

JS

JavaScript

Secrets

ABAP

Apex

C

C++

CloudFormation

COBOL

C#

CSS

Flex

Go

HTML

Java

JS

Kotlin

Objective C

PHP

PL/I

PL/SQL

Python

RPG

Ruby

Scala

Swift

Terraform

Text

TypeScript

T-SQL

VB.NET

VB6

XML

JS

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

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

Analyze your code

Security Hotspot

Minor

cwe owasp express.js

Clickjacking attacks occur when an attacker try to trick a user to click on certain buttons/links of a legit website. This attack can take place with malicious HTML frames well hidden in an attacker website.

For instance, suppose a safe and authentic page of a social network (<https://socialnetworkexample.com/makemyprofilpublic>) which allows an user to change the visibility of his profile by clicking on a button. This is a critical feature with high privacy concerns. Users are generally well informed on the social network of the consequences of this action. An attacker can trick users, without their consent, to do this action with the below embedded code added on a malicious website:

```
<html>
<b>Click on the button below to win 5000$</b>
<br>
<iframe src="https://socialnetworkexample.com/makemyprofilpu
</html>
```

Playing with the size of the iframe it's sometimes possible to display only the critical parts of a page, in this case the button of the *makemyprofilpublic* page.

Ask Yourself Whether

- Critical actions** of the application are prone to clickjacking attacks because a simple click on a link or a button can trigger them.

There is a risk if you answered yes to this question.

Recommended Secure Coding Practices

Implement content security policy *frame-ancestors* directive which is supported by all modern browsers and will specify the origins of frame allowed to be loaded by the browser (this directive deprecates X-Frame-Options).

Sensitive Code Example

In Express.js application the code is sensitive if the helmet-csp or helmet middleware is used without the *frameAncestors* directive (or if *frameAncestors* is set to 'none'):





```
const express = require('express');
const helmet = require('helmet');

let app = express();

app.use(
  helmet.contentSecurityPolicy({
    directives: {
      // other directives
      frameAncestors: ["'none'"] // Sensitive: frameAncestor
```

https://rules.sonarsource.com/javascript/RSPEC-5732

1/2

Files should contain an empty newline at the end
 Code Smell
An open curly brace should be located at the end of a line
 Code Smell
Tabulation characters should not be used
 Code Smell
Function and method names should comply with a naming convention
 Code Smell

```
    }
  })
};
```

Compliant Solution

In Express.js application a standard way to implement CSP frame-ancestors directive is the [helmet-csp](#) or [helmet](#) middleware:

```
const express = require('express');
const helmet = require('helmet');

let app = express();

app.use(
  helmet.contentSecurityPolicy({
    directives: {
      // other directives
      frameAncestors: ["'example.com'"] // Compliant
    }
  })
);
```

See

- [OWASP Top 10 2021 Category A4](#) - Insecure Design
- [OWASP Top 10 2021 Category A5](#) - Security Misconfiguration
- [OWASP Top 10 2017 Category A6](#) - Security Misconfiguration
- [OWASP Cheat Sheets](#) - Clickjacking Defense Cheat Sheet
- [developer.mozilla.org](#) - Frame-ancestors
- [developer.mozilla.org](#) - Content Security Policy (CSP)
- [MITRE, CWE-451](#) - User Interface (UI) Misrepresentation of Critical Information
- [w3.org](#) - Content Security Policy Level 3

Available In:

