sonar RULES

Products

- 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...

should be removed
Code Smell

Nested blocks of code should not be left empty
Code Smell

Functions should not have too many parameters
Code Smell

OS commands should not be vulnerable to argument injection attacks
Vulnerability

Repeated patterns in regular expressions should not match the empty string
Bug

Empty collections should not be accessed or iterated
Bug

"delete" should be used only with object properties
Bug

Function parameters, caught exceptions and foreach variables' initial values should not be ignored
Bug

Forwarding client IP address is security-sensitive
Security Hotspot

Allowing confidential information to be logged is security-sensitive
Security Hotspot

Allowing browsers to perform DNS prefetching is security-sensitive
Security Hotspot

Disabling Certificate Transparency monitoring is security-sensitive

## "new" operators should be used with functions

Analyze your code

Bug Major

The `new` keyword should only be used with objects that define a constructor function. Use it with anything else, and you'll get a `TypeError` because there won't be a constructor function for the `new` keyword to invoke.

### Noncompliant Code Example

```
function MyClass() {
  this.foo = 'bar';
}

var someClass = 1;

var obj1 = new someClass;    // Noncompliant;
var obj2 = new MyClass();    // Noncompliant if considerJSDo
```

### Compliant Solution

```
/**
 * @constructor
 */
function MyClass() {
  this.foo = 'bar';
}

var someClass = function(){
  this.prop = 1;
}

var obj1 = new someClass;  // Compliant
var obj2 = new MyClass();  // Compliant regardless of consid
```

Available In:

sonarlint sonarcloud sonarqube

monitoring is security-sensitive

Security Hotspot

Disabling Strict-Transport-Security policy is security-sensitive

Security Hotspot

Disabling strict HTTP no-referrer policy is security-sensitive

Security Hotspot

Allowing browsers to sniff MIME types is security-sensitive

Security Hotspot

Disabling content security policy