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 rules 279 | Vulnerability 27 | Bug 51 | Security Hotspot 43 | Code Smell 158 | Quick Fix 50

Tags | Search by name...

is security-sensitive
Security Hotspot

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

Allowing mixed-content is security-sensitive
Security Hotspot

Disabling content security policy fetch directives is security-sensitive
Security Hotspot

Disabling resource integrity features is security-sensitive
Security Hotspot

Disclosing fingerprints from web application technologies is security-sensitive
Security Hotspot

Having a permissive Cross-Origin Resource Sharing policy is security-sensitive
Security Hotspot

Delivering code in production with debug features activated is security-sensitive
Security Hotspot

Creating cookies without the "HttpOnly" flag is security-sensitive
Security Hotspot

Creating cookies without the "secure" flag is security-sensitive
Security Hotspot

Using hardcoded IP addresses is security-sensitive
Security Hotspot

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

Analyze your code

Bug Major

There is no good reason to create a new object to not do anything with it. Most of the time, this is due to a missing piece of code and so could lead to an unexpected behavior in production.

If it was done on purpose because the constructor has side-effects, then that side-effect code should be moved into a separate method and called directly.

Noncompliant Code Example

```
new MyConstructor(); // Non-Compliant
```

Compliant Solution

```
var something = new MyConstructor();  // Compliant
```

Exceptions

Immediately dropped new objects inside `try`-statements are ignored.

```
try {
  new MyConstructor();
} catch (e) {
  /* ... */
}
```

Available In:

sonarlint sonarcloud sonarqube

Regular expression quantifiers and character classes should be used concisely

Code Smell

Regular expression literals should be used when possible

Code Smell

"await" should not be used redundantly

Code Smell

Redundant casts and non-null assertions should be avoided

Code Smell