# CWE Detail – CWE-788

## Description

The product reads or writes to a buffer using an index or pointer that references a memory location after the end of the buffer.

## Extended Description

This typically occurs when a pointer or its index is incremented to a position after the buffer; or when pointer arithmetic results in a position after the buffer.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2009-2550: Classic stack-based buffer overflow in media player using a long entry in a playlist

**•** CVE-2009-2403: Heap-based buffer overflow in media player using a long entry in a playlist

**•** CVE-2009-0689: large precision value in a format string triggers overflow

**•** CVE-2009-0558: attacker-controlled array index leads to code execution

**•** CVE-2008-4113: OS kernel trusts userland-supplied length value, allowing reading of sensitive information

**•** CVE-2007-4268: Chain: integer signedness error (CWE-195) passes signed comparison, leading to heap overflow (CWE-122)

## Common Consequences

**•** Impact: Read Memory — Notes: For an out-of-bounds read, the attacker may have access to sensitive information. If the sensitive information contains system details, such as the current buffer's position in memory, this knowledge can be used to craft further attacks, possibly with more severe consequences.

**•** Impact: Modify Memory, DoS: Crash, Exit, or Restart — Notes: Out of bounds memory access will very likely result in the corruption of relevant memory, and perhaps instructions, possibly leading to a crash. Other attacks leading to lack of availability are possible, including putting the program into an infinite loop.

**•** Impact: Modify Memory, Execute Unauthorized Code or Commands — Notes: If the memory accessible by the attacker can be effectively controlled, it may be possible to execute arbitrary code, as with a standard buffer overflow. If the attacker can overwrite a pointer's worth of memory (usually 32 or 64 bits), they can redirect a function pointer to their own malicious code. Even when the attacker can only modify a single byte arbitrary code execution can be possible. Sometimes this is because the same problem can be exploited repeatedly to the same effect. Other times it is because the attacker can overwrite security-critical application-specific data -- such as a flag indicating whether the user is an administrator.

## Applicable Platforms

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

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

## Demonstrative Examples

**•** This function allocates a buffer of 64 bytes to store the hostname, however there is no guarantee that the hostname will not be larger than 64 bytes. If an attacker specifies an address which resolves to a very large hostname, then the function may overwrite sensitive data or even relinquish control flow to the attacker.

**•** If returnChunkSize() happens to encounter an error it will return -1. Notice that the return value is not checked before the memcpy operation (CWE-252), so -1 can be passed as the size argument to memcpy() (CWE-805). Because memcpy() assumes that the value is unsigned, it will be interpreted as MAXINT-1 (CWE-195), and therefore will copy far more memory than is likely available to the destination buffer (CWE-787, CWE-788).

**•** The programmer attempts to encode the ampersand character in the user-controlled string, however the length of the string is validated before the encoding procedure is applied. Furthermore, the programmer assumes encoding expansion will only expand a given character by a factor of 4, while the encoding of the ampersand expands by 5. As a result, when the encoding procedure expands the string it is possible to overflow the destination buffer if the attacker provides a string of many ampersands.

**•** However, the message length variable from the structure is used as the condition for ending the for loop without validating that the message length variable accurately reflects the length of the message body (CWE-606). This can result in a buffer over-read (CWE-125) by reading from memory beyond the bounds of the buffer if the message length variable indicates a length that is longer than the size of a message body (CWE-130).