




 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 rules285

Vulnerability29

Bug62

Security Hotspot43

Code Smell151

Quick Fix41

Tags ▾

Search by name... 🔍

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

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

DOM updates should not lead to open redirect vulnerabilities

Analyze your code

Vulnerability

Blocker

injection cwe sans-top25 owasp

DOM open redirect vulnerabilities occur when user-controlled data like the `document.location.hash` property is directly used to perform redirections.

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

**Noncompliant Code Example**

Example of DOM open redirect vulnerability  
(`http://vulnerable/page.html#https://www.attacker.com/`):

```
document.location = document.location.hash.slice(1);
```

**Compliant Solution**

The URL can be validated with an allowlist:

```
function isValidUrl(url) {
  if(url.startsWith("https://www.example.com/")) {
    return true;
  }

  return false;
}

if(isValidUrl(document.location.hash.slice(1))) {
  document.location = document.location.hash.slice(1);
}
```

**See**

- OWASP Top 10 2021 Category A1 - Broken Access Control
- OWASP Top 10 2017 Category A5 - Broken Access Control
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-601 - URL Redirection to Untrusted Site ('Open Redirect')
- SANS Top 25 - Risky Resource Management

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/RSPEC-6105

1/2

<div>Disabling Vue.js built-in escaping is security-sensitive</div> <div> Security Hotspot</div>
<div>Disabling Angular built-in sanitization is security-sensitive</div> <div> Security Hotspot</div>
<div>Hard-coded credentials are security-sensitive</div> <div> Security Hotspot</div>
<div>Function returns should not be invariant</div> <div> Code Smell</div>