




 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... 🔍

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 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

Related "if/else if" statements should not have the same condition

Analyze your code

Bug Major ? unused pitfall

A chain of `if/else if` statements is evaluated from top to bottom. At most, only one branch will be executed: the first one with a condition that evaluates to `true`.

Therefore, duplicating a condition automatically leads to dead code. Usually, this is due to a copy/paste error. At best, it's simply dead code and at worst, it's a bug that is likely to induce further bugs as the code is maintained, and obviously it could lead to unexpected behavior.

Note that this rule requires `Node.js` to be available during analysis.




Noncompliant Code Example

```
if (param == 1)
  openWindow();
else if (param == 2)
  closeWindow();
else if (param == 1) // Noncompliant
  moveWindowToTheBackground();
```

Compliant Solution

```
if (param == 1)
  openWindow();
else if (param == 2)
  closeWindow();
else if (param == 3)
  moveWindowToTheBackground();
```





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-1862

1/2

<div>Creating cookies without the "secure" flag is security-sensitive</div> <div> Security Hotspot</div>
<div>Using hardcoded IP addresses is security-sensitive</div> <div> Security Hotspot</div>
<div>Regular expression quantifiers and character classes should be used concisely</div> <div> Code Smell</div>
<div>Regular expression literals should be used when possible</div> <div> Code Smell</div>