




 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... 🔍


 Code Smell


Variables should not be shadowed





 Redundant pairs of parentheses should be removed





 Nested blocks of code should not be left empty





 Functions should not have too many parameters





 OS commands should not be vulnerable to argument injection attacks




 Repeated patterns in regular expressions should not match the empty string



 Empty collections should not be accessed or iterated



 "delete" should be used only with object properties



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



 Forwarding client IP address is security-sensitive





 Allowing confidential information to be logged is security-sensitive




### Generators should "yield" something

Analyze your code

 Bug

 Major



api-design es2015

A generator without a `yield` statement is at best confusing, and at worst a bug in your code, since the iterator produced by your code will always be empty.




#### Noncompliant Code Example

```
function* myGen(a, b) { // Noncompliant
  let answer = 0;
  answer += a * b;
}
```

#### Compliant Solution

```
function* myGen(a, b) {
  let answer = 0;
  while (answer < 42) {
    answer += a * b;
    yield answer;
  }
}
```

Available In:






 |  | 

© 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-3531

1/2

 Security Hotspot
<b>Allowing browsers to perform DNS prefetching is security-sensitive</b>  Security Hotspot
<b>Disabling Certificate Transparency monitoring is security-sensitive</b>  Security Hotspot
<b>Disabling Strict-Transport-Security policy is security-sensitive</b>  Security Hotspot
<b>Disabling strict HTTP no-referrer policy is security-sensitive</b>  Security Hotspot