

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

JS

JavaScript static code analysis

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

All rules285

Vulnerability29

Bug62

Security Hotspot43

Code Smell151

Quick Fix41

Tags

Search by name...

HTTP responses should not be vulnerable to session fixation

Vulnerability

DOM updates should not lead to open redirect vulnerabilities

Vulnerability

Extracting archives should not lead to zip slip vulnerabilities

Vulnerability

DOM updates should not lead to cross-site scripting (XSS) attacks

Vulnerability

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

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

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

Vulnerability

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

```
module.exports.index = async function (req, res) {
  const value = req.query.value;

  res.setHeader("Set-Cookie", value); // Noncompliant
  res.cookie("connect.sid", value); // Noncompliant
};
```

Compliant Solution

```
module.exports.index = async function (req, res) {
  const value = req.query.value;

  res.setHeader("X-Data", value);
  res.cookie("data", value);
};
```

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

© 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

https://rules.sonarsource.com/javascript

1/2

<div>OS commands should not be vulnerable to command injection attacks</div> <div> Vulnerability</div>
<div>Callbacks of array methods should have return statements</div> <div> Bug</div>
<div>Loops should not be infinite</div> <div> Bug</div>
<div>Disabling Vue.js built-in escaping is security-sensitive</div> <div> Security Hotspot</div>
<div>Disabling Angular built-in sanitization</div>