EXP07-C. Do not diminish the benefits of constants by assuming their values in expressions

If a constant value is given for an identifier, do not diminish the maintainability of the code in which it is used by assuming its value in expressions. Simply giving the constant a name is not enough to ensure modifiability; you must be careful to always use the name, and remember that the value can change. This recommendation is related to DCL06-C. Use meaningful symbolic constants to represent literal values.

Noncompliant Code Example

The header stdio.h defines the BUFSIZ macro, which expands to an integer constant expression that is the size of the buffer used by the setbuf() function. This noncompliant code example defeats the purpose of defining BUFSIZ as a constant by assuming its value in the following expression:

```c
#include <stdio.h>
/* ... */
blocks = 1 + ((nbytes - 1) >> 9); /* BUFSIZ = 512 = 2^9 */
```

The programmer's assumption underlying this code is that "everyone knows that BUFSIZ equals 512," and right-shifting 9 bits is the same (for positive numbers) as dividing by 512. However, if BUFSIZ changes to 1024 on some systems, modifications are difficult and error prone.

Compliant Solution

This compliant solution uses the identifier assigned to the constant value in the expression:

```c
#include <stdio.h>
/* ... */
blocks = 1 + (nbytes - 1) / BUFSIZ;
```

Most modern C compilers will optimize this code appropriately.

Risk Assessment

Assuming the value of an expression diminishes the maintainability of code and can produce unexpected behavior under any circumstances in which the constant changes.

<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Severity</th>
<th>Likelihood</th>
<th>Remediation Cost</th>
<th>Priority</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXP07-C</td>
<td>Low</td>
<td>Unlikely</td>
<td>Medium</td>
<td>P2</td>
<td>L3</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>Axivion Bauhaus Suite</td>
<td>6.9.0</td>
<td>CertC-EXP07</td>
<td>Fully implemented</td>
</tr>
<tr>
<td>LDRA tool suite</td>
<td>9.7.1</td>
<td>201S</td>
<td>Partially Implemented</td>
</tr>
<tr>
<td>PRQA QA-C</td>
<td>9.7</td>
<td>3120, 3121, 3122, 3123, 3131, 3132</td>
<td></td>
</tr>
</tbody>
</table>

Related Vulnerabilities

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

Related Guidelines

| SEI CERT C++ Coding Standard | VOID EXP07-CPP. Do not diminish the benefits of constants by assuming their values in expressions |

Bibliography

| Plum 1985 | Rule 1-5 |