

- 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...



Dynamic code execution should not be vulnerable to injection attacks

Vulnerability

NoSQL operations should not be vulnerable to injection attacks

Vulnerability

HTTP request redirections should not be open to forging attacks

Vulnerability

Deserialization should not be vulnerable to injection attacks

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

HTTP responses should not be vulnerable to session fixation

Analyze your code

Vulnerability Blocker injection cwe owasp

User-provided data, such as URL parameters, should always be considered untrusted and tainted. Constructing cookies directly from tainted data enables attackers to set the session identifier to a known value, allowing the attacker to share the session with the victim. Successful attacks might result in unauthorized access to sensitive information, for example if the session identifier is not regenerated when the victim authenticates.

Typically, the solution to prevent this type of attack is to restrict the cookies that can be influenced with an allow-list.

Noncompliant Code Example

```
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletResponse;
import org.springframework.web.bind.annotation.RequestMapping;

@RequestMapping(value = "/")
public void index(HttpServletResponse res, String value) {
    res.setHeader("Set-Cookie", value); // Noncompliant
    Cookie cookie = new Cookie("jsessionId", value); // Non
    res.addCookie(cookie);
}
```

Compliant Solution

```
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletResponse;
import org.springframework.web.bind.annotation.RequestMapping;

@RequestMapping(value = "/")
public void index(HttpServletResponse res, String value) {
    res.setHeader("X-Data", value);
    Cookie cookie = new Cookie("data", value);
    res.addCookie(cookie);
}
```


See

- [OWASP Top 10 2021 Category A3](#) - Injection
- [OWASP Top 10 2017 Category A1](#) - Injection
- [MITRE, CWE-20](#) - Improper Input Validation
- [MITRE, CWE-384](#) - Session Fixation

Available In:

sonarcloud sonarqube Developer Edition

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

© 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](#)