





## Java static code analysis

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





Security Hotspot (36)

⊗ Code 389 Smell

O Quick 42

Analyze your code



NoSQL operations should not be

vulnerable to injection attacks



OS commands should not be vulnerable to command injection attacks

Vulnerability

Vulnerability

"@SpringBootApplication" and "@ComponentScan" should not be used in the default package

₩ Bug

"@Controller" classes that use "@SessionAttributes" must call "setComplete" on their "SessionStatus" objects

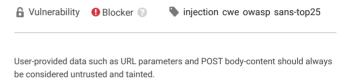
🖷 Bug

"wait" should not be called when multiple locks are held

i Bug

"PreparedStatement" and "ResultSet" methods should be called with valid indices

₩ Bug



Applications that perform NoSQL operations based on tainted data can be exploited similarly to regular SQL injection bugs. Depending on the code, the same risks exist as with SQL injections: The attacker aims to access sensitive information or compromise data integrity. Attacks may involve the injection of query operators,

This problem can be mitigated by using an Object Document Mapper (ODM) library or by validating user-supplied data based on its size or allowed characters.

## Noncompliant Code Example

JavaScript code, or string operations.

For the MongoDB Java Driver:

```
protected void doGet(HttpServletRequest req, HttpServletResp
    String input = req.getParameter("input");
    MongoClient mongoClient = new MongoClient();
                           = mongoClient.getDB("exampleData
    DB database
    DBCollection collection = database.getCollection("exampl
    BasicDBObject query
                          = new BasicDBObject();
    query.put("$where", "this.field == \"" + input + "\"");
}
```

## **Compliant Solution**

For the MongoDB Java Driver:

```
protected void doGet(HttpServletRequest req, HttpServletResp
    String input = req.getParameter("input");
    MongoClient mongoClient = new MongoClient();
    DB database
                           = mongoClient.getDB("ExampleData
    DBCollection collection = database.getCollection("exampl
    BasicDBObject query
                           = new BasicDBObject();
    query.put("field", input);
}
```

- OWASP Top 10 2021 Category A3 Injection
- OWASP Top 10 2017 Category A1 Injection
- MITRE, CWE-943 Improper Neutralization of Special Elements in Data Query

Files opened in append mode should not be used with ObjectOutputStream

👬 Bug

"wait(...)" should be used instead of "Thread.sleep(...)" when a lock is held

👬 Bug

Printf-style format strings should not lead to unexpected behavior at runtime

👬 Bug

Methods "wait(...)", "notify()" and "notifyAll()" should not be called on Thread instances

👬 Bug

- SANS Top 25 Insecure Interaction Between Components
- Morphia Java ODM

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