




 Secrets


 ABAP


 Apex


 C


 C++


 CloudFormation


 COBOL


 C#


 CSS


 Flex


 Go


 HTML


 **Java**


 JavaScript


 Kotlin


 Objective C


 PHP


 PL/I


 PL/SQL


 Python


 RPG


 Ruby


 Scala


 Swift


 Terraform


 Text


 TypeScript

 T-SQL

 VB.NET

 VB6

 XML



Java static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVA code

All rules632

Vulnerability53

Bug154

Security Hotspot36


Code Smell389

Quick Fix42


Tags ▾

Search by name... 🔍


assignment

 Bug


Ints and longs should not be shifted by zero or more than their number of bits-1

 Bug


"compareTo" should not return "Integer.MIN_VALUE"

 Bug


Boxing and unboxing should not be immediately reversed

 Bug


"equals(Object obj)" should test argument type

 Bug


"Serializable" inner classes of non-serializable classes should be "static"

 Bug


The non-serializable super class of a "Serializable" class should have a no-argument constructor

 Bug


Method parameters, caught exceptions and foreach variables' initial values should not be ignored

 Bug


"equals(Object obj)" and "hashCode()" should be overridden in pairs

 Bug





Disclosing fingerprints from web application technologies is security-sensitive

 Security Hotspot

Having a permissive Cross-Origin Resource Sharing policy is security-sensitive

 Security Hotspot

Return values from functions without side effects should not be ignored

 Bug  Major   cert





When the call to a function doesn't have any side effects, what is the point of making the call if the results are ignored? In such case, either the function call is useless and should be dropped or the source code doesn't behave as expected.

To prevent generating any false-positives, this rule triggers an issue only on the following predefined list of immutable classes in the Java API :

- java.lang.String
- java.lang.Boolean
- java.lang.Integer
- java.lang.Double
- java.lang.Float
- java.lang.Byte
- java.lang.Character
- java.lang.Short
- java.lang.StackTraceElement
- java.time.DayOfWeek
- java.time.Duration
- java.time.Instant
- java.time.LocalDate
- java.time.LocalDateTime
- java.time.LocalTime
- java.time.Month
- java.time.MonthDay
- java.time.OffsetDateTime
- java.time.OffsetTime
- java.time.Period
- java.time.Year
- java.time.YearMonth
- java.time.ZonedDateTime
- java.math.BigInteger
- java.math.BigDecimal
- java.util.Optional

As well as methods of the following classes:

- java.util.Collection:
 - size()
 - isEmpty()
 - contains(...)
 - containsAll(...)
 - iterator()
 - toArray()
- java.util.Map:
 - size()
 - isEmpty()
 - containsKey(...)
 - containsValue(...)
 - get(...)
 - getOrDefault(...)
 - keySet()
 - entrySet()

Delivering code in production with debug features activated is security-sensitive  Security Hotspot
Searching OS commands in PATH is security-sensitive  Security Hotspot
Allowing both safe and unsafe HTTP methods is security-sensitive  Security Hotspot
Creating cookies without the "HttpOnly" flag is security-sensitive 

- values()
- java.util.stream.Stream
 - toArray
 - reduce
 - collect
 - min
 - max
 - count
 - anyMatch
 - allMatch
 - noneMatch
 - findFirst
 - findAny
 - toList

Noncompliant Code Example

```
public void handle(String command){
    command.toLowerCase(); // Noncompliant; result of method t
    ...
}
```

Compliant Solution

```
public void handle(String command){
    String formattedCommand = command.toLowerCase();
    ...
}
```

Exceptions

This rule will not raise an issue when both these conditions are met:

- The method call is in a try block with an associated catch clause.
- The method name starts with "parse", "format", "decode" or "valueOf" or the method is String.getBytes(Charset).

```
private boolean textIsInteger(String textToCheck) {

    try {
        Integer.parseInt(textToCheck, 10); // OK
        return true;
    } catch (NumberFormatException ignored) {
        return false;
    }
}
```

See

- [CERT, EXP00-J](#). - Do not ignore values returned by methods

Available In:

