# CWE Detail – CWE-464

## Description

The accidental addition of a data-structure sentinel can cause serious programming logic problems.

## Extended Description

Data-structure sentinels are often used to mark the structure of data. A common example of this is the null character at the end of strings or a special sentinel to mark the end of a linked list. It is dangerous to allow this type of control data to be easily accessible. Therefore, it is important to protect from the addition or modification of sentinels.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Modify Application Data — Notes: Generally this error will cause the data structure to not work properly by truncating the data.

## Potential Mitigations

**•** Implementation: Encapsulate the user from interacting with data sentinels. Validate user input to verify that sentinels are not present. (Effectiveness: N/A)

**•** Implementation: Proper error checking can reduce the risk of inadvertently introducing sentinel values into data. For example, if a parsing function fails or encounters an error, it might return a value that is the same as the sentinel. (Effectiveness: N/A)

**•** Architecture and Design: Use an abstraction library to abstract away risky APIs. This is not a complete solution. (Effectiveness: N/A)

**•** Operation: Use OS-level preventative functionality. This is not a complete solution. (Effectiveness: N/A)

## Applicable Platforms

**•** C (Class: None, Prevalence: Undetermined)

**•** C++ (Class: None, Prevalence: Undetermined)

## Demonstrative Examples

**•** The first print statement will print each character separated by a space. However, if a NULL byte is read from stdin by fgetc, then it will return 0. When foo is printed as a string, the 0 at character foo[2] will act as a NULL terminator and foo[3] will never be printed.