# CWE Detail – CWE-401

## Description

The product does not sufficiently track and release allocated memory after it has been used, making the memory unavailable for reallocation and reuse.

## Extended Description

N/A

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2005-3119: Memory leak because function does not free() an element of a data structure.

**•** CVE-2004-0427: Memory leak when counter variable is not decremented.

**•** CVE-2002-0574: chain: reference count is not decremented, leading to memory leak in OS by sending ICMP packets.

**•** CVE-2005-3181: Kernel uses wrong function to release a data structure, preventing data from being properly tracked by other code.

**•** CVE-2004-0222: Memory leak via unknown manipulations as part of protocol test suite.

**•** CVE-2001-0136: Memory leak via a series of the same command.

## Modes of Introduction

**•** Implementation: Memory leaks have two common and sometimes overlapping causes: Error conditions and other exceptional circumstances often triggered by improper handling of malformed data or unexpectedly interrupted sessions. Confusion over which part of the program is responsible for freeing the memory, since in some languages, developers are responsible for tracking memory allocation and releasing the memory. If there are no more pointers or references to the memory, then it can no longer be tracked and identified for release.

## Common Consequences

**•** Impact: DoS: Crash, Exit, or Restart, DoS: Instability, DoS: Resource Consumption (CPU), DoS: Resource Consumption (Memory) — Notes: Most memory leaks result in general product reliability problems, but if an attacker can intentionally trigger a memory leak, the attacker might be able to launch a denial of service attack (by crashing or hanging the program) or take advantage of other unexpected program behavior resulting from a low memory condition.

**•** Impact: Reduce Performance — Notes:

## Potential Mitigations

**•** Implementation: Choose a language or tool that provides automatic memory management, or makes manual memory management less error-prone. For example, glibc in Linux provides protection against free of invalid pointers. When using Xcode to target OS X or iOS, enable automatic reference counting (ARC) [REF-391]. To help correctly and consistently manage memory when programming in C++, consider using a smart pointer class such as std::auto\_ptr (defined by ISO/IEC ISO/IEC 14882:2003), std::shared\_ptr and std::unique\_ptr (specified by an upcoming revision of the C++ standard, informally referred to as C++ 1x), or equivalent solutions such as Boost. (Effectiveness: N/A)

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

**•** Architecture and Design: The Boehm-Demers-Weiser Garbage Collector or valgrind can be used to detect leaks in code. (Effectiveness: N/A)

## Applicable Platforms

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

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

## Demonstrative Examples

**•** N/A

## Notes

**•** Relationship: This is often a resultant weakness due to improper handling of malformed data or early termination of sessions.

**•** Terminology: "memory leak" has sometimes been used to describe other kinds of issues, e.g. for information leaks in which the contents of memory are inadvertently leaked (CVE-2003-0400 is one such example of this terminology conflict).