2 Rules

Sections

Rule Listing

- EXP03-J. Do not use the equality operators when comparing values of boxed primitives
- EXP04-J. Do not pass arguments to certain Java Collections Framework methods that are a different type than the collection parameter type
- EXP05-J. Do not follow a write by a subsequent write or read of the same object within an expression
- EXP06-J. Expressions used in assertions must not produce side effects
- EXP07-J. Prevent loss of useful data due to weak references
- FIO00-J. Do not operate on files in shared directories
- FIO01-J. Create files with appropriate access permissions
- FIO02-J. Detect and handle file-related errors
- FIO03-J. Remove temporary files before termination
- FIO04-J. Release resources when they are no longer needed
- FIO05-J. Do not expose buffers created using the wrap() or duplicate() methods to untrusted code
- FIO06-J. Do not create multiple buffered wrappers on a single byte or character stream
- FIO07-J. Do not let external processes block on IO buffers
- FIO08-J. Distinguish between characters or bytes read from a stream and -1
- FIO09-J. Do not rely on the write() method to output integers outside the range 0 to 255
- FIO10-J. Ensure the array is filled when using read() to fill an array
- FIO11-J. Do not convert between strings and bytes without specifying a valid character encoding
- FIO12-J. Provide methods to read and write little-endian data
- FIO13-J. Do not log sensitive information outside a trust boundary
- FIO14-J. Perform proper cleanup at program termination
- FIO15-J. Do not reset a servlet's output stream after committing it
- FIO16-J. Canonicalize path names before validating them
- IDS00-J. Prevent SQL injection
- IDS01-J. Normalize strings before validating them
- IDS02-J. Canonicalize path names before validating them
- IDS03-J. Do not log unsanitized user input
- IDS04-J. Safely extract files from ZipInputStream
- IDS05-J. Use a safe subset of ASCII for file and path names
- IDS06-J. Exclude unsanitized user input from format strings
- IDS07-J. Sanitize untrusted data passed to the Runtime.exec() method
- IDS08-J. Sanitize untrusted data included in a regular expression
- IDS09-J. Specify an appropriate locale when comparing locale-dependent data
- IDS10-J. Don't form strings containing partial characters
- IDS11-J. Perform any string modifications before validation
- IDS13-J. Use compatible character encodings on both sides of file or network IO
- IDS14-J. Do not trust the contents of hidden form fields
- IDS15-J. Do not allow sensitive information to leak outside a trust boundary
- IDS16-J. Prevent XML Injection
- IDS17-J. Prevent XML External Entity Attacks
- JNI00-J. Define wrappers around native methods
- JNI01-J. Safely invoke standard APIs that perform tasks using the immediate caller's class loader instance (loadLibrary)
- JNI02-J. Do not assume object references are constant or unique
- JNI03-J. Do not use direct pointers to Java objects in JNI code
- JNI04-J. Do not assume that Java strings are null-terminated
- LCK00-J. Use private final lock objects to synchronize classes that may interact with untrusted code
- LCK01-J. Do not synchronize on objects that may be reused
- LCK02-J. Do not synchronize on the class object returned by getClass()
- LCK03-J. Do not synchronize on the intrinsic locks of high-level concurrency objects
- LCK04-J. Do not synchronize on a collection view if the backing collection is accessible
- LCK05-J. Synchronize access to static fields that can be modified by untrusted code

- LCK06-J. Do not use an instance lock to protect shared static data
- LCK07-J. Avoid deadlock by requesting and releasing locks in the same order
- LCK08-J. Ensure actively held locks are released on exceptional conditions
- LCK09-J. Do not perform operations that can block while holding a lock
- LCK10-J. Use a correct form of the double-checked locking idiom
- LCK11-J. Avoid client-side locking when using classes that do not commit to their locking strategy
- MET00-J. Validate method arguments
- MET01-J. Never use assertions to validate method arguments
- MET02-J. Do not use deprecated or obsolete classes or methods
- MET03-J. Methods that perform a security check must be declared private or final
- MET04-J. Do not increase the accessibility of overridden or hidden methods
- MET05-J. Ensure that constructors do not call overridable methods
- MET06-J. Do not invoke overridable methods in clone()
- MET07-J. Never declare a class method that hides a method declared in a superclass or superinterface
- MET08-J. Preserve the equality contract when overriding the equals() method
- MET09-J. Classes that define an equals() method must also define a hashCode() method
- MET10-J. Follow the general contract when implementing the compareTo() method
- MET11-J. Ensure that keys used in comparison operations are immutable
- MET12-J. Do not use finalizers
- MET13-J. Do not assume that reassigning method arguments modifies the calling environment
- MSC00-J. Use SSLSocket rather than Socket for secure data exchange
- MSC01-J. Do not use an empty infinite loop
- MSC02-J. Generate strong random numbers
- MSC03-J. Never hard code sensitive information
- MSC04-J. Do not leak memory
- MSC05-J. Do not exhaust heap space
- MSC06-J. Do not modify the underlying collection when an iteration is in progress
- MSC07-J. Prevent multiple instantiations of singleton objects
- MSC08-J. Do not store nonserializable objects as attributes in an HTTP session
- MSC09-J. For OAuth, ensure (a) [relying party receiving user's ID in last step] is same as (b) [relying party the access token was granted to].
- MSC10-J. Do not use OAuth 2.0 implicit grant (unmodified) for authentication
- MSC11-J. Do not let session information leak within a servlet
- NUM00-J. Detect or prevent integer overflow
- NUM01-J. Do not perform bitwise and arithmetic operations on the same data
- NUM02-J. Ensure that division and remainder operations do not result in divide-by-zero errors
- NUM03-J. Use integer types that can fully represent the possible range of unsigned data
- NUM04-J. Do not use floating-point numbers if precise computation is required
- NUM07-J. Do not attempt comparisons with NaN
- NUM08-J. Check floating-point inputs for exceptional values
- NUM09-J. Do not use floating-point variables as loop counters
- NUM10-J. Do not construct BigDecimal objects from floating-point literals
- NUM11-J. Do not compare or inspect the string representation of floating-point values
- NUM12-J. Ensure conversions of numeric types to narrower types do not result in lost or misinterpreted data
- NUM13-J. Avoid loss of precision when converting primitive integers to floating-point
- NUM14-J. Use shift operators correctly
- OBJ01-J. Limit accessibility of fields
- OBJ02-J. Preserve dependencies in subclasses when changing superclasses
- OBJ03-J. Prevent heap pollution
- OBJ04-J. Provide mutable classes with copy functionality to safely allow passing instances to untrusted code
- OBJ05-J. Do not return references to private mutable class members

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