




 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... 🔍


Attempts should not be made to update "const" variables

 Bug


Strict equality operators should not be used with dissimilar types

 Bug


"new" operators should be used with functions

 Bug


Non-existent operators '+=', '=-' and '!=' should not be used

 Bug


"NaN" should not be used in comparisons

 Bug


Setters should not return values

 Bug


Properties of variables with "null" or "undefined" values should not be accessed

 Bug


A "for" loop update clause should move the counter in the right direction

 Bug


Return values from functions without side effects should not be ignored

 Bug


Special identifiers should not be bound or assigned

 Bug

Values should not be uselessly incremented

 Bug

Related "if/else if" statements should not have the same condition

 Bug

Using publicly writable directories is security-sensitive

Analyze your code

Security HotspotCritical ⓘcwe owasp

Operating systems have global directories where any user has write access. Those folders are mostly used as temporary storage areas like /tmp in Linux based systems. An application manipulating files from these folders is exposed to race conditions on filenames: a malicious user can try to create a file with a predictable name before the application does. A successful attack can result in other files being accessed, modified, corrupted or deleted. This risk is even higher if the application runs with elevated permissions.

In the past, it has led to the following vulnerabilities:

- [CVE-2012-2451](#)
- [CVE-2015-1838](#)

This rule raises an issue whenever it detects a hard-coded path to a publicly writable directory like /tmp (see examples bellow). It also detects access to environment variables that point to publicly writable directories, e.g., TMP and TMPDIR.

- /tmp
- /var/tmp
- /usr/tmp
- /dev/shm
- /dev/mqueue
- /run/lock
- /var/run/lock
- /Library/Caches
- /Users/Shared
- /private/tmp
- /private/var/tmp
- \Windows\Temp
- \Temp
- \TMP

Ask Yourself Whether

- Files are read from or written into a publicly writable folder
- The application creates files with predictable names into a publicly writable folder

There is a risk if you answered yes to any of those questions.





Recommended Secure Coding Practices

- Use a dedicated sub-folder with tightly controlled permissions
- Use secure-by-design APIs to create temporary files. Such API will make sure:
 - The generated filename is unpredictable
 - The file is readable and writable only by the creating user ID
 - The file descriptor is not inherited by child processes
 - The file will be destroyed as soon as it is closed

Sensitive Code Example

https://rules.sonarsource.com/javascript/RSPEC-5443

1/2

 Bug
Objects should not be created to be dropped immediately without being used
 Bug
Identical expressions should not be used on both sides of a binary operator
 Bug
All code should be reachable
 Bug
Loops with at most one iteration should be refactored

```
const fs = require('fs');

let tmp_file = "/tmp/temporary_file"; // Sensitive
fs.readFile(tmp_file, 'utf8', function (err, data) {
  // ...
});
```

```
const fs = require('fs');

let tmp_dir = process.env.TMPDIR; // Sensitive
fs.readFile(tmp_dir + "/temporary_file", 'utf8', function (e
  // ...
});
```

Compliant Solution

```
const tmp = require('tmp');

const tmpobj = tmp.fileSync(); // Compliant
```

See

- [OWASP Top 10 2021 Category A1](#) - Broken Access Control
- [OWASP Top 10 2017 Category A5](#) - Broken Access Control
- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure
- [MITRE, CWE-377](#) - Insecure Temporary File
- [MITRE, CWE-379](#) - Creation of Temporary File in Directory with Incorrect Permissions
- [OWASP, Insecure Temporary File](#)

Available In:

