# CWE Detail – CWE-194

## Description

The product performs an operation on a number that causes it to be sign extended when it is transformed into a larger data type. When the original number is negative, this can produce unexpected values that lead to resultant weaknesses.

## Extended Description

N/A

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2018-10887: Chain: unexpected sign extension (CWE-194) leads to integer overflow (CWE-190), causing an out-of-bounds read (CWE-125)

**•** CVE-1999-0234: Sign extension error produces -1 value that is treated as a command separator, enabling OS command injection.

**•** CVE-2003-0161: Product uses "char" type for input character. When char is implemented as a signed type, ASCII value 0xFF (255), a sign extension produces a -1 value that is treated as a program-specific separator value, effectively disabling a length check and leading to a buffer overflow. This is also a multiple interpretation error.

**•** CVE-2007-4988: chain: signed short width value in image processor is sign extended during conversion to unsigned int, which leads to integer overflow and heap-based buffer overflow.

**•** CVE-2006-1834: chain: signedness error allows bypass of a length check; later sign extension makes exploitation easier.

**•** CVE-2005-2753: Sign extension when manipulating Pascal-style strings leads to integer overflow and improper memory copy.

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Read Memory, Modify Memory, Other — Notes: When an unexpected sign extension occurs in code that operates directly on memory buffers, such as a size value or a memory index, then it could cause the program to write or read outside the boundaries of the intended buffer. If the numeric value is associated with an application-level resource, such as a quantity or price for a product in an e-commerce site, then the sign extension could produce a value that is much higher (or lower) than the application's allowable range.

## Potential Mitigations

**•** Implementation: Avoid using signed variables if you don't need to represent negative values. When negative values are needed, perform validation after you save those values to larger data types, or before passing them to functions that are expecting unsigned values. (Effectiveness: N/A)

## Applicable Platforms

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

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

## Demonstrative Examples

**•** This code first exhibits an example of CWE-839, allowing "s" to be a negative number. When the negative short "s" is converted to an unsigned integer, it becomes an extremely large positive integer. When this converted integer is used by strncpy() it will lead to a buffer overflow (CWE-119).

## Notes

**•** Relationship: Sign extension errors can lead to buffer overflows and other memory-based problems. They are also likely to be factors in other weaknesses that are not based on memory operations, but rely on numeric calculation.

**•** Maintenance: This entry is closely associated with signed-to-unsigned conversion errors (CWE-195) and other numeric errors. These relationships need to be more closely examined within CWE.