FIO18-C. Never expect fwrite() to terminate the writing process at a null character

The C Standard, subclause 7.21.8.2 [ISO/IEC 9899:2011], defines the `fwrite()` function as follows:

> *Synopsis*
>
> *`size_t fwrite(const void *restrict ptr, size_t size, size_t nmemb, FILE *restrict stream);`*
>
> *Description*
>
> *The `fwrite()` function writes, from the array pointed to by `ptr`, up to nmemb elements whose size is specified by `size`, to the stream pointed to by `stream`. For each object, `size` calls are made to the `fputc()` function, taking the values (in order) from an array of `unsigned char` exactly overlaying the object. The file position indicator for the stream (if defined) is advanced by the number of bytes successfully written. If an error occurs, the resulting value of the file position indicator for the stream is indeterminate.*

The definition does not state that the `fwrite()` function will stop copying characters into the file if a null character is encountered. Therefore, when writing a null-terminated byte string to a file using the `fwrite()` function, always use the length of the string plus 1 (to account for the null character) as the `nmemb` parameter.

Noncompliant Code Example

In this noncompliant code example, the size of the buffer is stored in `size1`, but `size2` number of characters are written to the file. If `size2` is greater than `size1`, `write()` will not stop copying characters at the null character.

```
#include <stdio.h>
#include <stdlib.h>
char *buffer = NULL;
size_t size1;
size_t size2;
FILE *filedes;

/* Assume size1 and size2 are appropriately initialized */

filedes = fopen("out.txt", "w+");
if (filedes == NULL) {
  /* Handle error */
}

buffer = (char *)calloc( 1, size1);
if (buffer == NULL) {
  /* Handle error */
}

fwrite(buffer, 1, size2, filedes);

free(buffer);
buffer = NULL;
fclose(filedes);
```

Compliant Solution

This compliant solution ensures that the correct number of characters are written to the file:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char *buffer = NULL;
size_t size1;
size_t size2;
FILE *filedes;

/* Assume size1 is appropriately initialized */

filedes = fopen("out.txt", "w+");
if (filedes == NULL){
  /* Handle error */
}

buffer = (char *)calloc( 1, size1);
if (buffer == NULL) {
  /* Handle error */
}

/*
 * Accept characters in to the buffer.
 * Check for buffer overflow.
 */

size2 = strlen(buffer) + 1;

fwrite(buffer, 1, size2, filedes);

free(buffer);
buffer = NULL;
fclose(filedes);
```

Risk Assessment

Failure to follow the recommendation could result in a non-null-terminated string being written to a file, which will create problems when the program tries to read it back as a null-terminated byte string.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
FIO18-C	Medium	Probable	Medium	P8	L2

Automated Detection

Tool	Version	Checker	Description
LDRA tool suite	9.7.1	**44 S**	Enhanced enforcement

Related Guidelines

SEI CERT C++ Coding Standard	VOID FIO18-CPP. Never expect write() to terminate the writing process at a null character

Bibliography

[ISO/IEC 9899:2011]	Subclause 7.21.8.2, "The `fwrite` Function"
[IEEE Std 1003.1:2013]	XSH, System Interfaces, `fwrite`