# CWE Detail – CWE-783

## Description

The product uses an expression in which operator precedence causes incorrect logic to be used.

## Extended Description

While often just a bug, operator precedence logic errors can have serious consequences if they are used in security-critical code, such as making an authentication decision.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2008-2516: Authentication module allows authentication bypass because it uses "(x = call(args) == SUCCESS)" instead of "((x = call(args)) == SUCCESS)".

**•** CVE-2008-0599: Chain: Language interpreter calculates wrong buffer size (CWE-131) by using "size = ptr ? X : Y" instead of "size = (ptr ? X : Y)" expression.

**•** CVE-2001-1155: Chain: product does not properly check the result of a reverse DNS lookup because of operator precedence (CWE-783), allowing bypass of DNS-based access restrictions.

## Modes of Introduction

**•** Implementation: Logic errors related to operator precedence may cause problems even during normal operation, so they are probably discovered quickly during the testing phase. If testing is incomplete or there is a strong reliance on manual review of the code, then these errors may not be discovered before the software is deployed.

## Common Consequences

**•** Impact: Varies by Context, Unexpected State — Notes: The consequences will vary based on the context surrounding the incorrect precedence. In a security decision, integrity or confidentiality are the most likely results. Otherwise, a crash may occur due to the software reaching an unexpected state.

## Potential Mitigations

**•** Implementation: Regularly wrap sub-expressions in parentheses, especially in security-critical code. (Effectiveness: N/A)

## Applicable Platforms

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

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

**•** None (Class: Not Language-Specific, Prevalence: Rarely)

## Demonstrative Examples

**•** However, the method that authenticates the username and password is called within an if statement with incorrect operator precedence logic. Because the comparison operator "==" has a higher precedence than the assignment operator "=", the comparison operator will be evaluated first and if the method returns FAIL then the comparison will be true, the return variable will be set to true and SUCCESS will be returned. This operator precedence logic error can be easily resolved by properly using parentheses within the expression of the if statement, as shown below.

**•** However, the return on investment calculation will not produce correct results because of the incorrect operator precedence logic in the equation. The divide operator has a higher precedence than the minus operator, therefore the equation will divide the initial investment costs by the initial investment costs which will only subtract one from the current value. Again this operator precedence logic error can be resolved by the correct use of parentheses within the equation, as shown below.