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 rules 632 | Vulnerability 53 | Bug 154 | Security Hotspot 36 | Code Smell 389 | Quick Fix 42

Tags | Search by name...

not be left unused
Bug

All branches in a conditional structure should not have exactly the same implementation
Bug

Optional value should only be accessed after calling isPresent()
Bug

Overrides should match their parent class methods in synchronization
Bug

Value-based classes should not be used for locking
Bug

Expressions used in "assert" should not produce side effects
Bug

"volatile" variables should not be used with compound operators
Bug

"getClass" should not be used for synchronization
Bug

Min and max used in combination should not always return the same value
Bug

Assignment of lazy-initialized members should be the last step with double-checked locking
Bug

"String" calls should not go beyond their bounds
Bug

Using non-standard cryptographic algorithms is security-sensitive

Analyze your code

Security Hotspot Critical cwe sans-top25 owasp

The use of a non-standard algorithm is dangerous because a determined attacker may be able to break the algorithm and compromise whatever data has been protected. Standard algorithms like `SHA-256`, `SHA-384`, `SHA-512`, ... should be used instead.

This rule tracks creation of `java.security.MessageDigest` subclasses.

Recommended Secure Coding Practices

- Use a standard algorithm instead of creating a custom one.

Sensitive Code Example

```
public class MyCryptographicAlgorithm extends MessageDigest
  ...
}
```

Compliant Solution

```
MessageDigest digest = MessageDigest.getInstance("SHA-256");
```

See

- OWASP Top 10 2021 Category A2 - Cryptographic Failures
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- MITRE, CWE-327 - Use of a Broken or Risky Cryptographic Algorithm
- SANS Top 25 - Porous Defenses
- Derived from FindSecBugs rule MessageDigest is Custom

Available In:
sonarcloud | sonarqube

Raw byte values should not be used in bitwise operations in combination with shifts

Bug

Getters and setters should be synchronized in pairs

Bug

Non-thread-safe fields should not be static

Bug

"null" should not be used with "Optional"

Bug