sonar

RULES

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...

to from within the loop body

Code Smell

"for" loop increment clauses should modify the loops' counters

Code Smell

Functions should not be empty

Code Smell

Server-side requests should not be vulnerable to forging attacks

Vulnerability

Non-empty statements should change control flow or have at least one side-effect

Bug

Regular expressions with the global flag should be used with caution

Bug

Replacement strings should reference existing regular expression groups

Bug

Regular expressions should not contain control characters

Bug

Alternation in regular expressions should not contain empty alternatives

Bug

Mocha timeout should be disabled by setting it to "0".

Bug

Unicode Grapheme Clusters should be avoided inside regex character classes

Bug

Assertions should not be given twice the same argument

Origins should be verified during cross-origin communications

Analyze your code

VulnerabilityCriticalcwehtml5owasp

Browsers **allow message exchanges** between Window objects of different origins.

Because any window can send or receive messages from another window, it is important to verify the sender's/receiver's identity:

- When sending a message with the postMessage method, the identity's receiver should be defined (the wildcard keyword (*) should not be used).
- When receiving a message with a message event, the sender's identity should be verified using the origin and possibly source properties.

Noncompliant Code Example

When sending a message:

```
var iframe = document.getElementById("testiframe");
iframe.contentWindow.postMessage("secret", "*"); // Noncompl
```

When receiving a message:

```
window.addEventListener("message", function(event) { // Nonc
    console.log(event.data);
});
```

Compliant Solution

When sending a message:

```
var iframe = document.getElementById("testsecureiframe");
iframe.contentWindow.postMessage("hello", "https://secure.ex
```

When receiving a message:

```
window.addEventListener("message", function(event) {

    if (event.origin !== "http://example.org") // Compliant
        return;








    console.log(event.data)
});
```

See

- [OWASP Top 10 2021 Category A1](#) - Broken Access Control
- [OWASP Top 10 2017 Category A3](#) - Broken Authentication and Session Management
- [developer.mozilla.org](#) - postMessage API
- [MITRE, CWE-345](#) - Insufficient Verification of Data Authenticity

https://rules.sonarsource.com/javascript/RSPEC-2819

1/2

 Bug	<div>Available In:   </div>
Alternatives in regular expressions should be grouped when used with anchors	
 Bug	
Promise rejections should not be caught by 'try' block	
 Bug	
Collection elements should not be replaced unconditionally	<div>© 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</div>
 Bug	
Errors should not be created without being thrown	