# CWE Detail – CWE-456

## Description

The product does not initialize critical variables, which causes the execution environment to use unexpected values.

## Extended Description

N/A

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2020-6078: Chain: The return value of a function returning a pointer is not checked for success (CWE-252) resulting in the later use of an uninitialized variable (CWE-456) and a null pointer dereference (CWE-476)

**•** CVE-2019-3836: Chain: secure communications library does not initialize a local variable for a data structure (CWE-456), leading to access of an uninitialized pointer (CWE-824).

**•** CVE-2018-14641: Chain: C union member is not initialized (CWE-456), leading to access of invalid pointer (CWE-824)

**•** CVE-2009-2692: Chain: Use of an unimplemented network socket operation pointing to an uninitialized handler function (CWE-456) causes a crash because of a null pointer dereference (CWE-476).

**•** CVE-2020-20739: A variable that has its value set in a conditional statement is sometimes used when the conditional fails, sometimes causing data leakage

**•** CVE-2005-2978: Product uses uninitialized variables for size and index, leading to resultant buffer overflow.

**•** CVE-2005-2109: Internal variable in PHP application is not initialized, allowing external modification.

**•** CVE-2005-2193: Array variable not initialized in PHP application, leading to resultant SQL injection.

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Unexpected State, Quality Degradation, Varies by Context — Notes: The uninitialized data may be invalid, causing logic errors within the program. In some cases, this could result in a security problem.

## Potential Mitigations

**•** Implementation: Check that critical variables are initialized. (Effectiveness: N/A)

**•** Testing: Use a static analysis tool to spot non-initialized variables. (Effectiveness: N/A)

## Applicable Platforms

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

## Demonstrative Examples

**•** This code attempts to extract two integer values out of a formatted, user-supplied input. However, if an attacker were to provide an input of the form:

**•** N/A

**•** The $isAdmin variable is set to true if the user is an admin, but is uninitialized otherwise. If PHP's register\_globals feature is enabled, an attacker can set uninitialized variables like $isAdmin to arbitrary values, in this case gaining administrator privileges by setting $isAdmin to true.

**•** However, if the method setUser is not called before authenticateUser then the user variable will not have been initialized and will result in a NullPointerException. The code should verify that the user variable has been initialized before it is used, as in the following code.

**•** When the printf() is reached,
 test\_string might be an unexpected address, so the
 printf might print junk strings (CWE-457). To fix this code, there are a couple approaches to
 making sure that test\_string has been properly set once
 it reaches the printf(). One solution would be to set test\_string to an
 acceptable default before the conditional:

**•** In PgServiceResolver.java, when pgType is "card" indicating a card
 payment, orderPgData.validateAmount() is not called - that is, the
 amount is not validated to be the same as the expected price. Since isPaymentAmountTampered is declared as a private boolean, but it
 is not initialized, it is forcibly initialized to false by the Java
 compiler [REF-1476]. If the adversary modifies the price, e.g., changing paymentAmount from
 100 to 10, then no validation is performed. Since
 isPaymentAmountTampered is "false" because of the default
 initialization, the code finishes processing the payment because it
 does not believe that the amount has been changed.

## Notes

**•** Relationship: This weakness is a major factor in a number of resultant weaknesses, especially in web applications that allow global variable initialization (such as PHP) with libraries that can be directly requested.

**•** Research Gap: It is highly likely that a large number of resultant weaknesses have missing initialization as a primary factor, but researcher reports generally do not provide this level of detail.