sonar RULES

Products

- 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

JavaScript static code analysis

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

All rules 285 | Vulnerability 29 | Bug 62 | Security Hotspot 43 | Code Smell 151 | Quick Fix 41

Tags

Search by name...

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

OS commands should not be vulnerable to command injection attacks
Vulnerability

Callbacks of array methods should have return statements
Bug

Loops should not be infinite
Bug

Disabling Vue.js built-in escaping is security-sensitive
Security Hotspot

Disabling Angular built-in sanitization is security-sensitive
Security Hotspot

Hard-coded credentials are security-sensitive
Security Hotspot

Function returns should not be invariant

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

Analyze your code

Vulnerability   Blocker   injection cwe sans-top25 owasp

DOM cross site scripting vulnerabilities occur when user-controlled data like `document.location.hash` is directly used to execute client-side code.

User-controlled data should always be considered untrusted and validated before being used to modify the DOM.

Noncompliant Code Example

Example of basic DOM-XSS attack (http://vulnerable/page.html#<img onerror='alert(1); src='invalid-image' />):

```
const rootDiv = document.getElementById('root');
const hash = decodeURIComponent(location.hash.substr(1));
rootDiv.innerHTML = hash;
```

Compliant Solution

innerText property of an html element sets or returns the text content of the element (removing all child nodes):

```
const rootDiv = document.getElementById('root');
const hash = decodeURIComponent(location.hash.substr(1));
rootDiv.innerText = hash;
```

See

- OWASP Top 10 2021 Category A3 - Injection
- OWASP Cheat Sheet - XSS Prevention Cheat Sheet
- OWASP Top 10 2017 Category A7 - Cross-Site Scripting (XSS)
- webappsec.org - DOM Based Cross Site Scripting or XSS of the Third Kind
- MITRE, CWE-79 - Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
- SANS Top 25 - Insecure Interaction Between Components

Available In:
sonarcloud sonarqube Developer Edition

Code Smell

Assertions should be complete

Code Smell

Variables should be declared explicitly

Code Smell

Tests should include assertions

Code Smell

"future reserved words" should not be used as identifiers

Code Smell