




 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... 🔍

 Bug


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




Objects should not be created to be dropped immediately without being used




Identical expressions should not be used on both sides of a binary operator




All code should be reachable




Loops with at most one iteration should be refactored




Variables should not be self-assigned




Function argument names should be unique




Property names should not be duplicated within a class or object literal



Bitwise operators should not be used in boolean contexts






Constructing arguments of system commands from user input is security-sensitive



Allowing requests with excessive content length is security-sensitive

Dynamically executing code is security-sensitive

 Security Hotspot Critical ⓘ cwe owasp

Executing code dynamically is security-sensitive. It has led in the past to the following vulnerabilities:

- [CVE-2017-9807](#)
- [CVE-2017-9802](#)

Some APIs enable the execution of dynamic code by providing it as strings at runtime. These APIs might be useful in some very specific meta-programming use-cases. However most of the time their use is frowned upon as they also increase the risk of **Injected Code**. Such attacks can either run on the server or in the client (example: XSS attack) and have a huge impact on an application's security.

This rule raises issues on calls to `eval` and `Function` constructor. This rule does not detect code injections. It only highlights the use of APIs which should be used sparingly and very carefully. The goal is to guide security code reviews.

Ask Yourself Whether

- the executed code may come from an untrusted source and hasn't been sanitized.
- you really need to run code dynamically.

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

Recommended Secure Coding Practices

Regarding the execution of unknown code, the best solution is to not run code provided by an untrusted source. If you really need to do it, run the code in a **sandboxed** environment. Use jails, firewalls and whatever means your operating system and programming language provide (example: **Security Managers** in java, **iframes** and **same-origin policy** for javascript in a web browser).

Do not try to create a blacklist of dangerous code. It is impossible to cover all attacks that way.

Avoid using dynamic code APIs whenever possible. Hard-coded code is always safer.

Sensitive Code Example

```
let value = eval('obj.' + propName); // Sensitive
let func = Function('obj' + propName); // Sensitive
```

Exceptions







This rule will not raise an issue when the argument of the `eval` or `Function` is a literal string as it is reasonably safe.

See

- [OWASP Top 10 2021 Category A3](#) - Injection
- [OWASP Top 10 2017 Category A1](#) - Injection

https://rules.sonarsource.com/javascript/RSPEC-1523

1/2

<div> Security Hotspot</div>	<div><ul style="list-style-type: none">• MITRE, CWE-95 - Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')<div>Available In:</div><div> </div></div>
<div><div>Statically serving hidden files is security-sensitive</div><div> Security Hotspot</div></div>	
<div><div>Using intrusive permissions is security-sensitive</div><div> Security Hotspot</div></div>	
<div><div>Disabling auto-escaping in template engines is security-sensitive</div><div> Security Hotspot</div></div>	
<div><div>Using shell interpreter when executing OS commands is security-sensitive</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](#)