EXP09-C. Use sizeof to determine the size of a type or variable

Do not hard code the size of a type into an application. Because of alignment, padding, and differences in basic types (e.g., 32-bit versus 64-bit pointers), the size of most types can vary between compilers and even versions of the same compiler. Using the `sizeof` operator to determine sizes improves the clarity of what is meant and ensures that changes between compilers or versions will not affect the code.

Type alignment requirements can also affect the size of structures. For example, the size of the following structure is implementation-defined:

```c
struct s {
    int i;
    double d;
};
```

Assuming 32-bit integers and 64-bit doubles, for example, the size can range from 12 to 16 bytes, depending on alignment rules.

**Noncompliant Code Example**

This noncompliant code example attempts to declare a two-dimensional array of integers with variable length rows. On a platform with 64-bit integers, the loop will access memory outside the allocated memory section.

```c
int f(void) { /* Assuming 32-bit pointer, 32-bit integer */
    size_t i;
    int **matrix = (int **)calloc(100, 4);
    if (matrix == NULL) {
        return -1; /* Indicate calloc() failure */
    }

    for (i = 0; i < 100; i++) {
        matrix[i] = (int *)calloc(i, 4);
        if (matrix[i] == NULL) {
            return -1; /* Indicate calloc() failure */
        }
    }
    return 0;
}
```

**Compliant Solution**

This compliant solution replaces the hard-coded value 4 with `sizeof(int *)`:

```c
int f(void) {
    size_t i;
    int **matrix = (int **)calloc(100, sizeof(*matrix));
    if (matrix == NULL) {
        return -1; /* Indicate calloc() failure */
    }

    for (i = 0; i < 100; i++) {
        matrix[i] = (int *)calloc(i, sizeof(**matrix));
        if (matrix[i] == NULL) {
            return -1; /* Indicate calloc() failure */
        }
    }
    return 0;
}
```

Also see [MEM02-C](#). Immediately cast the result of a memory allocation function call into a pointer to the allocated type for a discussion on the use of the `sizeof` operator with memory allocation functions.

**Exceptions**
EXP09-C-EX1: The C Standard explicitly declares sizeof(char) == 1, so any sizes based on characters or character arrays may be evaluated without using sizeof. This does not apply to char* or any other data types.

Risk Assessment

Porting code with hard-coded sizes can result in a buffer overflow or related vulnerability.

<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>EXP09-C</td>
<td>High</td>
<td>Unlikely</td>
<td>Medium</td>
<td>P6</td>
<td>L2</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>alloc-</td>
<td>Partially checked</td>
</tr>
<tr>
<td>Compass /ROSE</td>
<td></td>
<td>without-</td>
<td></td>
</tr>
<tr>
<td>ECLAIR</td>
<td>1.2</td>
<td>CC2. EXP09</td>
<td>Can detect violations of this recommendation. In particular, it looks for</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the size argument of malloc(), calloc(), or realloc() and flags when it</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sizeof</td>
<td>does not find a sizeof operator in the argument expression. It does not</td>
</tr>
<tr>
<td></td>
<td></td>
<td>char*</td>
<td>flag if the return value is assigned to a char*. In this case a string is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>realloc()</td>
<td>being allocated, and sizeof is unnecessary because sizeof(char) == 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>realloc()</td>
<td></td>
</tr>
<tr>
<td>LDRA tool suite</td>
<td>9.7.1</td>
<td>201 S</td>
<td>Partially implemented</td>
</tr>
<tr>
<td>Polyspace Bug Finder</td>
<td>R2019b</td>
<td>CERT C: Rec. EXP09-C</td>
<td>Checks for hard-coded object size used to manipulate memory (rec. fully covered)</td>
</tr>
<tr>
<td>RuleChecker</td>
<td>19.04</td>
<td>alloc-</td>
<td>Partially checked</td>
</tr>
<tr>
<td></td>
<td></td>
<td>without-</td>
<td></td>
</tr>
</tbody>
</table>

Related Vulnerabilities

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

Related Guidelines

<table>
<thead>
<tr>
<th>SEI CERT C++ Coding Standard</th>
<th>VOID EXP09-CPP. Use sizeof to determine the size of a type or variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>MITRE CWE</td>
<td>CWE-805, Buffer access with incorrect length value</td>
</tr>
</tbody>
</table>