MSC17-C. Finish every set of statements associated with a case label with a break statement

A `switch` statement consists of several case labels, plus a default label. The default label is optional but recommended. (See MSC01-C. Strive for logical completeness.) A series of statements following a case label conventionally ends with a `break` statement; if omitted, control flow falls through to the next case in the `switch` statement block. Because the `break` statement is not required, omitting it does not produce compiler diagnostics. If the omission was unintentional, it can result in an unexpected control flow.

Noncompliant Code Example

In this noncompliant code example, the case where `widget_type` is `WE_W` lacks a `break` statement. Consequently, statements that should be executed only when `widget_type` is `WE_X` are executed even when `widget_type` is `WE_W`.

```
enum WidgetEnum { WE_W, WE_X, WE_Y, WE_Z } widget_type;
widget_type = WE_X;

switch (widget_type) {
  case WE_W:
    /* ... */
  case WE_X:
    /* ... */
    break;
  case WE_Y:
  case WE_Z:
    /* ... */
    break;
  default: /* Can't happen */
         /* Handle error condition */
}
```

Compliant Solution

In this compliant solution, each sequence of statements following a case label ends with a `break` statement:

```
enum WidgetEnum { WE_W, WE_X, WE_Y, WE_Z } widget_type;
widget_type = WE_X;

switch (widget_type) {
  case WE_W:
    /* ... */
    break;
  case WE_X:
    /* ... */
    break;
  case WE_Y:
  case WE_Z:
    /* ... */
    break;
  default: /* Can't happen */
         /* Handle error condition */
}
```

A `break` statement is not required following the case where `widget_type` is `WE_Y` because there are no statements before the next case label, indicating that both `WE_Y` and `WE_Z` should be handled in the same fashion.

A `break` statement is not required following the default case because it would not affect the control flow.

Exceptions

MSC17-C-EX1: The last label in a `switch` statement requires no final `break`. It will conventionally be the `default` label.

MSC17-C-EX2: When control flow is intended to cross statement labels, it is permissible to omit the `break` statement. In these instances, the unusual control flow must be explicitly documented.

```
enum WidgetEnum { WE_W, WE_X, WE_Y, WE_Z } widget_type;
widget_type = WE_X;

switch (widget_type) {
  case WE_W:
    /* ... */
    /* No break; process case for WE_X as well */
  case WE_X:
    /* ... */
    break;
  case WE_Y: case WE_Z:
    /* ... */
    break;
  default: /* Can't happen */
         /* Handle error condition */
}
```

Risk Assessment

Failure to include `break` statements leads to unexpected control flow.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
MSC17-C	Medium	Likely	Low	**P18**	**L1**

Automated Detection

Tool	Version	Checker	Description
Astrée	19.04	**switch-clause-break**	Fully checked
CodeSonar	5.1p0	**LANG.STRUCT.SW.MB**	Missing break
Compass/ROSE			
Coverity	2017.07	**MISSING_BREAK**	Can find instances of missing break statement between cases in `switch` statement
ECLAIR	1.2	**CC2.MSC17**	Fully implemented
Klocwork	2018	**MISRA.SWITCH.WELL_FORMED.BREAK.2012**	
LDRA tool suite	9.7.1	**62 S**	Fully implemented
Parasoft C/C++test	10.4.2	**CERT_C-MSC17-a**	Missing break statement between cases in a switch statement
Polyspace Bug Finder	R2019b	CERT C: Rec. MSC17-C	Checks for missing break of switch case (rec. fully covered)
PRQA QA-C	9.7	**2003**	
PVS-Studio	6.23	**V796**	
RuleChecker	19.04	**switch-clause-break**	Fully checked
SonarQube C/C++ Plugin	3.11	**NonEmptyCaseWithoutBreak**	

Related Vulnerabilities

Search for vulnerabilities resulting from the violation of this rule on the CERT website.

Related Guidelines

SEI CERT C++ Coding Standard	VOID MSC18-CPP. Finish every set of statements associated with a case label with a break statement
CERT Oracle Secure Coding Standard for Java	MSC52-J. Finish every set of statements associated with a case label with a break statement

← ↑ →