# CWE Detail – CWE-822

## Description

The product obtains a value from an untrusted source, converts this value to a pointer, and dereferences the resulting pointer.

## Extended Description

An attacker can supply a pointer for memory locations that the product is not expecting. If the pointer is dereferenced for a write operation, the attack might allow modification of critical state variables, cause a crash, or execute code. If the dereferencing operation is for a read, then the attack might allow reading of sensitive data, cause a crash, or set a variable to an unexpected value (since the value will be read from an unexpected memory location). There are several variants of this weakness, including but not necessarily limited to: The untrusted value is directly invoked as a function call. In OS kernels or drivers where there is a boundary between "userland" and privileged memory spaces, an untrusted pointer might enter through an API or system call (see CWE-781 for one such example). Inadvertently accepting the value from an untrusted control sphere when it did not have to be accepted as input at all. This might occur when the code was originally developed to be run by a single user in a non-networked environment, and the code is then ported to or otherwise exposed to a networked environment.

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## Observed Examples (CVEs)

**•** CVE-2007-5655: message-passing framework interprets values in packets as pointers, causing a crash.

**•** CVE-2010-2299: labeled as a "type confusion" issue, also referred to as a "stale pointer." However, the bug ID says "contents are simply interpreted as a pointer... renderer ordinarily doesn't supply this pointer directly". The "handle" in the untrusted area is replaced in one function, but not another - thus also, effectively, exposure to wrong sphere (CWE-668).

**•** CVE-2009-1719: Untrusted dereference using undocumented constructor.

**•** CVE-2009-1250: An error code is incorrectly checked and interpreted as a pointer, leading to a crash.

**•** CVE-2009-0311: An untrusted value is obtained from a packet and directly called as a function pointer, leading to code execution.

**•** CVE-2010-1818: Undocumented attribute in multimedia software allows "unmarshaling" of an untrusted pointer.

**•** CVE-2010-3189: ActiveX control for security software accepts a parameter that is assumed to be an initialized pointer.

**•** CVE-2010-1253: Spreadsheet software treats certain record values that lead to "user-controlled pointer" (might be untrusted offset, not untrusted pointer).

## Related Attack Patterns (CAPEC)

* CAPEC-129

## Common Consequences

**•** Impact: Read Memory — Notes: If the untrusted pointer is used in a read operation, an attacker might be able to read sensitive portions of memory.

**•** Impact: DoS: Crash, Exit, or Restart — Notes: If the untrusted pointer references a memory location that is not accessible to the product, or points to a location that is "malformed" or larger than expected by a read or write operation, the application may terminate unexpectedly.

**•** Impact: Execute Unauthorized Code or Commands, Modify Memory — Notes: If the untrusted pointer is used in a function call, or points to unexpected data in a write operation, then code execution may be possible.

## Notes

**•** Maintenance: There are close relationships between incorrect pointer dereferences and other weaknesses related to buffer operations. There may not be sufficient community agreement regarding these relationships. Further study is needed to determine when these relationships are chains, composites, perspective/layering, or other types of relationships. As of September 2010, most of the relationships are being captured as chains.

**•** Terminology: Many weaknesses related to pointer dereferences fall under the general term of "memory corruption" or "memory safety." As of September 2010, there is no commonly-used terminology that covers the lower-level variants.