




 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... 🔍

Code Smell

"switch" statements should not contain non-case labels

Code Smell

Vulnerability

A new session should be created during user authentication

Vulnerability

Vulnerability

JWT should be signed and verified with strong cipher algorithms

Vulnerability

Vulnerability

Cipher algorithms should be robust

Vulnerability

Vulnerability

Encryption algorithms should be used with secure mode and padding scheme

Vulnerability

Vulnerability

Server hostnames should be verified during SSL/TLS connections

Vulnerability

Vulnerability

Server certificates should be verified during SSL/TLS connections

Vulnerability

Vulnerability

Cryptographic keys should be robust

Vulnerability

Vulnerability

Weak SSL/TLS protocols should not be used

Vulnerability

Vulnerability

Origins should be verified during cross-origin communications

Vulnerability

Vulnerability

Regular expressions should not be vulnerable to Denial of Service attacks

Vulnerability

Vulnerability

File uploads should be restricted

Vulnerability

Loops should not be infinite

Analyze your code

Bug

Blocker

An infinite loop is one that will never end while the program is running, i.e., you have to kill the program to get out of the loop. Whether it is by meeting the loop's end condition or via a break, every loop should have an end condition.

**Known Limitations**

- False positives: when yield is used - [Issue #674](#).
- False positives: when an exception is raised by a function invoked within the loop.
- False negatives: when a loop condition is based on an element of an array or object.

**Noncompliant Code Example**

```
for (;;) { // Noncompliant; end condition omitted
  // ...
}

var j = 0;
while (true) { // Noncompliant; constant end condition
  j++;
}

var k;
var b = true;
while (b) { // Noncompliant; constant end condition
  k++;
}
```

**Compliant Solution**








```
while (true) { // break will potentially allow leaving the loop
  if (someCondition) {
    break;
  }
}

var k;
var b = true;
while (b) {
  k++;
  b = k < 10;
}

outer:
while(true) {
  while(true) {
    break outer;
  }
}
```

https://rules.sonarsource.com/javascript/RSPEC-2189

1/2

 Vulnerability	<div>Available In:</div> <div>      </div>
Function calls should not pass extra arguments	
 Bug	
Regular expressions should be syntactically valid	
 Bug	
Getters and setters should access the expected fields	<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. <a href="#">Privacy Policy</a></div>
 Bug	
"super()" should be invoked appropriately	