# CWE Detail – CWE-590

## Description

The product calls free() on a pointer to memory that was not allocated using associated heap allocation functions such as malloc(), calloc(), or realloc().

## Extended Description

When free() is called on an invalid pointer, the program's memory management data structures may become corrupted. This corruption can cause the program to crash or, in some circumstances, an attacker may be able to cause free() to operate on controllable memory locations to modify critical program variables or execute code.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Execute Unauthorized Code or Commands, Modify Memory — Notes: There is the potential for arbitrary code execution with privileges of the vulnerable program via a "write, what where" primitive. If pointers to memory which hold user information are freed, a malicious user will be able to write 4 bytes anywhere in memory.

## Potential Mitigations

**•** Implementation: Only free pointers that you have called malloc on previously. This is the recommended solution. Keep track of which pointers point at the beginning of valid chunks and free them only once. (Effectiveness: N/A)

**•** Implementation: Before freeing a pointer, the programmer should make sure that the pointer was previously allocated on the heap and that the memory belongs to the programmer. Freeing an unallocated pointer will cause undefined behavior in the program. (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

**•** This example shows the array allocated globally, as part of the data segment of memory and the programmer attempts to call free() on the array.

## Notes

**•** Other: In C++, if the new operator was used to allocate the memory, it may be allocated with the malloc(), calloc() or realloc() family of functions in the implementation. Someone aware of this behavior might choose to map this problem to CWE-590 or to its parent, CWE-762, depending on their perspective.