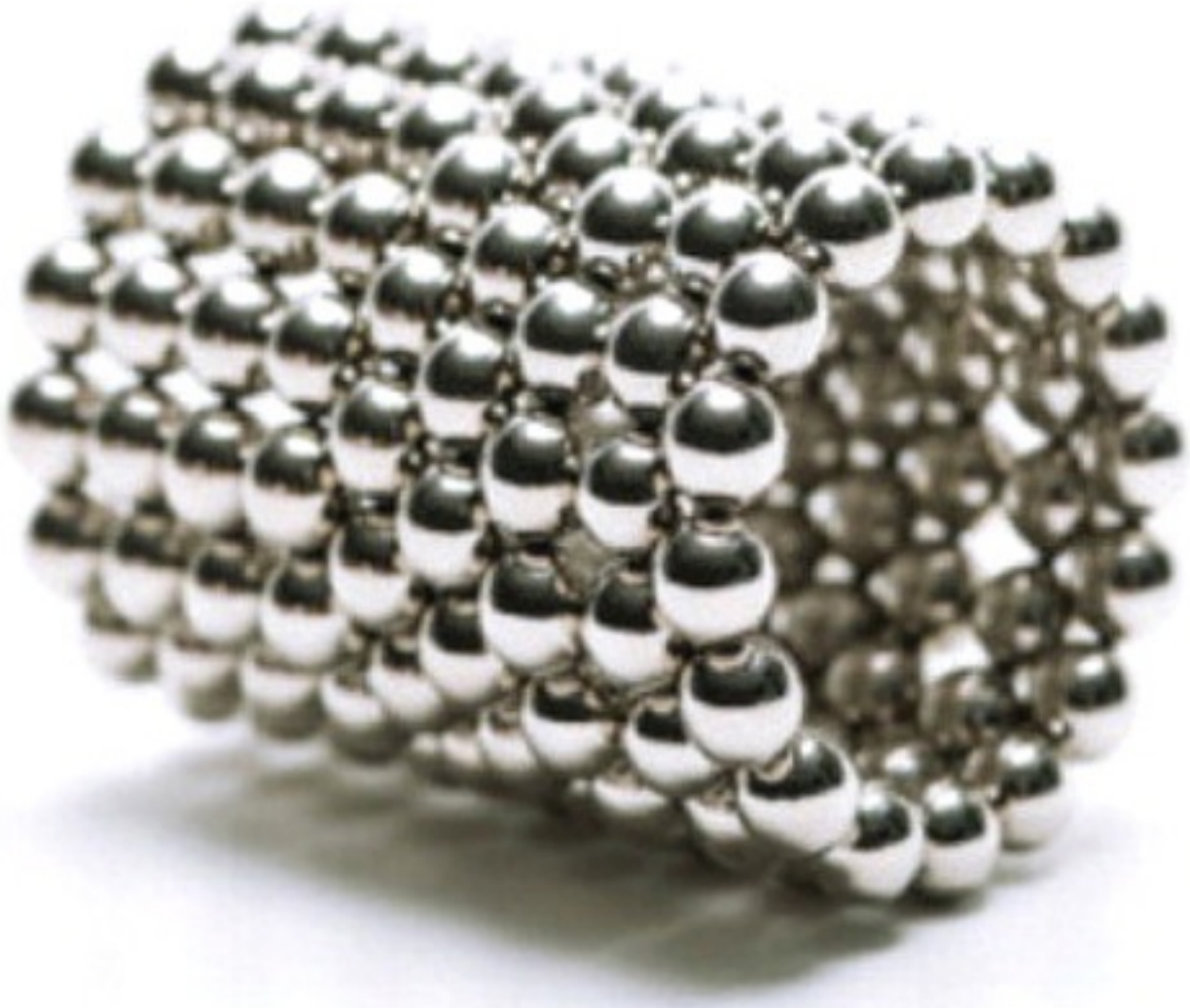
SEI SERIES • A CERT® BOOK

The CERT® Oracle® Secure Coding Standard for Java



Fred Long | Dhruv Mohlndra | Robert C. Seacord
Dean F. Sutherland | David Svoboda

SEI SERIES • A CERT® BOOK

The CERT® Oracle® Secure Coding Standard for Java

Fred Long | Dhruv Mohlndra | Robert C. Seacord
Dean F. Sutherland | David Svoboda

The CERT® Oracle® Secure Coding Standard for Java™

Fred Long
Dhruv Mohindra
Robert C. Seacord
Dean F. Sutherland
David Svoboda

Addison-Wesley
Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

The SEI Series in Software Engineering

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States please contact:

International Sales
international@pearson.com

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data

The CERT Oracle secure coding standard for Java / Fred Long ... [et al.].
p. cm.—(The SEI series in software engineering)
Includes bibliographical references and index.
ISBN-13: 978-0-321-80395-5 (pbk. : alk. paper)
ISBN-10: 0-321-80395-7 (pbk. : alk. paper)
1. Java (Computer program language) 2. Computer security. 3. Oracle (Computer file) 4. Computer programming—Standards. I. Long, F. W. (Frederick W.), 1947- II. Carnegie-Mellon University. CERT Coordination Center.
QA76.73.J38C44 2012
005.8—dc23

2011027284

ISBN-13: 978-0-321-80395-5
ISBN-10: 0-321-80395-7

Text printed in the United States on recycled paper at Edwards Brothers in Ann Arbor, Michigan.

First printing, September 2011

To my late wife, Ann, for all her love, help, and support over the years.

—Fred Long

To my parents Deepak and Eta Mohindra, my grandmother Shashi Mohindra, and our very peppy, spotted Dalmatian Google.

—Dhruv Mohindra

To my wife, Alfie, for making this book worthwhile, and to my parents, Bill and Lois, for making it possible.

—David Svoboda

To my wife, Rhonda, and our children, Chelsea and Jordan.

—Robert C. Seacord

For Libby, who makes everything worthwhile.

—Dean Sutherland

Contents

Foreword

James Gosling

Security in computer systems has been a serious issue for decades. This past decade's explosion in the dependence on networks and the computers connected to them has raised the issue to stratospheric levels. When Java was first designed, dealing with security was a key component. And in the years since then, all of the various standard libraries, frameworks, and containers that have been built have had to deal with security too. In the Java world, security is not viewed as an add-on feature. It is a pervasive way of thinking. Those who forget to think in a secure mindset end up in trouble.

But just because the facilities are there doesn't mean that security is assured automatically. A set of standard practices has evolved over the years. *The CERT® Oracle® Secure Coding Standard for Java™* is a compendium of these practices. These are not theoretical research papers or product marketing blurbs. This is all serious, mission-critical, battle-tested, enterprise-scale stuff.

Preface

An essential element of secure coding in the Java programming language is a well-documented and enforceable coding standard. The CERT Oracle Secure Coding Standard for Java provides rules for secure coding in the Java programming language. The goal of these rules is to eliminate insecure coding practices that can lead to exploitable vulnerabilities. The application of the secure coding standard leads to higher quality systems that are safe, secure, reliable, dependable, robust, resilient, available, and maintainable and can be used as a metric to evaluate source code for these properties (using manual or automated processes).

This coding standard affects a wide range of software systems developed in the Java programming language.

Scope

The CERT Oracle Secure Coding Standard for Java focuses on the Java Standard Edition 6 Platform (Java SE 6) environment and includes rules for secure coding using the Java programming language and libraries. *The Java Language Specification,* 3rd edition [JLS 2005] prescribes the behavior of the Java programming language and served as the primary reference for the development of this standard. This coding standard also addresses new features of the Java SE 7 Platform. Primarily, these features provide alternative compliant solutions to secure coding problems that exist in both the Java SE 6 and Java SE 7 platforms.

Languages such as C and C++ allow undefined, unspecified, or implementation-defined behaviors, which can lead to vulnerabilities when a programmer makes incorrect assumptions about the underlying behavior of an API or language construct. The Java Language Specification goes further to standardize language requirements because Java is designed to be a “write once, run anywhere” language. Even then, certain behaviors are left to the discretion of the implementor of the Java Virtual Machine (JVM) or the Java compiler. This standard identifies such language peculiarities and demonstrates secure coding practices to avoid them.

Focusing only on language issues does not translate to writing secure software. Design flaws in Java application programming interfaces (APIs) sometimes lead to their deprecation. At other times, the APIs or the relevant documentation may be interpreted incorrectly by the programming community. This standard identifies such problematic APIs and highlights their correct use. Examples of commonly used faulty design patterns (anti-patterns) and idioms are also included.

The Java language, its core and extension APIs, and the JVM provide security features such as the security manager, access controller, cryptography, automatic memory management, strong type checking, and bytecode verification. These features provide sufficient security for most applications, but their proper use is of paramount importance. This standard highlights the pitfalls and caveats associated with the security architecture and stresses its correct implementation. Adherence to this standard safeguards the confidentiality, integrity, and availability (CIA) of trusted

programs and helps eliminate exploitable security flaws that can result in denial-of-service attacks, time-of-check-to-time-of-use attacks, information leaks, erroneous computations, and privilege escalation.

Software that complies with this standard provides its users the ability to define fine-grained security policies and safely execute trusted mobile code on untrusted systems or untrusted mobile code on trusted systems.

Included Libraries

This secure coding standard addresses security issues primarily applicable to the `lang` and `util` libraries, as well as to the Collections, Concurrency Utilities, Logging, Management, Reflection, Regular Expressions, Zip, I/O, JMX, JNI, Math, Serialization, and XML JAXP libraries. This standard avoids the inclusion of open bugs that have already been fixed or those that lack security ramifications. A functional bug is included only when it is likely that it occurs with high frequency, causes considerable security concerns, or affects most Java technologies that rely on the core platform. This standard is not limited to security issues specific to the Core API but also includes important security concerns pertaining to the standard extension APIs (`javax` package).

Issues Not Addressed

The following issues are not addressed by this standard:

- **Design and Architecture.** This standard assumes that the design and architecture of the product is secure—that is, that the product is free of design-level vulnerabilities that would otherwise compromise its security.
- **Content.** This coding standard does not address concerns specific to only one Java-based platform but applies broadly to all platforms. For example, rules that are applicable to Java Micro Edition (ME) or Java Enterprise Edition (EE) alone and not to Java SE are typically not included. Within Java SE, APIs that deal with the user interface (User Interface Toolkits) or with the web interface for providing features such as sound, graphical rendering, user account access control, session management, authentication, and authorization are beyond the scope of this standard. However, this does not preclude the standard from discussing networked Java systems given the risks associated with improper input validation and injection flaws and suggesting appropriate mitigation strategies.
- **Coding Style.** Coding style issues are subjective; it has proven impossible to develop a consensus on appropriate style rules. Consequently, *The CERT® Oracle® Secure Coding Standard for Java™* recommends only that the user define style rules and apply those rules consistently; requirements that mandate use of any particular coding style are deliberately omitted. The easiest way to consistently apply a coding style is with the use of a code formatting tool. Many integrated development environments (IDEs) provide such capabilities.
- **Tools.** As a federally funded research and development center (FFRDC), the Software Engineering Institute (SEI) is not in a position to recommend particular vendors or tools to enforce the restrictions adopted. Users of this document are

free to choose tools; vendors are encouraged to provide tools to enforce these rules.

- **Controversial Rules.** In general, the CERT secure coding standards try to avoid the inclusion of controversial rules that lack a broad consensus.

Audience

The CERT® Oracle® Secure Coding Standard for Java™ is primarily intended for developers of Java language programs. While this standard focuses on the Java Platform SE 6, it should also be informative (although incomplete) for Java developers working with Java ME or Java EE and other Java language versions.

While primarily designed for secure systems, this standard is also useful for achieving other quality attributes such as safety, reliability, dependability, robustness, resiliency, availability, and maintainability.

This standard may also be used by

- Developers of analyzer tools who wish to diagnose insecure or nonconforming Java language programs
- Software development managers, software acquirers, or other software development and acquisition specialists to establish a proscriptive set of secure coding standards
- Educators as a primary or secondary text for software security courses that teach secure coding in Java

The rules in this standard may be extended with organization-specific rules. However, a program must comply with existing rules to be considered conforming to the standard.

Training may be developed to educate software professionals regarding the appropriate application of secure coding standards. After passing an examination, these trained programmers may also be certified as secure coding professionals.

Contents and Organization

The standard is organized into an introductory chapter and 17 chapters containing rules in specific topic areas. Each of the rule chapters contains a list of rules in that section, and a risk assessment summary for the rules. There is also a common glossary and bibliography. This preface is meant to be read first, followed by the introductory chapter. The rule chapters may be read in any order or used as reference material as appropriate. The rules are loosely organized in each chapter but, in general, may also be read in any order.

Rules have a consistent structure. Each rule has a unique identifier, which is included in the title. The title of the rules and the introductory paragraphs define the conformance requirements. This is typically followed by one or more sets of noncompliant code examples and corresponding compliant solutions. Each rule also includes a risk assessment and bibliographical references specific to that rule. When applicable, rules also list related vulnerabilities and related guidelines from the following sources:

- *The CERT® C Secure Coding Standard* [Seacord 2008]
- *The CERT® C++ Secure Coding Standard* [CERT 2011]
- ISO/IEC TR 24772. Information Technology—Programming Languages—Guidance to Avoiding Vulnerabilities in Programming Languages through Language Selection and Use [ISO/IEC TR 24772:2010]
- MITRE CWE [MITRE 2011]
- Secure Coding Rules for the Java Programming Language, version 3.0 [SCG 2009]
- *The Elements of Java™ Style* [Rogue 2000]

Identifiers

Each rule has a unique identifier, consisting of three parts:

- A three-letter mnemonic, representing the section of the standard, is used to group similar rules and make them easier to find.
- A two-digit numeric value in the range of 00 to 99, which ensures each rule has a unique identifier.
- The letter J, which indicates that this is a Java language rule and is included to prevent ambiguity with similar rules in CERT secure coding standards for other languages.

Identifiers may be used by static analysis tools to reference a particular rule in a diagnostic message or otherwise used as shorthand for the rule title.

System Qualities

Security is one of many system attributes that must be considered in the selection and application of a coding standard. Other attributes of interest include safety, portability, reliability, availability, maintainability, readability, and performance.

Many of these attributes are interrelated in interesting ways. For example, readability is an attribute of maintainability; both are important for limiting the introduction of defects during maintenance that can result in security flaws or reliability issues. In addition, readability facilitates code inspection by safety officers. Reliability and availability require proper resource management, which also contributes to the safety and security of the system. System attributes such as performance and security are often in conflict, requiring tradeoffs to be made.

The purpose of the secure coding standard is to promote software security. However, because of the relationship between security and other system attributes, the coding standards may include requirements and recommendations that deal primarily with other system attributes that also have a significant impact on security.

Priority and Levels

Each rule has an assigned priority. Priorities are assigned using a metric based on Failure Mode, Effects, and Criticality Analysis (FMECA) [IEC 60812]. Three values are assigned for each rule on a scale of 1 to 3 for

- Severity—How serious are the consequences of the rule being ignored:

  1 = low (denial-of-service attack, abnormal termination)

  2 = medium (data integrity violation, unintentional information disclosure)

  3 = high (run arbitrary code, privilege escalation)
- Likelihood—How likely is it that a flaw introduced by violating the rule could lead to an exploitable vulnerability:

  1 = unlikely

  2 = probable

  3 = likely
- Remediation cost—How expensive is it to remediate existing code to comply with the rule:

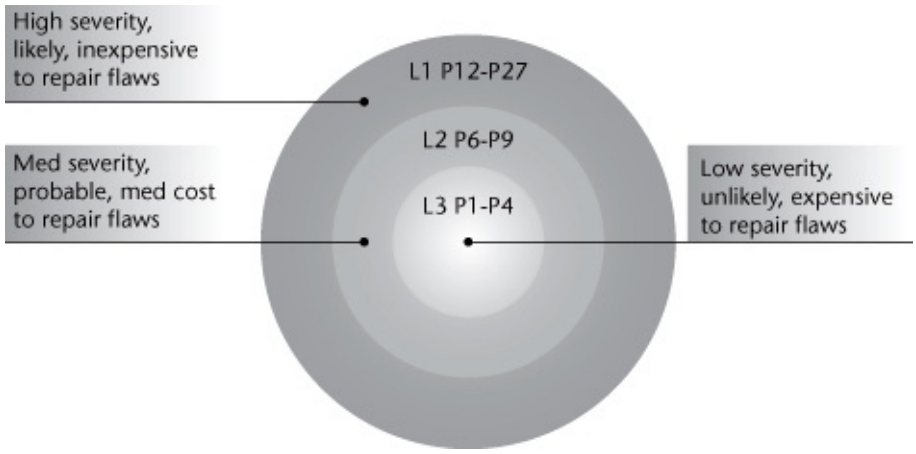
  1 = high (manual detection and correction)

  2 = medium (automatic detection and manual correction)

  3 = low (automatic detection and correction)

The three values are multiplied together for each rule. This product provides a measure that can be used in prioritizing the application of the rules. These products range from 1 to 27. Rules with a priority in the range of 1 to 4 are level 3 rules, 6 to 9 are level 2, and 12 to 27 are level 1. As a result, it is possible to claim level 1, level 2, or complete compliance (level 3) with a standard by implementing all rules in a level, as shown in Figure P–1.

Figure P–1. Levels and priority ranges



The metric is designed primarily for remediation projects and does not apply to new development efforts that are implemented to the standard.

Conformance Testing

Software systems can be validated as conforming to *The CERT® Oracle® Secure Coding Standard for Java™*.

Normative vs. Nonnormative Text

Portions of this coding standard are intended to be normative; other portions are