# CWE Detail – CWE-197

## Description

Truncation errors occur when a primitive is cast to a primitive of a smaller size and data is lost in the conversion.

## Extended Description

When a primitive is cast to a smaller primitive, the high order bits of the large value are lost in the conversion, potentially resulting in an unexpected value that is not equal to the original value. This value may be required as an index into a buffer, a loop iterator, or simply necessary state data. In any case, the value cannot be trusted and the system will be in an undefined state. While this method may be employed viably to isolate the low bits of a value, this usage is rare, and truncation usually implies that an implementation error has occurred.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2020-17087: Chain: integer truncation (CWE-197) causes small buffer allocation (CWE-131) leading to out-of-bounds write (CWE-787) in kernel pool, as exploited in the wild per CISA KEV. (KEV)

**•** CVE-2009-0231: Integer truncation of length value leads to heap-based buffer overflow.

**•** CVE-2008-3282: Size of a particular type changes for 64-bit platforms, leading to an integer truncation in document processor causes incorrect index to be generated.

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Modify Memory — Notes: The true value of the data is lost and corrupted data is used.

## Potential Mitigations

**•** Implementation: Ensure that no casts, implicit or explicit, take place that move from a larger size primitive or a smaller size primitive. (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 above code, when compiled and run on certain systems, returns the following output:

**•** However, a numeric truncation error can occur if the integer values are higher than the maximum value allowed for the primitive type short. This can cause unexpected results or loss or corruption of data. In this case the sales database may be corrupted with incorrect data. Explicit casting from a from a larger size primitive type to a smaller size primitive type should be prevented. The following example an if statement is added to validate that the integer values less than the maximum value for the primitive type short before the explicit cast and the call to the sales method.

## Notes

**•** Research Gap: This weakness has traditionally been under-studied and under-reported, although vulnerabilities in popular software have been published in 2008 and 2009.