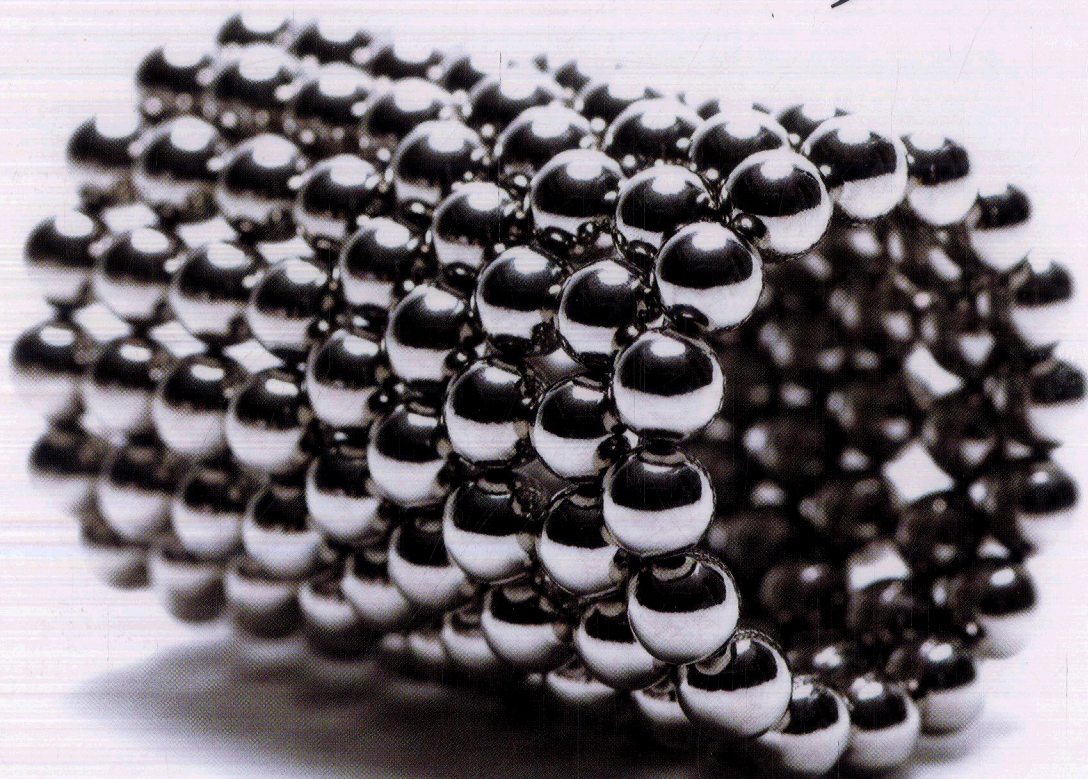


THE CERT® ORACLE® SECURE CODING STANDARD FOR JAVA™



FRED LONG | DHURUV MOHINDRA | ROBERT C. SEACORD
DEAN F. SUTHERLAND | DAVID SVOBODA

Contents

Foreword	xvii
Preface	xix
Acknowledgments	xxxi
About the Authors	xxxiii

Chapter 1	Introduction	1
	Misplaced Trust	2
	Injection Attacks	2
	Leaking Sensitive Data	4
	Leaking Capabilities	6
	Denial of Service	7
	Serialization	10
	Concurrency, Visibility, and Memory	11
	Principle of Least Privilege	18
	Security Managers	19
	Class Loaders	21
	Summary	21
Chapter 2	Input Validation and Data Sanitization (IDS)	23
	Rules	23
	Risk Assessment Summary	24
	IDS00-J. Sanitize untrusted data passed across a trust boundary	24

IDS01-J. Normalize strings before validating them	34
IDS02-J. Canonicalize path names before validating them	36
IDS03-J. Do not log unsanitized user input	41
IDS04-J. Limit the size of files passed to <code>ZipInputStream</code>	43
IDS05-J. Use a subset of ASCII for file and path names	46
IDS06-J. Exclude user input from format strings	48
IDS07-J. Do not pass untrusted, unsanitized data to the <code>Runtime.exec()</code> method	50
IDS08-J. Sanitize untrusted data passed to a regex	54
IDS09-J. Do not use locale-dependent methods on locale-dependent data without specifying the appropriate locale	59
IDS10-J. Do not split characters between two data structures	60
IDS11-J. Eliminate noncharacter code points before validation	66
IDS12-J. Perform lossless conversion of String data between differing character encodings	68
IDS13-J. Use compatible encodings on both sides of file or network I/O	71

Chapter 3	Declarations and Initialization (DCL)	75
	Rules	75
	Risk Assessment Summary	75
	DCL00-J. Prevent class initialization cycles	75
	DCL01-J. Do not reuse public identifiers from the Java Standard Library	79
	DCL02-J. Declare all enhanced for statement loop variables final	81

Chapter 4	Expressions (EXP)	85
	Rules	85
	Risk Assessment Summary	85
	EXP00-J. Do not ignore values returned by methods	86
	EXP01-J. Never dereference null pointers	88
	EXP02-J. Use the two-argument <code>Arrays.equals()</code> method to compare the contents of arrays	90
	EXP03-J. Do not use the equality operators when comparing values of boxed primitives	91
	EXP04-J. Ensure that autoboxed values have the intended type	97
	EXP05-J. Do not write more than once to the same variable within an expression	100
	EXP06-J. Do not use side-effecting expressions in assertions	103

Chapter 5	Numeric Types and Operations (NUM)	105
	Rules	105
	Risk Assessment Summary	106
	NUM00-J. Detect or prevent integer overflow	106
	NUM01-J. Do not perform bitwise and arithmetic operations on the same data	114
	NUM02-J. Ensure that division and modulo operations do not result in divide-by-zero errors	119
	NUM03-J. Use integer types that can fully represent the possible range of unsigned data	121
	NUM04-J. Do not use floating-point numbers if precise computation is required	122
	NUM05-J. Do not use denormalized numbers	125
	NUM06-J. Use the <code>strictfp</code> modifier for floating-point calculation consistency across platforms	128
	NUM07-J. Do not attempt comparisons with NaN	132
	NUM08-J. Check floating-point inputs for exceptional values	134
	NUM09-J. Do not use floating-point variables as loop counters	136
	NUM10-J. Do not construct <code>BigDecimal</code> objects from floating-point literals	138
	NUM11-J. Do not compare or inspect the string representation of floating-point values	139
	NUM12-J. Ensure conversions of numeric types to narrower types do not result in lost or misinterpreted data	141
	NUM13-J. Avoid loss of precision when converting primitive integers to floating-point	146
Chapter 6	Object Orientation (OBJ)	151
	Rules	151
	Risk Assessment Summary	152
	OBJ00-J. Limit extensibility of classes and methods with invariants to trusted subclasses only	152
	OBJ01-J. Declare data members as private and provide accessible wrapper methods	159
	OBJ02-J. Preserve dependencies in subclasses when changing superclasses	162
	OBJ03-J. Do not mix generic with nongeneric raw types in new code	169
	OBJ04-J. Provide mutable classes with copy functionality to safely allow passing instances to untrusted code	175

	OBJ05-J. Defensively copy private mutable class members before returning their references	180
	OBJ06-J. Defensively copy mutable inputs and mutable internal components	185
	OBJ07-J. Sensitive classes must not let themselves be copied	189
	OBJ08-J. Do not expose private members of an outer class from within a nested class	192
	OBJ09-J. Compare classes and not class names	194
	OBJ10-J. Do not use public static nonfinal variables	197
	OBJ11-J. Be wary of letting constructors throw exceptions	199
Chapter 7	Methods (MET)	209
	Rules	209
	Risk Assessment Summary	210
	MET00-J. Validate method arguments	210
	MET01-J. Never use assertions to validate method arguments	213
	MET02-J. Do not use deprecated or obsolete classes or methods	215
	MET03-J. Methods that perform a security check must be declared private or final	217
	MET04-J. Do not increase the accessibility of overridden or hidden methods	218
	MET05-J. Ensure that constructors do not call overridable methods	220
	MET06-J. Do not invoke overridable methods in <code>clone()</code>	223
	MET07-J. Never declare a class method that hides a method declared in a superclass or superinterface	226
	MET08-J. Ensure objects that are equated are equatable	229
	MET09-J. Classes that define an <code>equals()</code> method must also define a <code>hashCode()</code> method	238
	MET10-J. Follow the general contract when implementing the <code>compareTo()</code> method	241
	MET11-J. Ensure that keys used in comparison operations are immutable	243
	MET12-J. Do not use finalizers	248
Chapter 8	Exceptional Behavior (ERR)	255
	Rules	255
	Risk Assessment Summary	255
	ERR00-J. Do not suppress or ignore checked exceptions	256
	ERR01-J. Do not allow exceptions to expose sensitive information	263

ERR02-J. Prevent exceptions while logging data	268
ERR03-J. Restore prior object state on method failure	270
ERR04-J. Do not exit abruptly from a <code>finally</code> block	275
ERR05-J. Do not let checked exceptions escape from a <code>finally</code> block	277
ERR06-J. Do not throw undeclared checked exceptions	280
ERR07-J. Do not throw <code>RuntimeException</code> , <code>Exception</code> , or <code>Throwable</code>	285
ERR08-J. Do not catch <code>NullPointerException</code> or any of its ancestors	288
ERR09-J. Do not allow untrusted code to terminate the JVM	296

Chapter 9 Visibility and Atomicity (VNA) 301

Rules	301
Risk Assessment Summary	301
VNA00-J. Ensure visibility when accessing shared primitive variables	302
VNA01-J. Ensure visibility of shared references to immutable objects	306
VNA02-J. Ensure that compound operations on shared variables are atomic	309
VNA03-J. Do not assume that a group of calls to independently atomic methods is atomic	317
VNA04-J. Ensure that calls to chained methods are atomic	323
VNA05-J. Ensure atomicity when reading and writing 64-bit values	328

Chapter 10 Locking (LCK) 331

Rules	331
Risk Assessment Summary	332
LCK00-J. Use private final lock objects to synchronize classes that may interact with untrusted code	332
LCK01-J. Do not synchronize on objects that may be reused	339
LCK02-J. Do not synchronize on the class object returned by <code>getClass()</code>	343
LCK03-J. Do not synchronize on the intrinsic locks of high-level concurrency objects	347
LCK04-J. Do not synchronize on a collection view if the backing collection is accessible	348

LCK05-J. Synchronize access to static fields that can be modified by untrusted code	351
LCK06-J. Do not use an instance lock to protect shared static data	352
LCK07-J. Avoid deadlock by requesting and releasing locks in the same order	355
LCK08-J. Ensure actively held locks are released on exceptional conditions	365
LCK09-J. Do not perform operations that can block while holding a lock	370
LCK10-J. Do not use incorrect forms of the double-checked locking idiom	375
LCK11-J. Avoid client-side locking when using classes that do not commit to their locking strategy	381
Chapter 11 Thread APIs (THI)	387
Rules	387
Risk Assessment Summary	387
THI00-J. Do not invoke <code>Thread.run()</code>	388
THI01-J. Do not invoke <code>ThreadGroup</code> methods	390
THI02-J. Notify all waiting threads rather than a single thread	394
THI03-J. Always invoke <code>wait()</code> and <code>await()</code> methods inside a loop	401
THI04-J. Ensure that threads performing blocking operations can be terminated	404
THI05-J. Do not use <code>Thread.stop()</code> to terminate threads	412
Chapter 12 Thread Pools (TPS)	417
Rules	417
Risk Assessment Summary	417
TPS00-J. Use thread pools to enable graceful degradation of service during traffic bursts	418
TPS01-J. Do not execute interdependent tasks in a bounded thread pool	421
TPS02-J. Ensure that tasks submitted to a thread pool are interruptible	428
TPS03-J. Ensure that tasks executing in a thread pool do not fail silently	431
TPS04-J. Ensure <code>ThreadLocal</code> variables are reinitialized when using thread pools	436

Chapter 13	Thread-Safety Miscellaneous (TSM)	441
	Rules	441
	Risk Assessment Summary	441
	TSM00-J. Do not override thread-safe methods with methods that are not thread-safe	442
	TSM01-J. Do not let the <code>this</code> reference escape during object construction	445
	TSM02-J. Do not use background threads during class initialization	454
	TSM03-J. Do not publish partially initialized objects	459
Chapter 14	Input Output (FIO)	467
	Rules	467
	Risk Assessment Summary	468
	FIO00-J. Do not operate on files in shared directories	468
	FIO01-J. Create files with appropriate access permissions	478
	FIO02-J. Detect and handle file-related errors	481
	FIO03-J. Remove temporary files before termination	483
	FIO04-J. Close resources when they are no longer needed	487
	FIO05-J. Do not expose buffers created using the <code>wrap()</code> or <code>duplicate()</code> methods to untrusted code	493
	FIO06-J. Do not create multiple buffered wrappers on a single <code>InputStream</code>	496
	FIO07-J. Do not let external processes block on input and output streams	500
	FIO08-J. Use an <code>int</code> to capture the return value of methods that read a character or byte	504
	FIO09-J. Do not rely on the <code>write()</code> method to output integers outside the range 0 to 255	507
	FIO10-J. Ensure the array is filled when using <code>read()</code> to fill an array	509
	FIO11-J. Do not attempt to read raw binary data as character data	511
	FIO12-J. Provide methods to read and write little-endian data	513
	FIO13-J. Do not log sensitive information outside a trust boundary	516
	FIO14-J. Perform proper cleanup at program termination	519
Chapter 15	Serialization (SER)	527
	Rules	527
	Risk Assessment Summary	528

SER00-J. Maintain serialization compatibility during class evolution	528
SER01-J. Do not deviate from the proper signatures of serialization methods	531
SER02-J. Sign then seal sensitive objects before sending them across a trust boundary	534
SER03-J. Do not serialize unencrypted, sensitive data	541
SER04-J. Do not allow serialization and deserialization to bypass the security manager	546
SER05-J. Do not serialize instances of inner classes	549
SER06-J. Make defensive copies of private mutable components during deserialization	551
SER07-J. Do not use the default serialized form for implementation-defined invariants	553
SER08-J. Minimize privileges before deserializing from a privileged context	558
SER09-J. Do not invoke overridable methods from the <code>readObject()</code> method	562
SER10-J. Avoid memory and resource leaks during serialization	563
SER11-J. Prevent overwriting of externalizable objects	566

Chapter 16 Platform Security (SEC) 569

Rules	569
Risk Assessment Summary	570
SEC00-J. Do not allow privileged blocks to leak sensitive information across a trust boundary	570
SEC01-J. Do not allow tainted variables in privileged blocks	574
SEC02-J. Do not base security checks on untrusted sources	577
SEC03-J. Do not load trusted classes after allowing untrusted code to load arbitrary classes	579
SEC04-J. Protect sensitive operations with security manager checks	582
SEC05-J. Do not use reflection to increase accessibility of classes, methods, or fields	585
SEC06-J. Do not rely on the default automatic signature verification provided by <code>URLClassLoader</code> and <code>java.util.jar</code>	592
SEC07-J. Call the superclass's <code>getPermissions()</code> method when writing a custom class loader	597
SEC08-J. Define wrappers around native methods	599

Chapter 17	Runtime Environment (ENV)	603
	Rules	603
	Risk Assessment Summary	603
	ENV00-J. Do not sign code that performs only unprivileged operations	604
	ENV01-J. Place all security-sensitive code in a single jar and sign and seal it	606
	ENV02-J. Do not trust the values of environment variables	610
	ENV03-J. Do not grant dangerous combinations of permissions	613
	ENV04-J. Do not disable bytecode verification	617
	ENV05-J. Do not deploy an application that can be remotely monitored	618
Chapter 18	Miscellaneous (MSC)	625
	Rules	625
	Risk Assessment Summary	625
	MSC00-J. Use <code>SSLSocket</code> rather than <code>Socket</code> for secure data exchange	626
	MSC01-J. Do not use an empty infinite loop	630
	MSC02-J. Generate strong random numbers	632
	MSC03-J. Never hard code sensitive information	635
	MSC04-J. Do not leak memory	638
	MSC05-J. Do not exhaust heap space	647
	MSC06-J. Do not modify the underlying collection when an iteration is in progress	653
	MSC07-J. Prevent multiple instantiations of singleton objects	657
	Glossary	669
	References	677
	Index	693