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...

Vulnerability

XML parsers should not be vulnerable to XXE attacks
Vulnerability

A secure password should be used when connecting to a database
Vulnerability

XPath expressions should not be vulnerable to injection attacks
Vulnerability

I/O function calls should not be vulnerable to path injection attacks
Vulnerability

LDAP queries should not be vulnerable to injection attacks
Vulnerability

OS commands should not be vulnerable to command injection attacks
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
Bug

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

NoSQL operations should not be vulnerable to injection attacks

Analyze your code

Vulnerability Blocker injection cwe owasp sans-top25

User-provided data such as URL parameters and POST body-content should always be considered untrusted and tainted.

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, JavaScript code, or string operations.

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

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("$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);
}
```

See

- 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 Logic

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