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

Endpoints should not be vulnerable to reflected cross-site scripting (XSS) attacks
Vulnerability

Database queries should not be vulnerable to injection attacks
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

Dynamic code execution should not be vulnerable to injection attacks

Analyze your code

Vulnerability | Blocker | injection cwe owasp sans-top25

Applications that execute code dynamically should neutralize any externally-provided values used to construct the code. Failure to do so could allow an attacker to execute arbitrary code. This could enable a wide range of serious attacks like accessing/modifying sensitive information or gain full system access.

The mitigation strategy should be based on whitelisting of allowed values or casting to safe types.

Noncompliant Code Example

```
protected void doGet(HttpServletRequest req, HttpServletResp
  String input = req.getParameter("input");

  ScriptEngineManager manager = new ScriptEngineManager();
  ScriptEngine engine = manager.getEngineByName("JavaScript"
  engine.eval(input); // Noncompliant
}
```

Compliant Solution

```
protected void doGet(HttpServletRequest req, HttpServletResp
  String input = req.getParameter("input");

  // Match the input against a whitelist
  if (!whiteList.contains(input))
    throw new IOException();

  ScriptEngineManager manager = new ScriptEngineManager();
  ScriptEngine engine = manager.getEngineByName("JavaScript"
  engine.eval(input);
}
```

See

- OWASP Top 10 2021 Category A3 - Injection
- OWASP Top 10 2017 Category A1 - Injection
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-95 - Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')
- SANS Top 25 - Risky Resource Management

Available In:

sonarcloud | sonarqube Developer Edition

multiple locks are held

Bug

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

Bug

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