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...

- Exceptions should not be created without being thrown — Bug
- Collection sizes and array length comparisons should make sense — Bug
- Consumed Stream pipelines should not be reused — Bug
- Intermediate Stream methods should 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

Setting JavaBean properties is security-sensitive

Analyze your code

Security Hotspot   Critical   cwe owasp cert

Setting JavaBean properties is security sensitive. Doing it with untrusted values has led in the past to the following vulnerability:

- CVE-2014-0114

JavaBeans can have their properties or nested properties set by population functions. An attacker can leverage this feature to push into the JavaBean malicious data that can compromise the software integrity. A typical attack will try to manipulate the ClassLoader and finally execute malicious code.

This rule raises an issue when:

- BeanUtils.populate(...) or BeanUtilsBean.populate(...) from Apache Commons BeanUtils are called
- BeanUtils.setProperty(...) or BeanUtilsBean.setProperty(...) from Apache Commons BeanUtils are called
- org.springframework.beans.BeanWrapper.setPropertyValue(...) or org.springframework.beans.BeanWrapper.setPropertyValues(...) from Spring is called

Ask Yourself Whether

- the new property values might have been tampered with or provided by an untrusted source.
- sensitive properties can be modified, for example: `class.classLoader`

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

Sanitize all values used as JavaBean properties.

Don't set any sensitive properties. Keep full control over which properties are set. If the property names are provided by an unstrusted source, filter them with a whitelist.

Sensitive Code Example

```
Company bean = new Company();
HashMap map = new HashMap();
Enumeration names = request.getParameterNames();
while (names.hasMoreElements()) {
    String name = (String) names.nextElement();
    map.put(name, request.getParameterValues(name));
}
BeanUtils.populate(bean, map); // Sensitive: "map" is popula
```

See

- OWASP Top 10 2021 Category A3 - Injection
- OWASP Top 10 2021 Category A8 - Software and Data Integrity Failures
- OWASP Top 10 2017 Category A1 - Injection

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

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

- MITRE, CWE-915 - Improperly Controlled Modification of Dynamically-Determined Object Attributes
- CERT, MSC61-J. - Do not use insecure or weak cryptographic algorithms
- Derived from FindSecBugs rule BEAN_PROPERTY_INJECTION

Available In:

sonarcloud sonarqube