# CWE Detail – CWE-416

## Description

The product reuses or references memory after it has been freed. At some point afterward, the memory may be allocated again and saved in another pointer, while the original pointer references a location somewhere within the new allocation. Any operations using the original pointer are no longer valid because the memory "belongs" to the code that operates on the new pointer.

## Extended Description

N/A

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2022-20141: Chain: an operating system kernel has insufficent resource locking (CWE-413) leading to a use after free (CWE-416).

**•** CVE-2022-2621: Chain: two threads in a web browser use the same resource (CWE-366), but one of those threads can destroy the resource before the other has completed (CWE-416).

**•** CVE-2021-0920: Chain: mobile platform race condition (CWE-362) leading to use-after-free (CWE-416), as exploited in the wild per CISA KEV. (KEV)

**•** CVE-2020-6819: Chain: race condition (CWE-362) leads to use-after-free (CWE-416), as exploited in the wild per CISA KEV. (KEV)

**•** CVE-2010-4168: Use-after-free triggered by closing a connection while data is still being transmitted.

**•** CVE-2010-2941: Improper allocation for invalid data leads to use-after-free.

**•** CVE-2010-2547: certificate with a large number of Subject Alternate Names not properly handled in realloc, leading to use-after-free

**•** CVE-2010-1772: Timers are not disabled when a related object is deleted

**•** CVE-2010-1437: Access to a "dead" object that is being cleaned up

**•** CVE-2010-1208: object is deleted even with a non-zero reference count, and later accessed

**•** CVE-2010-0629: use-after-free involving request containing an invalid version number

**•** CVE-2010-0378: unload of an object that is currently being accessed by other functionality

**•** CVE-2010-0302: incorrectly tracking a reference count leads to use-after-free

**•** CVE-2010-0249: use-after-free related to use of uninitialized memory

**•** CVE-2010-0050: HTML document with incorrectly-nested tags

**•** CVE-2009-3658: Use after free in ActiveX object by providing a malformed argument to a method

**•** CVE-2009-3616: use-after-free by disconnecting during data transfer, or a message containing incorrect data types

**•** CVE-2009-3553: disconnect during a large data transfer causes incorrect reference count, leading to use-after-free

**•** CVE-2009-2416: use-after-free found by fuzzing

**•** CVE-2009-1837: Chain: race condition (CWE-362) from improper handling of a page transition in web client while an applet is loading (CWE-368) leads to use after free (CWE-416)

**•** CVE-2009-0749: realloc generates new buffer and pointer, but previous pointer is still retained, leading to use after free

**•** CVE-2010-3328: Use-after-free in web browser, probably resultant from not initializing memory.

**•** CVE-2008-5038: use-after-free when one thread accessed memory that was freed by another thread

**•** CVE-2008-0077: assignment of malformed values to certain properties triggers use after free

**•** CVE-2006-4434: mail server does not properly handle a long header.

**•** CVE-2010-2753: chain: integer overflow leads to use-after-free

**•** CVE-2006-4997: freed pointer dereference

**•** CVE-2003-0813: Chain: A multi-threaded race condition (CWE-367) allows attackers to cause two threads to process the same RPC request, which causes a use-after-free (CWE-416) in one thread

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Modify Memory — Notes: The use of previously freed memory may corrupt valid data, if the memory area in question has been allocated and used properly elsewhere.

**•** Impact: DoS: Crash, Exit, or Restart — Notes: If chunk consolidation occurs after the use of previously freed data, the process may crash when invalid data is used as chunk information.

**•** Impact: Execute Unauthorized Code or Commands — Notes: If malicious data is entered before chunk consolidation can take place, it may be possible to take advantage of a write-what-where primitive to execute arbitrary code. If the newly allocated data happens to hold a class, in C++ for example, various function pointers may be scattered within the heap data. If one of these function pointers is overwritten with an address to valid shellcode, execution of arbitrary code can be achieved.

## Potential Mitigations

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

**•** Implementation: When freeing pointers, be sure to set them to NULL once they are freed. However, the utilization of multiple or complex data structures may lower the usefulness of this strategy. (Effectiveness: Defense in Depth)

## Applicable Platforms

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

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

## Demonstrative Examples

**•** N/A

**•** When an error occurs, the pointer is immediately freed. However, this pointer is later incorrectly used in the logError function.