




 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... 🔍

cross-origin communications

Vulnerability

Regular expressions should not be vulnerable to Denial of Service attacks

Vulnerability

File uploads should be restricted

Vulnerability

Regular expressions should be syntactically valid

Bug

Types without members, 'any' and 'never' should not be used in type intersections

Bug

Getters and setters should access the expected fields

Bug

"super()" should be invoked appropriately

Bug

Results of "in" and "instanceof" should be negated rather than operands

Bug

A compare function should be provided when using "Array.prototype.sort()"

Bug

Jump statements should not occur in "finally" blocks

Bug

Using slow regular expressions is security-sensitive

Security Hotspot

Using publicly writable directories is security-sensitive

Function returns should not be invariant

Analyze your code

Code SmellBlocker?

When a function is designed to return an invariant value, it may be poor design, but it shouldn't adversely affect the outcome of your program. However, when it happens on all paths through the logic, it is likely a mistake.

This rule raises an issue when a function contains several `return` statements that all return the same value.

Noncompliant Code Example





```
function foo(a) { // Noncompliant
  let b = 12;
  if (a) {
    return b;
  }
  return b;
}
```

Available In:
sonarlint | sonarcloud | sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.
[Privacy Policy](#)

https://rules.sonarsource.com/typescript/RSPEC-3516

1/2

 Security Hotspot
Using clear-text protocols is security-sensitive  Security Hotspot
Expanding archive files without controlling resource consumption is security-sensitive  Security Hotspot
Using weak hashing algorithms is security-sensitive  Security Hotspot
Disabling CSRF protections is security-sensitive