

C#

3 CSS

 $\mathbb{X}$ Flex

-GO Go

5 HTML Java

JavaScript

Kotlin

Kubernetes

Objective C

PHP

PL/I

PL/SQL

Python

**RPG** 

Ruby

Scala

Swift

Terraform

Text

**TypeScript** 

T-SQL

**VB.NET** 

VB6

XML



## C static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your C code

ΑII 311 6 Vulnerability (13) rules

₩ Bug (74)

Security 18 Hotspot

⊗ Code 206 Smell

O Quick 14

Tags Search by name.

"memset" should not be used to delete sensitive data

6 Vulnerability

POSIX functions should not be called with arguments that trigger buffer overflows

♠ Vulnerability

XML parsers should not be vulnerable to XXE attacks

Vulnerability

Function-like macros should not be invoked without all of their arguments

₩ Bug

The address of an automatic object should not be assigned to another object that may persist after the first object has ceased to exist

👬 Bug

"pthread\_mutex\_t" should be unlocked in the reverse order they were locked

"pthread\_mutex\_t" should be properly initialized and destroyed

# Bua

"pthread\_mutex\_t" should not be consecutively locked or unlocked

# Bug

Functions with "noreturn" attribute should not return

₩ Bua

"memcmp" should only be called with pointers to trivially copyable types with no padding

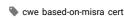
🖷 Bug

"switch" statements should have "default" clauses

Analyze your code







The requirement for a final default clause is defensive programming. The clause should either take appropriate action, or contain a suitable comment as to why no action is taken. When the switch covers all current values of an enum - and especially when it doesn't - a default case should still be used because there is no guarantee that the enum won't be extended.

Note that there is a more nuanced version of this rule: {rule:cpp:S3562}. Use this rule if you want to require a default case for every switch even if it already handles all enumerators of an enum. Otherwise, use {rule:cpp:S3562}.

## Noncompliant Code Example

```
switch (param) { // Noncompliant - default clause is missing
 case 0:
   doSomething();
   break:
  case 1:
   doSomethingElse():
    break;
```

## **Compliant Solution**

```
switch (param) {
 case 0:
    doSomething();
  case 1:
    doSomethingElse();
    break;
  default:
    doDefault();
    break;
}
```

## See

- MISRA C:2004, 15.0 The MISRA C switch syntax shall be used.
- MISRA C:2004, 15.3 The final clause of a switch statement shall be the default clause
- MISRA C++:2008, 6-4-3 A switch statement shall be a well-formed switch statement.
- MISRA C++:2008, 6-4-6 The final clause of a switch statement shall be the default-clause
- MISRA C:2012, 16.1 All switch statements shall be well-formed
- MISRA C:2012, 16.4 Every switch statement shall have a default label
- MISRA C:2012, 16.5 A default label shall appear as either the first or the last switch label of a switch statement
- MITRE, CWE-478 Missing Default Case in Switch Statement

Stack allocated memory and nonowned memory should not be freed

R
Bug

Closed resources should not be
accessed

Bug

Dynamically allocated memory should
be released

Bug

Freed memory should not be used

• CERT, MSC01-C. - Strive for logical completeness

See Also
• {rule:cpp:S3562}

Available In:
sonarlint ⊖ sonarcloud ₺ sonarqube Developer Edition

© 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