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

JS

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...

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 Strict-Transport-Security policy is security-sensitive

Analyze your code

Security Hotspot   Minor   cwe express.js owasp

When implementing the HTTPS protocol, the website mostly continue to support the HTTP protocol to redirect users to HTTPS when they request a HTTP version of the website. These redirects are not encrypted and are therefore vulnerable to man in the middle attacks. The Strict-Transport-Security policy header (HSTS) set by an application instructs the web browser to convert any HTTP request to HTTPS.

Web browsers that see the Strict-Transport-Security policy header for the first time record information specified in the header:

- the `max-age` directive which specify how long the policy should be kept on the web browser.
- the `includeSubDomains` optional directive which specify if the policy should apply on all sub-domains or not.
- the `preload` optional directive which is not part of the HSTS specification but supported on all modern web browsers.

With the `preload` directive the web browser never connects in HTTP to the website and to use this directive, it is required to submit the concerned application to a preload service maintained by Google.

Ask Yourself Whether

- The website is accessible with the unencrypted HTTP protocol.

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

Recommended Secure Coding Practices

Implement Strict-Transport-Security policy header, it is recommended to apply this policy to all subdomains (`includeSubDomains`) and for at least 6 months (`max-age=15552000`) or even better for 1 year (`max-age=31536000`).

Sensitive Code Example

In Express.js application the code is sensitive if the helmet or hsts middleware are disabled or used without recommended values:

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

let app = express();

app.use(helmet.hsts({
  maxAge: 3153600, // Sensitive, recommended >= 15552000
  includeSubDomains: false // Sensitive, recommended 'true'
}));
```

Compliant Solution

In Express.js application a standard way to implement HSTS is with the helmet or hsts middleware:

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

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

let app = express();

app.use(helmet.hsts({
  maxAge: 31536000,
  includeSubDomains: true
})); // Compliant
```

See

- OWASP Top 10 2021 Category A5 - Security Misconfiguration
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- developer.mozilla.org - Strict Transport Security

Available In:

sonarcloud | sonarqube

Privacy Policy