




 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... 🔍

Using clear-text protocols is security-sensitive

Security Hotspot

Accessing Android external storage is security-sensitive

Security Hotspot

Receiving intents is security-sensitive

Security Hotspot

Broadcasting intents is security-sensitive

Security Hotspot

Expanding archive files without controlling resource consumption is security-sensitive

Security Hotspot

Configuring loggers is security-sensitive

Security Hotspot

Using weak hashing algorithms is security-sensitive

Security Hotspot

Using unsafe Jackson deserialization configuration is security-sensitive

Security Hotspot

Setting JavaBean properties is security-sensitive

Security Hotspot

Disabling CSRF protections is security-sensitive

Security Hotspot

Using non-standard cryptographic algorithms is security-sensitive

Security Hotspot

Using pseudorandom number generators (PRNGs) is security-sensitive

Security Hotspot

Insecure temporary file creation methods should not be used

Analyze your code

Vulnerability Critical cwe owasp

Using `File.createTempFile` as the first step in creating a temporary directory causes a race condition and is inherently unreliable and insecure. Instead, `Files.createTempDirectory` (Java 7+) should be used.

This rule raises an issue when the following steps are taken in immediate sequence:

- call to `File.createTempFile`
- delete resulting file
- call `mkdir` on the File object

Note that this rule is automatically disabled when the project's `sonar.java.source` is lower than 7.

Noncompliant Code Example

```
File tempDir;
tempDir = File.createTempFile("", ".");
tempDir.delete();
tempDir.mkdir(); // Noncompliant
```

Compliant Solution

```
Path tempPath = Files.createTempDirectory("");
File tempDir = tempPathToFile();
```

See

- [OWASP Top 10 2021 Category A1](#) - Broken Access Control
- [OWASP Top 10 2017 Category A9](#) - Using Components with Known Vulnerabilities
- [MITRE, CWE-377](#) - Insecure Temporary File
- [MITRE, CWE-379](#) - Creation of Temporary File in Directory with Incorrect Permissions
- [OWASP, Insecure Temporary File](#)





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](#)

https://rules.sonarsource.com/java/RSPEC-5445

1/2

<div>generators (i.e. Files) is security sensitive</div> <div> Security Hotspot</div>
<div>Mocking all non-private methods of a class should be avoided</div> <div> Code Smell</div>
<div>Empty lines should not be tested with regex MULTILINE flag</div> <div> Code Smell</div>
<div>Methods setUp() and tearDown() should be correctly annotated starting with JUnit4</div> <div> Code Smell</div>