

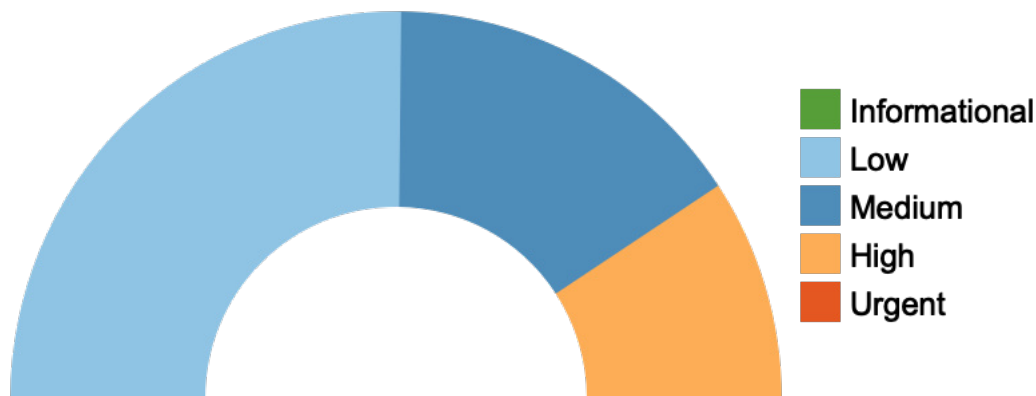
The CERT Oracle Secure Coding Standard for Java provides rules for secure coding in the Java programming language.

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

The rules and recommendations in this standard are a work in progress and reflect the current thinking of the secure coding community. As rules and recommendations mature, they are published in report or book form as official releases. These releases are issued as dictated by the needs and interests of the secure software development community.

The list of rules and recommendations in this tool were last updated on 2023/05/23.

Checks by Severity



Checks

Check ID	Check Name	Supported	Severity
DCL00-J	Prevent class initialization cycles	Yes	Low
DCL01-J	Do not reuse public identifiers from the Java Standard Library	No	Low
DCL02-J	Do not modify the collection's elements during an enhanced for statement	Yes	Low
ENV00-J	Do not sign code that performs only unprivileged operations	No	High
ENV01-J	Place all security-sensitive code in a single JAR and sign and seal it	No	High
ENV02-J	Do not trust the values of environment variables	Yes	Low
ENV03-J	Do not grant dangerous combinations of	Yes	High

	permissions		
ENV04-J	Do not disable bytecode verification	No	High
ENV05-J	Do not deploy an application that can be remotely monitored	No	High
ENV06-J	Production code must not contain debugging entry points	Yes	High
ERR00-J	Do not suppress or ignore checked exceptions	Yes	Low
ERR01-J	Do not allow exceptions to expose sensitive information	Yes	Medium
ERR02-J	Prevent exceptions while logging data	Yes	Medium
ERR03-J	Restore prior object state on method failure	Yes	Low
ERR04-J	Do not complete abruptly from a finally block	Yes	Low
ERR05-J	Do not let checked exceptions escape from a finally block	Yes	Low
ERR06-J	Do not throw undeclared checked exceptions	Yes	Low
ERR07-J	Do not throw RuntimeException, Exception, or Throwable	Yes	Low
ERR08-J	Do not catch NullPointerException or any of its ancestors	Yes	Medium
ERR09-J	Do not allow untrusted code to terminate the JVM	Yes	Low
EXP00-J	Do not ignore values returned by methods.	Yes	Medium
EXP01-J	Do not use a null in a case where an object is required	No	Low
EXP02-J	Do not use the Object.equals() method to compare two arrays.	Yes	Low
EXP03-J	Do not use the equality operators when comparing values of boxed primitives	Yes	Low
EXP04-J	Do not pass arguments to certain Java Collections Framework methods that are a different type than the collection parameter type	Yes	Low
EXP05-J	Do not follow a write by a subsequent write or read of the same object within an expression	Yes	Low
EXP06-J	Expressions used in assertions must not produce side effects	Yes	Low
FIO00-J	Do not operate on files in shared directories	Yes	Medium
FIO01-J	Create files with appropriate access permissions	Yes	Medium
FIO02-J	Detect and handle file-related errors	Yes	Medium
FIO03-J	Remove temporary files before termination	Yes	Medium

FIO04-J	Release resources when they are no longer needed	Yes	Low
FIO05-J	Do not expose buffers or their backing arrays methods to untrusted code	Yes	Medium
FIO06-J	Do not create multiple buffered wrappers on a single byte or character stream	No	Low
FIO07-J	Do not let external processes block on IO buffers	No	Low
FIO08-J	Distinguish between characters or bytes read from a stream and -1	No	High
FIO09-J	Do not rely on the write() method to output integers outside the range 0 to 255	Yes	Low
FIO10-J	Ensure the array is filled when using read() to fill an array	Yes	Low
FIO12-J	Provide methods to read and write little-endian data	No	Low
FIO13-J	Do not log sensitive information outside a trust boundary	Yes	Medium
FIO14-J	Perform proper cleanup at program termination	Yes	Medium
FIO15-J	Do not reset a servlet's output stream after committing it	No	Low
FIO16-J	Canonicalize path names before validating them	No	Medium
IDS00-J	Prevent SQL Injection	Yes	High
IDS01-J	Normalize strings before validating them	Yes	High
IDS03-J	Do not log unsanitized user input	No	Medium
IDS04-J	Safely extract files from ZipInputStream	Yes	Low
IDS06-J	Exclude unsanitized user input from format strings	Yes	Medium
IDS07-J	Sanitize untrusted data passed to the Runtime.exec() method	No	High
IDS08-J	Sanitize untrusted data included in a regular expression	Yes	Medium
IDS11-J	Perform any string modifications before validation	Yes	High
IDS14-J	Do not trust the contents of hidden form fields	No	High
IDS16-J	Prevent XML Injection	Yes	High
IDS17-J	Prevent XML External Entity Attacks	No	Medium
JNI00-J	Define wrappers around native methods	Yes	Medium
LCK00-J	Use private final lock objects to synchronize classes that may interact with untrusted code	Yes	Low

LCK01-J	Do not synchronize on objects that may be reused	Yes	Medium
LCK02-J	Do not synchronize on the class object returned by getClass()	Yes	Medium
LCK03-J	Do not synchronize on the intrinsic locks of high-level concurrency objects	No	Medium
LCK04-J	Do not synchronize on a collection view if the backing collection is accessible	Yes	Low
LCK05-J	Synchronize access to static fields that can be modified by untrusted code	Yes	Low
LCK06-J	Do not use an instance lock to protect shared static data	Yes	Medium
LCK07-J	Avoid deadlock by requesting and releasing locks in the same order	Yes	Low
LCK08-J	Ensure actively held locks are released on exceptional conditions	Yes	Low
LCK09-J	Do not perform operations that can block while holding a lock	Yes	Low
LCK10-J	Use a correct form of the double-checked locking idiom	Yes	Low
LCK11-J	Avoid client-side locking when using classes that do not commit to their locking strategy	Yes	Low
MET00-J	Validate method arguments	Yes	High
MET01-J	Never use assertions to validate method arguments	Yes	Medium
MET02-J	Do not use deprecated or obsolete classes or methods	Yes	Low
MET03-J	Methods that perform a security check must be declared private or final.	Yes	Medium
MET04-J	Do not increase the accessibility of overridden or hidden methods	Yes	Medium
MET05-J	Ensure that constructors do not call overridable methods	Yes	Medium
MET06-J	Do not invoke overridable methods in clone()	Yes	Medium
MET07-J	Never declare a class method that hides a method declared in a superclass or superinterface	Yes	Low
MET08-J	Preserve the equality contract when overriding the equals() method	Yes	Low
MET09-J	Classes that define an equals() method must also define a hashCode() method	Yes	Low

MET10-J	Follow the general contract when implementing the compareTo() method	Yes	Medium
MET11-J	Ensure that keys used in comparison operations are immutable	Yes	Low
MET12-J	Do not use finalizers	Yes	Medium
MET13-J	Do not assume that reassigning method arguments modifies the calling environment	No	Medium
MSC00-J	Use SSLSocket rather than Socket for secure data exchange	Yes	Medium
MSC01-J	Do not use an empty infinite loop	Yes	Low
MSC02-J	Generate strong random numbers	Yes	High
MSC03-J	Never hard code sensitive information	No	High
MSC04-J	Do not leak memory	No	Low
MSC05-J	Do not exhaust heap space	No	Low
MSC06-J	Do not modify the underlying collection when an iteration is in progress	No	Low
MSC07-J	Prevent multiple instantiations of singleton objects	No	Low
NUM00-J	Detect or prevent integer overflow	Yes	Medium
NUM01-J	Do not perform bitwise and arithmetic operations on the same data	No	Medium
NUM02-J	Ensure that division and remainder operations do not result in divide-by-zero errors	Yes	Low
NUM03-J	Use integer types that can fully represent the possible range of unsigned data	No	Low
NUM04-J	Do not use floating-point numbers if precise computation is required	No	Low
NUM07-J	Do not attempt comparisons with NaN	Yes	Low
NUM08-J	Check floating-point inputs for exceptional values	No	Low
NUM09-J	Do not use floating-point variables as loop counters	Yes	Low
NUM10-J	Do not construct BigDecimal objects from floating-point literals	Yes	Low
NUM11-J	Do not compare or inspect the string representation of floating-point values	Yes	Low
NUM12-J	Ensure conversions of numeric types to narrower types do not result in lost or misinterpreted data	Yes	Low
NUM13-J	Avoid loss of precision when converting primitive integers to floating-point	Yes	Low

NUM14-J	Use shift operators correctly	No	Low
OBJ01-J	Limit accessibility of fields	Yes	Medium
OBJ02-J	Preserve dependencies in subclasses when changing superclasses	No	Medium
OBJ03-J	Prevent heap pollution	No	Low
OBJ04-J	Provide mutable classes with copy functionality to safely allow passing instances to untrusted code	Yes	Low
OBJ05-J	Do not return references to private mutable class members	Yes	High
OBJ06-J	Defensively copy mutable inputs and mutable internal components	No	Medium
OBJ07-J	Sensitive classes must not let themselves be copied	Yes	Medium
OBJ08-J	Do not expose private members of an outer class from within a nested class	Yes	Medium
OBJ09-J	Compare classes and not class names	Yes	High
OBJ10-J	Do not use public static nonfinal fields	Yes	Medium
OBJ11-J	Be wary of letting constructors throw exceptions	Yes	High
OBJ13-J	Ensure that references to mutable objects are not exposed	Yes	Medium
SEC00-J	Do not allow privileged blocks to leak sensitive information across a trust boundary	No	Medium
SEC01-J	Do not allow tainted variables in privileged blocks	Yes	High
SEC02-J	Do not base security checks on untrusted sources	No	High
SEC03-J	Do not load trusted classes after allowing untrusted code to load arbitrary classes	No	High
SEC04-J	Protect sensitive operations with security manager checks	No	High
SEC05-J	Do not use reflection to increase accessibility of classes, methods, or fields	No	High
SEC06-J	Do not rely on the default automatic signature verification provided by URLClassLoader and java.util.jar	No	High
SEC07-J	Call the superclass's getPermissions() method when writing a custom class loader	Yes	High
SER00-J	Enable serialization compatibility during class evolution	No	Low

SER01-J	Do not deviate from the proper signatures of serialization methods	Yes	High
SER02-J	Sign then seal objects before sending them outside a trust boundary	No	Medium
SER03-J	Do not serialize unencrypted sensitive data	No	Medium
SER04-J	Do not allow serialization and deserialization to bypass the security manager	Yes	High
SER05-J	Do not serialize instances of inner classes	Yes	Medium
SER06-J	Make defensive copies of private mutable components during deserialization	Yes	Low
SER07-J	Do not use the default serialized form for classes with implementation-defined invariants	Yes	Medium
SER08-J	Minimize privileges before deserializing from a privileged context	No	High
SER09-J	Do not invoke overridable methods from the readObject() method	Yes	Low
SER10-J	Avoid memory and resource leaks during serialization	No	Low
SER11-J	Prevent overwriting of externalizable objects	No	Low
SER12-J	Prevent deserialization of untrusted data	Yes	High
STR00-J	Don't form strings containing partial characters from variable-width encodings	No	Low
STR01-J	Do not assume that a Java char fully represents a Unicode code point	Yes	Low
STR02-J	Specify an appropriate locale when comparing locale-dependent data	No	Medium
STR03-J	Do not encode noncharacter data as a string	Yes	Low
STR04-J	Use compatible character encodings when communicating string data between JVMs	No	Low
THI00-J	Do not invoke Thread.run()	Yes	Low
THI01-J	Do not invoke ThreadGroup methods	Yes	Low
THI02-J	Notify all waiting threads rather than a single thread.	Yes	Low
THI03-J	Always invoke wait() and await() methods inside a loop	Yes	Low
THI04-J	Ensure that threads performing blocking operations can be terminated	Yes	Low
THI05-J	Do not use Thread.stop() to terminate threads.	Yes	Low
TPS00-J	Use thread pools to enable graceful degradation of service during traffic bursts	Yes	Low
TPS01-J	Do not execute interdependent tasks in a	Yes	Low

	bounded thread pool		
TPS02-J	Ensure that tasks submitted to a thread pool are interruptible	Yes	Low
TPS03-J	Ensure that tasks executing in a thread pool do not fail silently	Yes	Low
TPS04-J	Ensure ThreadLocal variables are reinitialized when using thread pools	Yes	Medium
TSM00-J	Do not override thread-safe methods with methods that are not thread-safe	Yes	Low
TSM01-J	Do not let the this reference escape during object construction	Yes	Medium
TSM02-J	Do not use background threads during class initialization	Yes	Low
TSM03-J	Do not publish partially initialized objects	Yes	Medium
VNA00-J	Ensure visibility when accessing shared primitive variables	Yes	Medium
VNA01-J	Ensure visibility of shared references to immutable objects	Yes	Low
VNA02-J	Ensure that compound operations on shared variables are atomic	Yes	Medium
VNA03-J	Do not assume that a group of calls to independently atomic methods is atomic	Yes	Low
VNA04-J	Ensure that calls to chained methods are atomic	Yes	Low
VNA05-J	Ensure atomicity when reading and writing 64-bit values	Yes	Low