FIO37-C. Do not assume that fgets() or fgetws() returns a nonempty string when successful

Errors can occur when incorrect assumptions are made about the type of data being read. These assumptions may be violated, for example, when binary data has been read from a file instead of text from a user's terminal or the output of a process is piped to `stdin`. (See FIO14-C. Understand the difference between text mode and binary mode with file streams.) On some systems, it may also be possible to input a null byte (as well as other binary codes) from the keyboard.

Subclause 7.21.7.2 of the C Standard [ISO/IEC 9899:2011] says,

> *The `fgets` function returns `s` if successful. If end-of-file is encountered and no characters have been read into the array, the contents of the array remain unchanged and a null pointer is returned.*

The wide-character function `fgetws()` has the same behavior. Therefore, if `fgets()` or `fgetws()` returns a non-null pointer, it is safe to assume that the array contains data. However, it is erroneous to assume that the array contains a nonempty string because the data may contain null characters.

Noncompliant Code Example

This noncompliant code example attempts to remove the trailing newline (`\n`) from an input line. The `fgets()` function is typically used to read a newline-terminated line of input from a stream. It takes a size parameter for the destination buffer and copies, at most, `size - 1` characters from a stream to a character array.

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

enum { BUFFER_SIZE = 1024 };

void func(void) {
  char buf[BUFFER_SIZE];

  if (fgets(buf, sizeof(buf), stdin) == NULL) {
    /* Handle error */
  }
  buf[strlen(buf) - 1] = '\0';
}
```

The `strlen()` function computes the length of a string by determining the number of characters that precede the terminating null character. A problem occurs if the first character read from the input by `fgets()` happens to be a null character. This may occur, for example, if a binary data file is read by the `fgets()` call [Lai 2006]. If the first character in `buf` is a null character, `strlen(buf)` returns 0, the expression `strlen(buf) - 1` wraps around to a large positive value, and a write-outside-array-bounds error occurs.

Compliant Solution

This compliant solution uses `strchr()` to replace the newline character in the string if it exists:

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

enum { BUFFER_SIZE = 1024 };

void func(void) {
  char buf[BUFFER_SIZE];
  char *p;

  if (fgets(buf, sizeof(buf), stdin)) {
    p = strchr(buf, '\n');
    if (p) {
      *p = '\0';
    }
  } else {
    /* Handle error */
  }
}
```

Risk Assessment

Incorrectly assuming that character data has been read can result in an out-of-bounds memory write or other flawed logic.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
FIO37-C	High	Probable	Medium	P12	L1

Automated Detection

Tool	Version	Checker	Description
Astrée	19.04		Supported: Astrée reports defects due to returned (empty) strings.
Axivion Bauhaus Suite	6.9.0	**CertC-FIO37**	
CodeSonar	5.1p0	**(general)**	Considers the possibility that `fgets()` and `fgetws()` may return empty strings (Warnings of various classes may be triggered depending on subsequent operations on those strings. For example, the noncompliant code example cited above would trigger a buffer underrun warning.)
Compass/ROSE			Could detect some violations of this rule (In particular, it could detect the noncompliant code example by searching for `fgets()`, followed by `strlen() - 1`, which could be 1. The crux of this rule is that a string returned by `fgets()` could still be empty, because the first `char` is '`\0`'. There are probably other code examples that violate this guideline; they would need to be enumerated before ROSE could detect them.)
LDRA tool suite	9.7.1	**44 S**	Enhanced enforcement
Parasoft C/C++test	10.4.2	**CERT_C-FIO37-a**	Avoid accessing arrays out of bounds
Polyspace Bug Finder	R2019b	CERT C: Rule FIO37-C	Checks for use of indeterminate string (rule fully covered)
PRQA QA-C++	4.3	**2840, 2841, 2842, 2843,** **2844, 2935, 2936, 2937,** **2938, 2939**	

Related Vulnerabilities

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

Related Guidelines

Key here (explains table format and definitions)

Taxonomy	Taxonomy item	Relationship
CERT C Secure Coding Standard	FIO14-C. Understand the difference between text mode and binary mode with file streams	Prior to 2018-01-12: CERT: Unspecified Relationship
CERT C Secure Coding Standard	FIO20-C. Avoid unintentional truncation when using fgets() or fgetws()	Prior to 2018-01-12: CERT: Unspecified Relationship
CWE 2.11	CWE-241, Improper Handling of Unexpected Data Type	2017-07-05: CERT: Rule subset of CWE

CERT-CWE Mapping Notes

Key here for mapping notes

CWE-241 and FIO37-C

CWE-241 = Union(FIO37-C, list) where list =

- Improper handling of unexpected data type that does not come from the fgets() function.

Bibliography

[ISO/IEC 9899:2011]	Subclause 7.21.7.2, "The `fgets` Function" Subclause 7.29.3.2, "The `fgetws` Function"
[Lai 2006]	
[Seacord 2013]	Chapter 2, "Strings"