




 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





TypeScript static code analysis


Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your TYPESCRIPT code


All rules279

 Vulnerability27


 Bug51


 Security Hotspot43

 Code Smell158


 Quick Fix50


Tags


Search by name...


 Code Smell


Primitive types should be omitted from initialized or defaulted declarations





 Non-null assertions should not be used





 "undefined" should not be assigned





 Trailing commas should not be used





 Array constructors should not be used





 Quotes for string literals should be used consistently





 Statements should end with semicolons





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





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






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



 Lines should not end with trailing whitespaces



Disabling content security policy fetch directives is security-sensitive

 Security Hotspot Minor owasp express.js

Content security policy (CSP) (fetch directives) is a [W3C standard](#) which is used by a server to specify, via a http header, the origins from where the browser is allowed to load resources. It can help to mitigate the risk of cross site scripting (XSS) attacks and reduce privileges used by an application. If the website doesn't define CSP header the browser will apply [same-origin policy](#) by default.

```
Content-Security-Policy: default-src 'self'; script-src 'sel
```

In the above example, all resources are allowed from the website where this header is set and script resources fetched from example.com are also authorized:

```
</script> <!-- will be loaded
</script> <!-- w
<script src="http://www.example.com/library.js"></script> <!--
<script src="selfhostedscript.js"></script> <!-- will be load
<script src="http://www.otherexample.com/library.js"></script
```

Ask Yourself Whether

- The resources of the application are fetched from various untrusted locations.

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

Recommended Secure Coding Practices

Implement content security policy fetch directives, in particular *default-src* directive and continue to properly sanitize and validate all inputs of the application, indeed CSP fetch directives is only a tool to reduce the impact of cross site scripting attacks.

Sensitive Code Example

In a Express.js application, the code is sensitive if the [helmet](#) contentSecurityPolicy middleware is disabled:





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

let app = express();
app.use(
  helmet({
    contentSecurityPolicy: false, // sensitive
  })
);
```

Compliant Solution

https://rules.sonarsource.com/typescript/RSPEC-5728

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

In a Express.js application, a standard way to implement CSP is the [helmet contentSecurityPolicy middleware](#):

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

let app = express();
app.use(helmet.contentSecurityPolicy()); // Compliant
```

See

- [OWASP Top 10 2021 Category A5](#) - Security Misconfiguration
- [w3.org](#) - Content Security Policy Level 3
- [OWASP Top 10 2017 Category A6](#) - Security Misconfiguration
- [developer.mozilla.org](#) - Content Security Policy (CSP)

Available In:  