FIO20-C. Avoid unintentional truncation when using fgets() or fgetws()

The `fgets()` and `fgetws()` functions are typically used to read a newline-terminated line of input from a stream. Both functions read at most one less than the number of narrow or wide characters specified by an argument `n` from a stream to a string. Truncation errors can occur if `n - 1` is less than the number of characters appearing in the input string prior to the new-line narrow or wide character (which is retained) or after end-of-file. This can result in the accidental truncation of user input.

Noncompliant Code Example

This noncompliant code example copies the input string into a buffer, and assumes it captured all of the user's input.

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

bool get_data(char *buffer, int size) {
  if (fgets(buffer, size, stdin)) {
    return true;
  }
  return false;
}

void func(void) {
  char buf[8];
  if (get_data(buf, sizeof(buf))) {
    printf("The user input %s\n", buf);
  } else {
    printf("Error getting data from the user\n");
  }
}
```

However, if the last character in `buf` is not a newline and the stream is not at the end-of-file marker, the buffer was too small to contain all of the data from the user. For example, because the buffer is only 8 characters in length, if the user input `"Hello World\n"`, the buffer would contain `"Hello W"` terminated by a null character.

Compliant Solution (Fail on Truncation)

This compliant solution examines the end-of-file marker for the stream and the last character in the buffer to determine whether it is a newline or not. If it is the end of file, or the last character is a newline, then the buffer contains all of the user's input. However, if the last character is not at the end-of-file and not a newline then the user's input has been truncated.

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

bool get_data(char *buffer, int size) {
  if (fgets(buffer, size, stdin)) {
    size_t len = strlen(buffer);
    return feof(stdin) || (len != 0 && buffer[len-1] == '\n');
  }
  return false;
}

void func(void) {
  char buf[8];
  if (get_data(buf, sizeof(buf))) {
    printf("The user input %s\n", buf);
  } else {
    printf("Error getting data from the user\n");
  }
}
```

Compliant Solution (Expanding Buffer)

This compliant solution solves the problem by expanding the buffer to read the entire contents from `stdin` instead of failing if the caller did not allocate enough space. If the allocation fails, it will return `NULL`, but otherwise, it returns a buffer of the received data, which the caller must free.

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

char *get_filled_buffer(void) {
  char temp[32];
  char *ret = NULL;
  size_t full_length = 0;

  while (fgets(temp, sizeof(temp), stdin)) {
    size_t len = strlen(temp);
    if (SIZE_MAX - len - 1 < full_length) {
      break;
    }
    char *r_temp = realloc(ret, full_length + len + 1);
    if (r_temp == NULL) {
      break;
    }
    ret = r_temp;
    strcpy(ret + full_length, temp); /* concatenate */
    full_length += len;

    if (feof(stdin) || temp[len-1] == '\n') {
      return ret;
    }
  }

  free(ret);
  return NULL;
}
```

Compliant Solution (POSIX `getline()`)

The `getline()` function was originally a GNU extension, but is now standard in POSIX.1-2008. It also fills a string with characters from an input stream. In this case, the program passes it a NULL pointer for a string, indicating that `getline()` should allocate sufficient space for the string and the caller frees it later.

```
#include <stdio.h>

void func(void) {
  char* buf = NULL;
  size_t dummy = 0;
  if (getline(&buf, &dummy, stdin) == -1) {
        /* handle error */
  }
  printf("The user input %s\n", buf);
  free(buf);
}
```

Risk Assessment

Incorrectly assuming a newline character is read by `fgets()` or `fgetws()` can result in data truncation.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
FIO20-C	Medium	Likely	Medium	P12	L1

Automated Detection

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

Related Vulnerabilities

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

Bibliography

[Lai 2006]	
[Seacord 2013]	Chapter 2, "Strings"