
































-  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 rules 632

 Vulnerability 53


 Bug 154













 Security Hotspot 36

 Code Smell 389

 Quick Fix 42

Tags ▾

Search by name... 

	Code Smell
	JUnit test cases should call super methods
	Code Smell
	TestCases should contain tests
	Code Smell
	Short-circuit logic should be used in boolean contexts
	Code Smell
	Methods and field names should not be the same or differ only by capitalization
	Code Smell
	Switch cases should end with an unconditional "break" statement
	Code Smell
	"switch" statements should not contain non-case labels
	Code Smell
	Future keywords should not be used as names
	Code Smell
	Thread suspensions should not be vulnerable to Denial of Service attacks
	Vulnerability
	A new session should be created during user authentication
	Vulnerability
	JWT should be signed and verified with strong cipher algorithms
	Vulnerability
	Cipher algorithms should be robust
	Vulnerability
	Encryption algorithms should be used

Files opened in append mode should not be used with ObjectOutputStream

Analyze your code

 Bug

 Blocker

 ?

 serialization

ObjectOutputStreams are used with serialization, and the first thing an ObjectOutputStream writes is the serialization stream header. This header should appear once per file, at the beginning. Pass a file opened in append mode into an ObjectOutputStream constructor and the serialization stream header will be added to the end of the file before your object is then also appended.

When you're trying to read your object(s) back from the file, only the first one will be read successfully, and a StreamCorruptedException will be thrown after that.

Noncompliant Code Example

```
FileOutputStream fos = new FileOutputStream (fileName , true
ObjectOutputStream out = new ObjectOutputStream(fos); // No
```

Compliant Solution

```
FileOutputStream fos = new FileOutputStream (fileName);
ObjectOutputStream out = new ObjectOutputStream(fos);
```





Available In:

 sonarlint

 sonarcloud

 sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.  
[Privacy Policy](#)

<b>with secure mode and padding scheme</b>  Vulnerability
<b>Server hostnames should be verified during SSL/TLS connections</b>  Vulnerability
<b>Insecure temporary file creation methods should not be used</b>  Vulnerability
<b>Passwords should not be stored in plain-text or with a fast hashing algorithm</b>  Vulnerability