# CWE Detail – CWE-192

## Description

Integer coercion refers to a set of flaws pertaining to the type casting, extension, or truncation of primitive data types.

## Extended Description

Several flaws fall under the category of integer coercion errors. For the most part, these errors in and of themselves result only in availability and data integrity issues. However, in some circumstances, they may result in other, more complicated security related flaws, such as buffer overflow conditions.

## Threat-Mapped Scoring

Score: 1.5

Priority: P4 - Informational (Low)

## Observed Examples (CVEs)

**•** CVE-2022-2639: Chain: integer coercion error (CWE-192) prevents a return value from indicating an error, leading to out-of-bounds write (CWE-787)

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: DoS: Resource Consumption (CPU), DoS: Resource Consumption (Memory), DoS: Crash, Exit, or Restart — Notes: Integer coercion often leads to undefined states of execution resulting in infinite loops or crashes.

**•** Impact: Execute Unauthorized Code or Commands — Notes: In some cases, integer coercion errors can lead to exploitable buffer overflow conditions, resulting in the execution of arbitrary code.

**•** Impact: Other — Notes: Integer coercion errors result in an incorrect value being stored for the variable in question.

## Potential Mitigations

**•** Requirements: A language which throws exceptions on ambiguous data casts might be chosen. (Effectiveness: N/A)

**•** Architecture and Design: Design objects and program flow such that multiple or complex casts are unnecessary (Effectiveness: N/A)

**•** Implementation: Ensure that any data type casting that you must used is entirely understood in order to reduce the plausibility of error in use. (Effectiveness: N/A)

## Applicable Platforms

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

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

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

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

## Demonstrative Examples

**•** The code performs a check to make sure that the packet does not contain too many headers. However, numHeaders is defined as a signed int, so it could be negative. If the incoming packet specifies a value such as -3, then the malloc calculation will generate a negative number (say, -300 if each header can be a maximum of 100 bytes). When this result is provided to malloc(), it is first converted to a size\_t type. This conversion then produces a large value such as 4294966996, which may cause malloc() to fail or to allocate an extremely large amount of memory (CWE-195). With the appropriate negative numbers, an attacker could trick malloc() into using a very small positive number, which then allocates a buffer that is much smaller than expected, potentially leading to a buffer overflow.

**•** 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

**•** Maintenance: Within C, it might be that "coercion" is semantically different than "casting", possibly depending on whether the programmer directly specifies the conversion, or if the compiler does it implicitly. This has implications for the presentation of this entry and others, such as CWE-681, and whether there is enough of a difference for these entries to be split.