



 **sonar** RULES


Products ▾


 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... 🔍

should not be too high

Code Smell

Factory method injection should be used in "@Configuration" classes

Code Smell

"static" base class members should not be accessed via derived types

Code Smell

Instance methods should not write to "static" fields

Code Smell

"indexOf" checks should not be for positive numbers

Code Smell

Method overrides should not change contracts

Code Smell

Whitespace and control characters in literals should be explicit

Code Smell

Null should not be returned from a "Boolean" method

Code Smell

Classes should not access their own subclasses during initialization

Code Smell

"Object.wait(...)" and "Condition.await(...)" should be called inside a "while" loop

Code Smell

IllegalMonitorStateException should not be caught

Code Smell

JUnit assertions should not be used in "run" methods

**Hashes should include an unpredictable salt**

Analyze your code

VulnerabilityCritical🔍cwe sans-top25 owasp

In cryptography, a "salt" is an extra piece of data which is included when hashing a password. This makes rainbow-table attacks more difficult. Using a cryptographic hash function without an unpredictable salt increases the likelihood that an attacker could successfully find the hash value in databases of precomputed hashes (called rainbow-tables).

This rule raises an issue when a hashing function which has been specifically designed for hashing passwords, such as PBKDF2, is used with a non-random, reused or too short salt value. It does not raise an issue on base hashing algorithms such as sha1 or md5 as they should not be used to hash passwords.

**Recommended Secure Coding Practices**

- Use hashing functions generating their own secure salt or generate a secure random value of at least 16 bytes.
- The salt should be unique by user password.

**Noncompliant Code Example**

Below, the hashed password use a predictable salt:

```
byte[] salt = "notrandom".getBytes();

PBEPParameterSpec cipherSpec = new PBEPParameterSpec(salt, 100
PBEKeySpec spec = new PBEKeySpec(chars, salt, 10000, 256); /
```

**Compliant Solution**

Use java.security.SecureRandom to generate an unpredictable salt:




```
SecureRandom random = new SecureRandom();
byte[] salt = new byte[16];
random.nextBytes(salt);

PBEPParameterSpec cipherSpec = new PBEPParameterSpec(salt, 100
PBEKeySpec spec = new PBEKeySpec(chars, salt, 10000, 256); /
```

**See**






- [OWASP Top 10 2021 Category A2](#) - Cryptographic Failures
- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure
- [MITRE, CWE-759](#) - Use of a One-Way Hash without a Salt
- [MITRE, CWE-760](#) - Use of a One-Way Hash with a Predictable Salt
- [SANS Top 25](#) - Porous Defenses

Available In:

 |  | 

https://rules.sonarsource.com/java/RSPEC-2053

1/2

 Code Smell
<b>Class names should not shadow interfaces or superclasses</b>  Code Smell
<b>"Cloneables" should implement "clone"</b>  Code Smell
<b>Try-with-resources should be used</b>  Code Smell
<b>"readResolve" methods should be inheritable</b>  Code Smell

---

© 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](#)