




 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... 🔍

Redundant casts and non-null assertions should be avoided

Code Smell

Type aliases should be used

Code Smell

Type guards should be used

Code Smell

"module" should not be used

Code Smell

"for of" should be used with Iterables

Code Smell

Imports from the same modules should be merged

Code Smell

Jump statements should not be redundant

Code Smell

Default export names and file names should match

Code Smell

The global "this" object should not be used

Code Smell

"catch" clauses should do more than rethrow

Code Smell

Boolean checks should not be inverted

Code Smell

Deprecated APIs should not be used

Code Smell

Wrapper objects should not be used

Code Smell

Statically serving hidden files is security-sensitive

Analyze your code

Security Hotspot

Major ?

cwe owasp express.js

Hidden files are created automatically by many tools to save user-preferences, well-known examples are .profile, .bashrc, .bash_history or .git. To simplify the view these files are not displayed by default using operating system commands like ls.

Outside of the user environment, hidden files are sensitive because they are used to store privacy-related information or even hard-coded secrets.

Ask Yourself Whether

- Hidden files may have been inadvertently uploaded to the static server's public directory and it accepts requests to hidden files.
- There is no business use cases linked to serve files in .name format but the server is not configured to reject requests to this type of files.

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

Recommended Secure Coding Practices

- Disable the serving of hidden files.

Sensitive Code Example

Express.js [serve-static](#) middleware:

```
let serveStatic = require("serve-static");
let app = express();
let serveStaticMiddleware = serveStatic('public', { 'index': fa
app.use(serveStaticMiddleware);
```

Compliant Solution

Express.js [serve-static](#) middleware:

```
let serveStatic = require("serve-static");
let app = express();
let serveStaticMiddleware = serveStatic('public', { 'index':
let serveStaticDefault = serveStatic('public', { 'index': fa
app.use(serveStaticMiddleware);
```





See

- [OWASP Top 10 2021 Category A5](#) - Security Misconfiguration
- [github.com/mtojek/go-url-fuzzer](#) - Discover hidden files and directories on a web server.
- [OWASP Web Top 10 2017 Category A6](#) - Security Misconfiguration.
- [MITRE, CWE-538](#) - File and Directory Information Exposure

Available In:

https://rules.sonarsource.com/typescript/RSPEC-5691

1/2

for primitive types
 Code Smell
Multiline string literals should not be used
 Code Smell
Local variables should not be declared and then immediately returned or thrown
 Code Smell
Function call arguments should not start on new lines
 Code Smell