# CWE Detail – CWE-761

## Description

The product calls free() on a pointer to a memory resource that was allocated on the heap, but the pointer is not at the start of the buffer.

## Extended Description

This can cause the product to crash, or in some cases, modify critical program variables or execute code. This weakness often occurs when the memory is allocated explicitly on the heap with one of the malloc() family functions and free() is called, but pointer arithmetic has caused the pointer to be in the interior or end of the buffer.

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## Observed Examples (CVEs)

**•** CVE-2019-11930: function "internally calls 'calloc' and returns a pointer at an index... inside the allocated buffer. This led to freeing invalid memory."

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Modify Memory, DoS: Crash, Exit, or Restart, Execute Unauthorized Code or Commands — Notes:

## Potential Mitigations

**•** Implementation: When utilizing pointer arithmetic to traverse a buffer, use a separate variable to track progress through memory and preserve the originally allocated address for later freeing. (Effectiveness: N/A)

**•** Implementation: When programming in C++, consider using smart pointers provided by the boost library to help correctly and consistently manage memory. (Effectiveness: N/A)

**•** Architecture and Design: Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, glibc in Linux provides protection against free of invalid pointers. (Effectiveness: N/A)

**•** Architecture and Design: Use a language that provides abstractions for memory allocation and deallocation. (Effectiveness: N/A)

**•** Testing: Use a tool that dynamically detects memory management problems, such as valgrind. (Effectiveness: N/A)

## Demonstrative Examples

**•** However, if the character is not at the beginning of the string, or if it is not in the string at all, then the pointer will not be at the start of the buffer when the programmer frees it.

**•** Since strsep is not allocating any new memory, freeing an element in the middle of the array is equivalent to free a pointer in the middle of inputstring.

**•** While the above code attempts to free memory associated with bad commands, since the memory was all allocated in one chunk, it must all be freed together.

## Notes

**•** Maintenance: Currently, CWE-763 is the parent, however it may be desirable to have an intermediate parent which is not function-specific, similar to how CWE-762 is an intermediate parent between CWE-763 and CWE-590.