# CWE Detail – CWE-1335

## Description

An integer value is specified to be shifted by a negative amount or an amount greater than or equal to the number of bits contained in the value causing an unexpected or indeterminate result.

## Extended Description

Specifying a value to be shifted by a negative amount is undefined in various languages. Various computer architectures implement this action in different ways. The compilers and interpreters when generating code to accomplish a shift generally do not do a check for this issue. Specifying an over-shift, a shift greater than or equal to the number of bits contained in a value to be shifted, produces a result which varies by architecture and compiler. In some languages, this action is specifically listed as producing an undefined result.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2009-4307: An unexpected large value in the ext4 filesystem causes an overshift condition resulting in a divide by zero.

**•** CVE-2012-2100: An unexpected large value in the ext4 filesystem causes an overshift condition resulting in a divide by zero - fix of CVE-2009-4307.

**•** CVE-2020-8835: An overshift in a kernel allowed out of bounds reads and writes resulting in a root takeover.

**•** CVE-2015-1607: Program is not properly handling signed bitwise left-shifts causing an overlapping memcpy memory range error.

**•** CVE-2016-9842: Compression function improperly executes a signed left shift of a negative integer.

**•** CVE-2018-18445: Some kernels improperly handle right shifts of 32 bit numbers in a 64 bit register.

**•** CVE-2013-4206: Putty has an incorrectly sized shift value resulting in an overshift.

**•** CVE-2018-20788: LED driver overshifts under certain conditions resulting in a DoS.

## Modes of Introduction

**•** Implementation: Adding shifts without properly verifying the size and sign of the shift amount.

## Common Consequences

**•** Impact: DoS: Crash, Exit, or Restart — Notes:

## Potential Mitigations

**•** Implementation: Implicitly or explicitly add checks and mitigation for negative or over-shift values. (Effectiveness: N/A)

## Applicable Platforms

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

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

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

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

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

## Demonstrative Examples

**•** The example above ends up with a shift amount of -5. The hexadecimal value is FFFFFFFFFFFFFFFD which, when bits above the 6th bit are masked off, the shift amount becomes a binary shift value of 111101 which is 61 decimal. A shift of 61 produces a very different result than -5. The previous example is a very simple version of the following code which is probably more realistic of what happens in a real system.