

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVA code

Quick Fix (42)

Search by name...



String offset-based methods should be preferred for finding substrings from offsets



"default" clauses should be last



"equals" method parameters should not be marked "@Nonnull"



A conditionally executed single line should be denoted by indentation



Conditionals should start on new lines



Cognitive Complexity of methods should not be too high



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



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



Instance methods should not write to "static" fields



"indexOf" checks should not be for positive numbers



Method overrides should not change contracts



Whitespace and control characters in

Analyze your code



- injection cwe owasp
- denial-of-service

Most of the regular expression engines use `backtracking` to try all possible execution paths of the regular expression when evaluating an input, in some cases it can cause performance issues, called `catastrophic backtracking` situations. In the worst case, the complexity of the regular expression is exponential in the size of the input, this means that a small carefully-crafted input (like 20 chars) can trigger `catastrophic backtracking` and cause a denial of service of the application. Super-linear regex complexity can lead to the same impact too with, in this case, a large carefully-crafted input (thousands chars).

The first regex evaluation will never end in OpenJDK <= 9 and the second regex evaluation will never end in any versions of OpenJDK:

```
java.util.regex.Pattern.compile("(+)+").matcher(
"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"+
"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"+
"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"+
aaaaaaaaaaaaaaa!").matches(); // Sensitive

java.util.regex.Pattern.compile("(h|h|ih((i|a|c|c|a|i|i|i|j|b|
"hchcchcihcchciicchihcchcihcchcihihchiciiihhchcih")+
"chhcchcihcchiihcichhcccihcchcihcchcihcchcihcchcihcchci
"chicihhcccihcchihhhchichichcihihiicchihcchcihcchcihcchci
"chhhhiihchihcchhhiiiiiiiccihcchcihcchcihcchhhchchcii
"ichicchhhihcchcihihcccihcchcihcchcihcchcihcchcihcchcih
"iihhhhihhihcchiihiiiihhhhhhchhchicchihihiiiiiihcchcihcch
```

It is not recommended to construct a regular expression pattern from a user-controlled input, if no other choice, sanitize the input to remove/annihilate regex metacharacters.

Noncompliant Code Example





```
public boolean validate(javax.servlet.http.HttpServletRequest
    String regex = request.getParameter("regex");
    String input = request.getParameter("input");

    input.matches(regex); // Noncompliant
}
```

Compliant Solution

```
public boolean validate(javax.servlet.http.HttpServletRequest
    String regex = request.getParameter("regex");
    String input = request.getParameter("input");

    input.matches(Pattern.quote(regex)); // Compliant : with
}
```


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

See


- [OWASP Top 10 2021 Category A3](#) - Injection
- [OWASP Top 10 2017 Category A1](#) - Injection
- [MITRE, CWE-20](#) - Improper Input Validation
- [MITRE, CWE-400](#) - Uncontrolled Resource Consumption
- [MITRE, CWE-1333](#) - Inefficient Regular Expression Complexity
- [OWASP Regular expression Denial of Service - ReDoS](#)

Available In:

sonarcloud



sonarqube



Developer Edition

© 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](#)