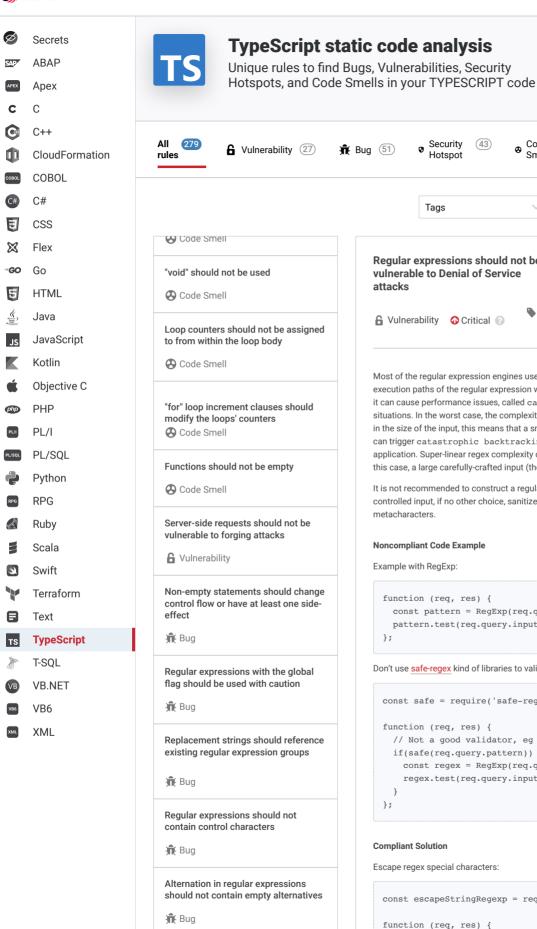


Products ✓



```
⊗ Code
                                                                                        O Quick 50 Fix
                                              Security
                                                        (43)
                                                                           (158)
6 Vulnerability (27)
                         # Bug (51)
                                             Hotspot
                                                                    Smell
                                             Tags
                                                                           Search by name.
                                 Regular expressions should not be
                                 vulnerable to Denial of Service
                                                                                 Analyze your code
                                 attacks
                                                                    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).
                                 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
                                 Example with RegExp:
                                   function (req, res) {
                                     const pattern = RegExp(req.query.pattern); // Noncompliant
                                     pattern.test(req.query.input);
                                   };
                                 Don't use safe-regex kind of libraries to validate regexes, as it is prone to FPs/FNs:
                                   const safe = require('safe-regex');
                                   function (reg, res) {
                                     // Not a good validator, eg of FN: http://test/noncomplian
                                     if(safe(req.query.pattern)) {
                                        const regex = RegExp(req.query.pattern); // Noncompliant
                                        regex.test(req.query.input);
                                   };
                                 Compliant Solution
                                 Escape regex special characters:
                                   const escapeStringRegexp = require('escape-string-regexp');
                                   function (reg, res) {
```

Mocha timeout should be disabled by

Unicode Granheme Clusters should be

setting it to "0".

📆 Bug

const pattern = RegExp(escapeStringRegexp(req.query.patter

pattern.test(req.query.input);

};

TypeScript static code analysis: Regular expressions should not be vulnerable to Denial of Service attacks

Oraphierne Oraștera anouiu pe avoided inside regex character # Bug Assertions should not be given twice the same argument 📆 Bug Alternatives in regular expressions should be grouped when used with anchors

Promise rejections should not be

caught by 'try' block

📆 Bug

Bug

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