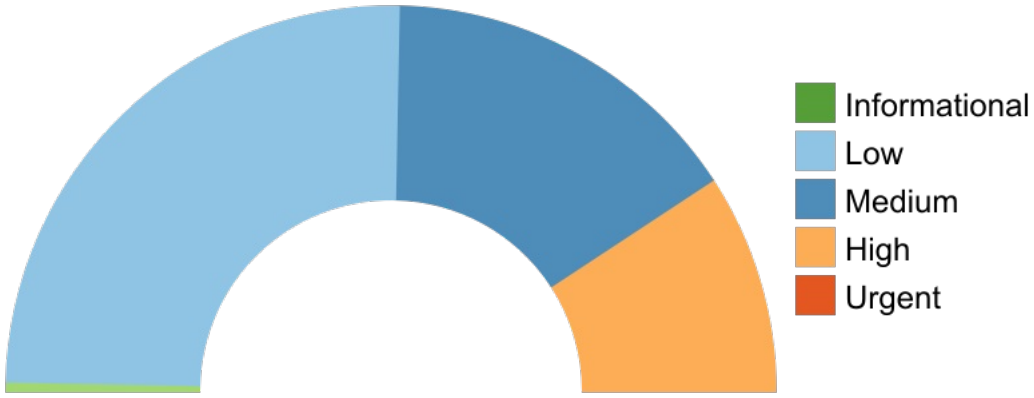
Checks by Severity



Checks

Check ID	Check Name	Supported	Severity
AC_00	No Control Code Characters	Yes	
AC_01	No Direct or Indirect Recursion	Yes	
AC_HIS_04	Cyclomatic Complexity (v(G))	Yes	
AC_HIS_13	Statements Changed (SCHG)	Yes	
AC_HIS_14	Statements Deleted (SDEL)	Yes	
AC_HIS_15	New Statements (SNEW)	Yes	
AC_HIS_16	Stability Index (S)	Yes	
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 permissions	Yes	High
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	Yes	High

	debugging entry points		
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
HIS_03	3. Number of Goto Statements(GOTO)	Yes	
HIS_04	4. Cyclomatic Complexity (v(G))	Yes	
HIS_13	13. Statements Changed (SCHG)	Yes	
HIS_14	14. Statements Deleted (SDEL)	Yes	
HIS_15	15. New Statements (SNEW)	Yes	
HIS_16	16. Stability Index (S)	Yes	
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
JAVA_01	All fixed values will be defined final.	Yes	
JAVA_02	Unused Instance Variables.	Yes	
JAVA_03	Single exit point at end	Yes	
JAVA_04	Defined methods shall be called at least once.	Yes	
JAVA_05	Unused Local Variables	Yes	
JAVA_06	Package names shall be in all lowercase alphabetic characters and integers	Yes	
JAVA_07	If multiple classes are imported, the list shall be imported alphabetically	Yes	
JAVA_08	Each package group shall be separated with a newline	Yes	
JAVA_09	Each Java class shall be assigned to a named package	Yes	
JAVA_10	Use package names instead of prefixes on class identifiers	Yes	
JAVA_11	A class shall be declared an interface if no method of the class is implemented.	Yes	
JAVA_12	If a constructor catches an exception that causes the failure of	Yes	

	the object, that exception shall be thrown.		
JAVA_13	Order class definitions by scope	Yes	
JAVA_14	Capitalize classes and interfaces	Yes	
JAVA_15	Method names should be camelCase	Yes	
JAVA_16	Prepend method names with "get" if they return data member values	Yes	
JAVA_17	Prepend method names with "set" if they assign data member values	Yes	
JAVA_18	Any method that tests the truth or falsity of a condition shall be prepended with a boolean expression	Yes	
JAVA_19	A method definition statement and member (nested) class definition statement shall begin at the standard indentation relative to the enclosing class definition statement.	Yes	
JAVA_20	The last call of the finalizer shall be to super.finalize(), unless Object is the immediate superclass	Yes	
JAVA_21	Finalizers shall catch and manage their own exceptions as well as any propagated exceptions that may be thrown from functions called by the finalizer.	Yes	
JAVA_22	A variable's use shall not be redefined within a method.	Yes	
JAVA_23	Only one variable shall be specified for every declaration key word.	Yes	
JAVA_24	Each new block shall be indented one increment further than its parent block.	Yes	
JAVA_25	Each statement shall be indented to the level of its block.	Yes	
JAVA_26	There shall be at most one statement per line.	Yes	
JAVA_27	A wrapped line shall be indented one increment further than its	Yes	

	originating line.		
JAVA_28	A new instance of java.lang.Exception shall not be thrown.	Yes	
JAVA_29	Braces shall be used for all control structures, even if there is only one statement.	Yes	
JAVA_30	White space before a comma or semicolon shall not be used.	Yes	
JAVA_31	The loop control variable shall not be modified in the body of a for loop.	Yes	
JAVA_32	The last choice of a switch or case statement shall end with a break statement.	Yes	
JAVA_33	Statements under case labels shall be indented one level.	Yes	
JAVA_34	In a switch statement, when a default case is presented, it shall be the last case.	Yes	
JAVA_35	For a do-while loop, the ending brace shall be on the same line as the while.	Yes	
JAVA_36	Ternary operators shall not be nested inside other ternary operators.	Yes	
JAVA_37	Calculations that resolve to the same value shall not be performed inside a loop.	Yes	
JAVA_38	For the control structures the terminating brace shall appear on a separate line at the same indentation as the initiating keyword.	Yes	
JAVA_39	White space shall not be used: between the name of an array and open bracket that introduces its index (e.g., array[i]), or between unary operators and the objects they operate on (e.g., -1).	Yes	
JAVA_40	Spaces shall be placed after	Yes	

	commas.		
JAVA_41	Any method that returns the object converted to another type shall be prepended with the word "to" (e.g., c.toString()).	Yes	
JAVA_42	The end of a closing brace shall be at the same indentation as the blocks' declaring line.	Yes	
JAVA_43	Method declarations shall have at least one blank line between them to improve readability.	Yes	
JAVA_44	Spaces shall be placed around all binary operators.	Yes	
JAVA_DCL00	Prevent class initialization cycles	Yes	
JAVA_DCL02	Do not modify the collection's elements during an enhanced for statement	Yes	
JAVA_ENV02	Do not trust the values of environment variables	Yes	
JAVA_ENV03	Do not grant dangerous combinations of permissions	Yes	
JAVA_ENV06	Production code must not contain debugging entry points	Yes	
JAVA_ERR00	Do not suppress or ignore checked exceptions	Yes	
JAVA_ERR01	Do not allow exceptions to expose sensitive information	Yes	
JAVA_ERR02	Prevent exceptions while logging data	Yes	
JAVA_ERR03	Restore prior object state on method failure	Yes	
JAVA_ERR04	Do not complete abruptly from a finally block	Yes	
JAVA_ERR05	Do not let checked exceptions escape from a finally block	Yes	
JAVA_ERR06	Do not throw undeclared checked exceptions	Yes	
JAVA_ERR07	Do not throw RuntimeException, Exception, or Throwable	Yes	
JAVA_ERR08	Do not catch NullPointerException or any of its ancestors	Yes	

JAVA_ERR09	Do not allow untrusted code to terminate the JVM	Yes	
JAVA_EXP00	Do not ignore values returned by methods.	Yes	
JAVA_EXP02	Do not use the Object.equals() method to compare two arrays.	Yes	
JAVA_EXP03	Do not use the equality operators when comparing values of boxed primitives	Yes	
JAVA_EXP04	Do not pass arguments to certain Java Collections Framework methods that are a different type than the collection parameter type	Yes	
JAVA_EXP05	Do not follow a write by a subsequent write or read of the same object within an expression	Yes	
JAVA_EXP06	Expressions used in assertions must not produce side effects	Yes	
JAVA_FIO00	Do not operate on files in shared directories	Yes	
JAVA_FIO01	Create files with appropriate access permissions	Yes	
JAVA_FIO02	Detect and handle file-related errors	Yes	
JAVA_FIO03	Remove temporary files before termination	Yes	
JAVA_FIO04	Release resources when they are no longer needed	Yes	
JAVA_FIO05	Do not expose buffers or their backing arrays methods to untrusted code	Yes	
JAVA_FIO09	Do not rely on the write() method to output integers outside the range 0 to 255	Yes	
JAVA_FIO10	Ensure the array is filled when using read() to fill an array	Yes	
JAVA_FIO13	Do not log sensitive information outside a trust boundary	Yes	
JAVA_FIO14	Perform proper cleanup at program termination	Yes	
JAVA_IDS00	Prevent SQL Injection	Yes	

JAVA_IDS01	Normalize strings before validating them	Yes	
JAVA_IDS04	Safely extract files from ZipInputStream	Yes	
JAVA_IDS06	Exclude unsanitized user input from format strings	Yes	
JAVA_IDS08	Sanitize untrusted data included in a regular expression	Yes	
JAVA_IDS11	Perform any string modifications before validation	Yes	
JAVA_IDS16	Prevent XML Injection	Yes	
JAVA_JNI00	Define wrappers around native methods	Yes	
JAVA_LCK00	Use private final lock objects to synchronize classes that may interact with untrusted code	Yes	
JAVA_LCK01	Do not synchronize on objects that may be reused	Yes	
JAVA_LCK02	Do not synchronize on the class object returned by getClass()	Yes	
JAVA_LCK04	Do not synchronize on a collection view if the backing collection is accessible	Yes	
JAVA_LCK05	Synchronize access to static fields that can be modified by untrusted code	Yes	
JAVA_LCK06	Do not use an instance lock to protect shared static data	Yes	
JAVA_LCK07	Avoid deadlock by requesting and releasing locks in the same order	Yes	
JAVA_LCK08	Ensure actively held locks are released on exceptional conditions	Yes	
JAVA_LCK09	Do not perform operations that can block while holding a lock	Yes	
JAVA_LCK10	Use a correct form of the double-checked locking idiom	Yes	
JAVA_LCK11	Avoid client-side locking when using classes that do not commit to their locking strategy	Yes	
JAVA_MET00	Validate method arguments	Yes	
JAVA_MET01	Never use assertions to validate	Yes	

	method arguments		
JAVA_MET02	Do not use deprecated or obsolete classes or methods	Yes	
JAVA_MET03	Methods that perform a security check must be declared private or final.	Yes	
JAVA_MET04	Do not increase the accessibility of overridden or hidden methods	Yes	
JAVA_MET05	Ensure that constructors do not call overridable methods	Yes	
JAVA_MET06	Do not invoke overridable methods in clone()	Yes	
JAVA_MET07	Never declare a class method that hides a method declared in a superclass or superinterface	Yes	
JAVA_MET08	Preserve the equality contract when overriding the equals() method	Yes	
JAVA_MET09	Classes that define an equals() method must also define a hashCode() method	Yes	
JAVA_MET10	Follow the general contract when implementing the compareTo() method	Yes	
JAVA_MET11	Ensure that keys used in comparison operations are immutable	Yes	
JAVA_MET12	Do not use finalizers	Yes	
JAVA_MSC00	Use SSLSocket rather than Socket for secure data exchange	Yes	
JAVA_MSC01	Do not use an empty infinite loop	Yes	
JAVA_MSC02	Generate strong random numbers	Yes	
JAVA_N000	Naming Convention: Classes	Yes	
JAVA_N001	Naming Convention: Files	Yes	
JAVA_N002	Naming Convention: Interface	Yes	
JAVA_N003	Naming Convention: Methods	Yes	
JAVA_N004	Naming Convention: Packages	Yes	
JAVA_N005	Naming Convention: Parameters	Yes	
JAVA_N006	Naming Convention: Variables	Yes	
JAVA_N007	Constants shall be in all uppercase, with underscores separating each	Yes	

	component word		
JAVA_NUM00	Detect or prevent integer overflow	Yes	
JAVA_NUM02	Ensure that division and remainder operations do not result in divide-by-zero errors	Yes	
JAVA_NUM07	Do not attempt comparisons with NaN	Yes	
JAVA_NUM09	Do not use floating-point variables as loop counters	Yes	
JAVA_NUM10	Do not construct BigDecimal objects from floating-point literals	Yes	
JAVA_NUM11	Do not compare or inspect the string representation of floating-point values	Yes	
JAVA_NUM12	Ensure conversions of numeric types to narrower types do not result in lost or misinterpreted data	Yes	
JAVA_NUM13	Avoid loss of precision when converting primitive integers to floating-point	Yes	
JAVA_OBJ01	Limit accessibility of fields	Yes	
JAVA_OBJ04	Provide mutable classes with copy functionality to safely allow passing instances to untrusted code	Yes	
JAVA_OBJ05	Do not return references to private mutable class members	Yes	
JAVA_OBJ07	Sensitive classes must not let themselves be copied	Yes	
JAVA_OBJ08	Do not expose private members of an outer class from within a nested class	Yes	
JAVA_OBJ09	Compare classes and not class names	Yes	
JAVA_OBJ10	Do not use public static nonfinal fields	Yes	
JAVA_OBJ11	Be wary of letting constructors throw exceptions	Yes	
JAVA_OBJ13	Ensure that references to mutable objects are not exposed	Yes	
JAVA_SEC01	Do not allow tainted variables in privileged blocks	Yes	

JAVA_SEC07	Call the superclass's getPermissions() method when writing a custom class loader	Yes	
JAVA_SER01	Do not deviate from the proper signatures of serialization methods	Yes	
JAVA_SER04	Do not allow serialization and deserialization to bypass the security manager	Yes	
JAVA_SER05	Do not serialize instances of inner classes	Yes	
JAVA_SER06	Make defensive copies of private mutable components during deserialization	Yes	
JAVA_SER07	Do not use the default serialized form for classes with implementation-defined invariants	Yes	
JAVA_SER09	Do not invoke overridable methods from the readObject() method	Yes	
JAVA_SER12	Prevent deserialization of untrusted data	Yes	
JAVA_STR01	Do not assume that a Java char fully represents a Unicode code point	Yes	
JAVA_STR03	Do not encode noncharacter data as a string	Yes	
JAVA_THI00	Do not invoke Thread.run()	Yes	
JAVA_THI01	Do not invoke ThreadGroup methods	Yes	
JAVA_THI02	Notify all waiting threads rather than a single thread.	Yes	
JAVA_THI03	Always invoke wait() and await() methods inside a loop	Yes	
JAVA_THI04	Ensure that threads performing blocking operations can be terminated	Yes	
JAVA_THI05	Do not use Thread.stop() to terminate threads.	Yes	
JAVA_TPS00	Use thread pools to enable graceful degradation of service during traffic bursts	Yes	
JAVA_TPS01	Do not execute interdependent	Yes	

	tasks in a bounded thread pool		
JAVA_TPS02	Ensure that tasks submitted to a thread pool are interruptible	Yes	
JAVA_TPS03	Ensure that tasks executing in a thread pool do not fail silently	Yes	
JAVA_TPS04	Ensure ThreadLocal variables are reinitialized when using thread pools	Yes	
JAVA_TSM00	Do not override thread-safe methods with methods that are not thread-safe	Yes	
JAVA_TSM01	Do not let the this reference escape during object construction	Yes	
JAVA_TSM02	Do not use background threads during class initialization	Yes	
JAVA_TSM03	Do not publish partially initialized objects	Yes	
JAVA_VNA00	Ensure visibility when accessing shared primitive variables	Yes	
JAVA_VNA01	Ensure visibility of shared references to immutable objects	Yes	
JAVA_VNA02	Ensure that compound operations on shared variables are atomic	Yes	
JAVA_VNA03	Do not assume that a group of calls to independently atomic methods is atomic	Yes	
JAVA_VNA04	Ensure that calls to chained methods are atomic	Yes	
JAVA_VNA05	Ensure atomicity when reading and writing 64-bit values	Yes	
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	No	Medium

	objects		
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()	Yes	Low

	method must also define a hashCode() method		
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
METRIC_00	Program Unit Call Count	Yes	
METRIC_01	Program Unit Callby Count	Yes	
METRIC_02	Program Unit Comment to Code Ratio	Yes	
METRIC_03	Program Unit Cyclomatic Complexity	Yes	
METRIC_04	Program Unit Max Length	Yes	
METRIC_05	Program Unit Max Nesting Depth	Yes	
METRIC_06	Program Unit Parameters Count	Yes	
METRIC_07	Program Unit Path Count	Yes	
METRIC_08	Program Unit Statement Count	Yes	
METRIC_09	Coupling Between Object Classes	Yes	
METRIC_13	Maintainability Index	Yes	
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	No	Medium

	data		
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
RECOMMENDED_04	Functions Too Long	Yes	
RECOMMENDED_06	Goto Statements	Yes	
RECOMMENDED_10	Overly Complex Functions	Yes	
RECOMMENDED_12	Unreachable Code	Yes	
RECOMMENDED_13	Unused Functions	Yes	
RECOMMENDED_18	Unused Local Variables	Yes	
RECOMMENDED_19	Comments Indicating Future Fixes	Yes	
RECOMMENDED_20	Duplicate Code	Yes	
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 bounded thread pool	Yes	Low
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	Yes	Medium

	on shared variables are atomic		
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