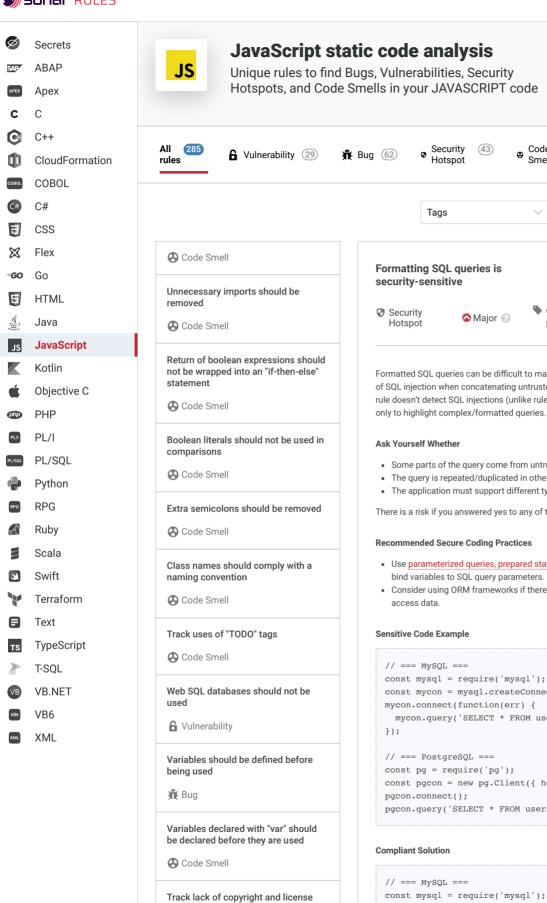
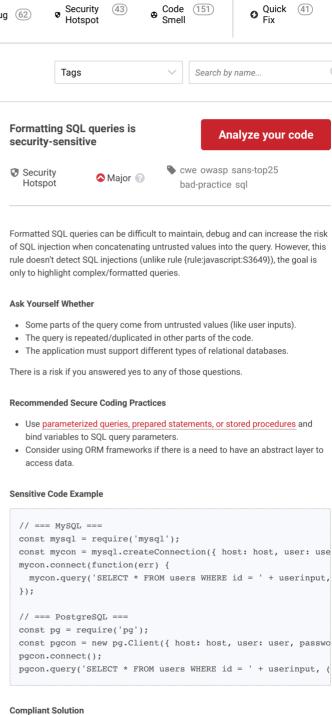


Products >

Q





const mycon = mysql.createConnection({ host: host, user: use

mycon.query('SELECT name FROM users WHERE id = ?', [userin

const pgcon = new pg.Client({ host: host, user: user, passwo

mycon.connect(function(err) {

// === PostgreSQL ===

const pg = require('pg');

});

headers

Code Smell

security-sensitive

Security Hotspot

Reading the Standard Input is

Using command line arguments is security-sensitive

Security Hotspot

Using Sockets is security-sensitive

Security Hotspot

Executing XPath expressions is security-sensitive

Security Hotspot

Encrypting data is security-sensitive

Security Hotspot

Using regular expressions is security-sensitive

pgcon.connect();
pgcon.query('SELECT name FROM users WHERE id = \$1', [userinp

Exceptions

This rule's current implementation does not follow variables. It will only detect SQL queries which are formatted directly in the function call.

const sql = 'SELECT * FROM users WHERE id = ' + userinput; mycon.query(sql, (err, res) => {}); // Sensitive but no issu

See

- OWASP Top 10 2021 Category A3 Injection
- OWASP Top 10 2017 Category A1 Injection
- <u>MITRE, CWE-89</u> Improper Neutralization of Special Elements used in an SQL Command
- MITRE, CWE-564 SQL Injection: Hibernate
- MITRE, CWE-20 Improper Input Validation
- <u>MITRE, CWE-943</u> Improper Neutralization of Special Elements in Data Query Logic
- SANS Top 25 Insecure Interaction Between Components
- Derived from FindSecBugs rules Potential SQL/JPQL Injection (JPA), Potential SQL/JDQQL Injection (JDO), Potential SQL/HQL Injection (Hibernate)

Available In:

sonarcloud 🙆 | sonarqube

© 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