# CWE Detail – CWE-415

## Description

The product calls free() twice on the same memory address.

## Extended Description

N/A

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2006-5051: Chain: Signal handler contains too much functionality (CWE-828), introducing a race condition (CWE-362) that leads to a double free (CWE-415).

**•** CVE-2004-0642: Double free resultant from certain error conditions.

**•** CVE-2004-0772: Double free resultant from certain error conditions.

**•** CVE-2005-1689: Double free resultant from certain error conditions.

**•** CVE-2003-0545: Double free from invalid ASN.1 encoding.

**•** CVE-2003-1048: Double free from malformed GIF.

**•** CVE-2005-0891: Double free from malformed GIF.

**•** CVE-2002-0059: Double free from malformed compressed data.

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Modify Memory, Execute Unauthorized Code or Commands — Notes:

## Potential Mitigations

**•** Architecture and Design: Choose a language that provides automatic memory management. (Effectiveness: N/A)

**•** Implementation: Ensure that each allocation is freed only once. After freeing a chunk, set the pointer to NULL to ensure the pointer cannot be freed again. In complicated error conditions, be sure that clean-up routines respect the state of allocation properly. If the language is object oriented, ensure that object destructors delete each chunk of memory only once. (Effectiveness: N/A)

**•** Implementation: Use a static analysis tool to find double free instances. (Effectiveness: N/A)

## Applicable Platforms

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

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

## Demonstrative Examples

**•** Double free vulnerabilities have two common (and sometimes overlapping) causes:

**•** N/A

## Notes

**•** Relationship: This is usually resultant from another weakness, such as an unhandled error or race condition between threads. It could also be primary to weaknesses such as buffer overflows.

**•** Theoretical: It could be argued that Double Free would be most appropriately located as a child of "Use after Free", but "Use" and "Release" are considered to be distinct operations within vulnerability theory, therefore this is more accurately "Release of a Resource after Expiration or Release", which doesn't exist yet.