MSC05-C. Do not manipulate time_t typed values directly

The type time_t is specified as an "arithmetic type capable of representing times." However, the way time is encoded within this arithmetic type by the function time() is unspecified. See unspecified behavior 48 in Annex J of the C Standard. Because the encoding is unspecified, there is no safe way to manually perform arithmetic on the type, and as a result, the values should not be modified directly.

Note that POSIX specifies that the time() function must return a value of type time_t, representing time in seconds since the Epoch. POSIX-conforming applications that are not intended to be portable to other environments therefore may safely perform arithmetic operations on time_t objects.

Noncompliant Code Example

This noncompliant code example attempts to execute do_work() multiple times until at least seconds_to_work has passed. However, because the encoding is not defined, there is no guarantee that adding start to seconds_to_work will result in adding seconds_to_work seconds.

```c
int do_work(int seconds_to_work) {  
    time_t start = time(NULL);  
    if (start == (time_t)(-1)) {  
        /* Handle error */  
    }  
    while (time(NULL) < start + seconds_to_work) {  
        /* ... */  
    }  
    return 0;  
}
```

Compliant Solution

This compliant solution uses difftime() to determine the difference between two time_t values. The difftime() function returns the number of seconds, from the second parameter until the first parameter and result, as a double.

```c
int do_work(int seconds_to_work) {  
    time_t start = time(NULL);  
    time_t current = start;  
    if (start == (time_t)(-1)) {  
        /* Handle error */  
    }  
    while (difftime(current, start) < seconds_to_work) {  
        current = time(NULL);  
        if (current == (time_t)(-1)) {  
            /* Handle error */  
        }  
        /* ... */  
    }  
    return 0;  
}
```

Note that this loop still might not exit because the range of time_t might not be able to represent two times seconds_to_work apart.

Risk Assessment

Using time_t incorrectly can lead to broken logic that can place a program in an infinite loop or cause an expected logic branch to not execute.

<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>MSC05-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>Compass/ROSE</td>
<td></td>
<td></td>
<td>Can detect violations of this recommendation</td>
</tr>
<tr>
<td>ECLAIR</td>
<td>1.2</td>
<td>CC2.MSC05</td>
<td>Fully implemented</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 MSC05-CPP. Do not manipulate time_t typed values directly

Bibliography

[Kettlewell 2002] Section 4.1, "time_t"