# CWE Detail – CWE-135

## Description

The product does not correctly calculate the length of strings that can contain wide or multi-byte characters.

## Extended Description

N/A

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Modes of Introduction

**•** Implementation: There are several ways in which improper string length checking may result in an exploitable condition. All of these, however, involve the introduction of buffer overflow conditions in order to reach an exploitable state. The first of these issues takes place when the output of a wide or multi-byte character string, string-length function is used as a size for the allocation of memory. While this will result in an output of the number of characters in the string, note that the characters are most likely not a single byte, as they are with standard character strings. So, using the size returned as the size sent to new or malloc and copying the string to this newly allocated memory will result in a buffer overflow. Another common way these strings are misused involves the mixing of standard string and wide or multi-byte string functions on a single string. Invariably, this mismatched information will result in the creation of a possibly exploitable buffer overflow condition.

## Common Consequences

**•** Impact: Execute Unauthorized Code or Commands — Notes: This weakness may lead to a buffer overflow. Buffer overflows often can be used to execute arbitrary code, which is usually outside the scope of a program's implicit security policy. This can often be used to subvert any other security service.

**•** Impact: Read Memory, DoS: Crash, Exit, or Restart, DoS: Resource Consumption (CPU), DoS: Resource Consumption (Memory) — Notes: Out of bounds memory access will very likely result in the corruption of relevant memory, and perhaps instructions, possibly leading to a crash. Other attacks leading to lack of availability are possible, including putting the program into an infinite loop.

**•** Impact: Read Memory — Notes: In the case of an out-of-bounds read, the attacker may have access to sensitive information. If the sensitive information contains system details, such as the current buffer's position in memory, this knowledge can be used to craft further attacks, possibly with more severe consequences.

## Potential Mitigations

**•** Implementation: Always verify the length of the string unit character. (Effectiveness: N/A)

**•** Implementation: Use length computing functions (e.g. strlen, wcslen, etc.) appropriately with their equivalent type (e.g.: byte, wchar\_t, etc.) (Effectiveness: N/A)

## Applicable Platforms

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

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

## Demonstrative Examples

**•** The output from the printf() statement would be: