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...

Security Hotspot

Allowing confidential information to be logged is security-sensitive

Security Hotspot

Allowing browsers to perform DNS prefetching is security-sensitive

Security Hotspot

Disabling Certificate Transparency monitoring is security-sensitive

Security Hotspot

Disabling Strict-Transport-Security policy is security-sensitive

Security Hotspot

Disabling strict HTTP no-referrer policy is security-sensitive

Security Hotspot

Allowing browsers to sniff MIME types is security-sensitive

Security Hotspot

Disabling content security policy frame-ancestors directive is security-sensitive

Security Hotspot

Allowing mixed-content is security-sensitive

Security Hotspot

Disabling content security policy fetch directives is security-sensitive

Security Hotspot

Disabling resource integrity features is security-sensitive

Security Hotspot

Disclosing fingerprints from web application technologies is security-sensitive

Special identifiers should not be bound or assigned

Analyze your code

Bug Major pitfall

JavaScript has special identifiers that, while not reserved, still should not be used as identifiers. They include:

- `eval` - evaluates a string as JavaScript code
- `arguments` - used to access function arguments through indexed properties.
- `undefined` - returned for values and properties that have not yet been assigned
- `NaN` - Not a Number; returned when math functions fail.
- `Infinity` - when a number exceeds the upper limit of the floating point numbers

These words should not be bound or assigned, because doing so would overwrite the original definitions of these identifiers. What's more, assigning or binding some of these names will generate an error in JavaScript strict mode code.

Noncompliant Code Example

```
eval = 17; // Noncompliant
arguments++; // Noncompliant
++eval; // Noncompliant
var obj = { set p(arguments) { } }; // Noncompliant
var eval; // Noncompliant
try { } catch (arguments) { } // Noncompliant
function x(eval) { } // Noncompliant
function arguments() { } // Noncompliant
var y = function eval() { }; // Noncompliant
var f = new Function("arguments", "return 17;"); // Noncompl

function fun() {
  if (arguments.length == 0) { // Compliant
    // do something
  }
}
```

Compliant Solution

```
result = 17;
args++;
++result;
var obj = { set p(arg) { } };
var result;
try { } catch (args) { }
function x(arg) { }
function args() { }
var y = function fun() { };
var f = new Function("args", "return 17;");

function fun() {
  if (arguments.length == 0) {
    // do something
```

Security Hotspot

Having a permissive Cross-Origin Resource Sharing policy is security-sensitive

Security Hotspot

Delivering code in production with debug features activated is security-sensitive

Security Hotspot

Creating cookies without the "HttpOnly" flag is security-sensitive

Security Hotspot

Creating cookies without the "secure"

```
    }
  }
```

Available In:

sonarlint | sonarcloud | sonarqube

Privacy Policy