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.

```c
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:

```c
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.

<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Severity</th>
<th>Likelihood</th>
<th>Remediaion Cost</th>
<th>Priority</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSC17-C</td>
<td>Medium</td>
<td>Likely</td>
<td>Low</td>
<td>P18</td>
<td>L1</td>
</tr>
</tbody>
</table>

**Automated Detection**

<table>
<thead>
<tr>
<th>Tool</th>
<th>Version</th>
<th>Checker</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Astrée</td>
<td>19.04</td>
<td>switch-clause-break</td>
<td>Fully checked</td>
</tr>
<tr>
<td>CodeSonar</td>
<td>5.2p0</td>
<td>LANG.STRUCT.SW.MB</td>
<td>Missing break</td>
</tr>
<tr>
<td>Compass/ROSE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coverity</td>
<td>2017.07</td>
<td>MISSING_BREAK</td>
<td>Can find instances of missing break statement between cases in <code>switch</code> statement</td>
</tr>
<tr>
<td>ECLAIR</td>
<td>1.2</td>
<td>CC2.MSC17</td>
<td>Fully implemented</td>
</tr>
<tr>
<td>Klocwork</td>
<td>2018</td>
<td>MISRA.SWITCH.WELLFORMED.BREAK.2012</td>
<td></td>
</tr>
<tr>
<td>LDRA tool suite</td>
<td>9.7.1</td>
<td>62S</td>
<td>Fully implemented</td>
</tr>
<tr>
<td>Parasoft C/C++test</td>
<td>10.4.2</td>
<td>CERT-C-MSC17-a</td>
<td>Missing break statement between cases in a switch statement</td>
</tr>
<tr>
<td>Polyspace Bug Finder</td>
<td>R2019b</td>
<td>CERT C: Rec. MSC17-C</td>
<td>Checks for missing break of switch case (rec. fully covered)</td>
</tr>
<tr>
<td>PRQA QA-C</td>
<td>9.7</td>
<td>2003</td>
<td></td>
</tr>
<tr>
<td>PVS-Studio</td>
<td>7.07</td>
<td>V796</td>
<td></td>
</tr>
<tr>
<td>RuleChecker</td>
<td>19.04</td>
<td>switch-clause-break</td>
<td>Fully checked</td>
</tr>
<tr>
<td>SonarQube C/C++ Plugin</td>
<td>3.11</td>
<td>NonEmptyCaseWithoutBreak</td>
<td></td>
</tr>
</tbody>
</table>

**Related Vulnerabilities**

Search for vulnerabilities resulting from the violation of this rule on the [CERT website](https://www.cert.org).

**Related Guidelines**

<table>
<thead>
<tr>
<th>SEI CERT C++ Coding Standard</th>
<th>VOID MSC18-CPP. Finish every set of statements associated with a case label with a break statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CERT Oracle Secure Coding Standard for Java</td>
<td>MSC52-J. Finish every set of statements associated with a case label with a break statement</td>
</tr>
</tbody>
</table>