




 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... 🔍

Object literal shorthand syntax should be used

Code Smell

Strings and non-strings should not be added

Code Smell

Object literal syntax should be used

Code Smell

"undefined" should not be assigned

Code Smell

Trailing commas should not be used

Code Smell

Array constructors should not be used

Code Smell

Quotes for string literals should be used consistently

Code Smell

Statements should end with semicolons

Code Smell

Comments should not be located at the end of lines of code

Code Smell

Loops should not contain more than a single "break" or "continue" statement

Code Smell

Variable, property and parameter names should comply with a naming convention

Code Smell

Lines should not end with trailing whitespaces

Code Smell

Using command line arguments is security-sensitive

Analyze your code

Security Hotspot

Critical

Using command line arguments is security-sensitive. It has led in the past to the following vulnerabilities:

- [CVE-2018-7281](#)
- [CVE-2018-12326](#)
- [CVE-2011-3198](#)

Command line arguments can be dangerous just like any other user input. They should never be used without being first validated and sanitized.

Remember also that any user can retrieve the list of processes running on a system, which makes the arguments provided to them visible. Thus passing sensitive information via command line arguments should be considered as insecure.

This rule raises an issue when on every program entry points (main methods) when command line arguments are used. The goal is to guide security code reviews.

Ask Yourself Whether

- any of the command line arguments are used without being sanitized first.
- your application accepts sensitive information via command line arguments.

If you answered yes to any of these questions you are at risk.

Recommended Secure Coding Practices

[Sanitize](#) all command line arguments before using them.

Any user or application can list running processes and see the command line arguments they were started with. There are safer ways of providing sensitive information to an application than exposing them in the command line. It is common to write them on the process' standard input, or give the path to a file containing the information.

Sensitive Code Example

```
// The process object is a global that provides information
var param = process.argv[2]; // Sensitive: check how the arg
console.log('Param: ' + param);
```

See

- [OWASP Top 10 2017 Category A1](#) - Injection
- [MITRE, CWE-88](#) - Argument Injection or Modification
- [MITRE, CWE-214](#) - Information Exposure Through Process Environment
- [SANS Top 25](#) - Insecure Interaction Between Components







Deprecated

This rule is deprecated, and will eventually be removed.

Available In:

https://rules.sonarsource.com/javascript/RSPEC-4823

1/2

<div>Files should contain an empty newlne at the end</div> <div> Code Smell</div>	<div>sonarcloud  sonarqube </div>
<div>An open curly brace should be located at the end of a line</div> <div> Code Smell</div>	
<div>Tabulation characters should not be used</div> <div> Code Smell</div>	
<div>Function and method names should comply with a naming convention</div> <div> Code Smell</div>	<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>