# CWE Detail – CWE-843

## Description

The product allocates or initializes a resource such as a pointer, object, or variable using one type, but it later accesses that resource using a type that is incompatible with the original type.

## Extended Description

When the product accesses the resource using an incompatible type, this could trigger logical errors because the resource does not have expected properties. In languages without memory safety, such as C and C++, type confusion can lead to out-of-bounds memory access. While this weakness is frequently associated with unions when parsing data with many different embedded object types in C, it can be present in any application that can interpret the same variable or memory location in multiple ways. This weakness is not unique to C and C++. For example, errors in PHP applications can be triggered by providing array parameters when scalars are expected, or vice versa. Languages such as Perl, which perform automatic conversion of a variable of one type when it is accessed as if it were another type, can also contain these issues.

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## Observed Examples (CVEs)

**•** CVE-2010-4577: Type confusion in CSS sequence leads to out-of-bounds read.

**•** CVE-2011-0611: Size inconsistency allows code execution, first discovered when it was actively exploited in-the-wild. (KEV)

**•** CVE-2010-0258: Improperly-parsed file containing records of different types leads to code execution when a memory location is interpreted as a different object than intended.

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Read Memory, Modify Memory, Execute Unauthorized Code or Commands, DoS: Crash, Exit, or Restart — Notes: When a memory buffer is accessed using the wrong type, it could read or write memory out of the bounds of the buffer, if the allocated buffer is smaller than the type that the code is attempting to access, leading to a crash and possibly code execution.

## Applicable Platforms

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

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

## Demonstrative Examples

**•** The code intends to process the message as a NAME\_TYPE, and sets the default message to "Hello World." However, since both buf.name and buf.nameID are part of the same union, they can act as aliases for the same memory location, depending on memory layout after compilation.

**•** When called with the following query string:

**•** In this case, the programmer intended to use "$UserPrivilegeArray->{$userID}" to access the proper position in the array. But because the subscript was omitted, the "user" string was compared to the scalar representation of the $UserPrivilegeArray reference, which might be of the form "ARRAY(0x229e8)" or similar.

## Notes

**•** Applicable Platform: This weakness is possible in any type-unsafe programming language.

**•** Research Gap: Type confusion weaknesses have received some attention by applied researchers and major software vendors for C and C++ code. Some publicly-reported vulnerabilities probably have type confusion as a root-cause weakness, but these may be described as "memory corruption" instead. For other languages, there are very few public reports of type confusion weaknesses. These are probably under-studied. Since many programs rely directly or indirectly on loose typing, a potential "type confusion" behavior might be intentional, possibly requiring more manual analysis.