# CWE Detail – CWE-908

## Description

The product uses or accesses a resource that has not been initialized.

## Extended Description

When a resource has not been properly initialized, the product may behave unexpectedly. This may lead to a crash or invalid memory access, but the consequences vary depending on the type of resource and how it is used within the product.

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## Observed Examples (CVEs)

**•** CVE-2019-9805: Chain: Creation of the packet client occurs before initialization is complete (CWE-696) resulting in a read from uninitialized memory (CWE-908), causing memory corruption.

**•** CVE-2008-4197: Use of uninitialized memory may allow code execution.

**•** CVE-2008-2934: Free of an uninitialized pointer leads to crash and possible code execution.

**•** CVE-2008-0063: Product does not clear memory contents when generating an error message, leading to information leak.

**•** CVE-2008-0062: Lack of initialization triggers NULL pointer dereference or double-free.

**•** CVE-2008-0081: Uninitialized variable leads to code execution in popular desktop application.

**•** CVE-2008-3688: Chain: Uninitialized variable leads to infinite loop.

**•** CVE-2008-3475: Chain: Improper initialization leads to memory corruption.

**•** CVE-2005-1036: Chain: Bypass of access restrictions due to improper authorization (CWE-862) of a user results from an improperly initialized (CWE-909) I/O permission bitmap

**•** CVE-2008-3597: Chain: game server can access player data structures before initialization has happened leading to NULL dereference

**•** CVE-2009-2692: Chain: uninitialized function pointers can be dereferenced allowing code execution

**•** CVE-2009-0949: Chain: improper initialization of memory can lead to NULL dereference

**•** CVE-2009-3620: Chain: some unprivileged ioctls do not verify that a structure has been initialized before invocation, leading to NULL dereference

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Read Memory, Read Application Data — Notes: When reusing a resource such as memory or a program variable, the original contents of that resource may not be cleared before it is sent to an untrusted party.

**•** Impact: DoS: Crash, Exit, or Restart — Notes: The uninitialized resource may contain values that cause program flow to change in ways that the programmer did not intend.

## Potential Mitigations

**•** Implementation: Explicitly initialize the resource before use. If this is performed through an API function or standard procedure, follow all required steps. (Effectiveness: N/A)

**•** Implementation: Pay close attention to complex conditionals that affect initialization, since some branches might not perform the initialization. (Effectiveness: N/A)

**•** Implementation: Avoid race conditions (CWE-362) during initialization routines. (Effectiveness: N/A)

**•** Build and Compilation: Run or compile the product with settings that generate warnings about uninitialized variables or data. (Effectiveness: N/A)

## Applicable Platforms

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

## Demonstrative Examples

**•** N/A

**•** If the application is unable to extract the state information - say, due to a database timeout - then the $uid variable will not be explicitly set by the programmer. This will cause $uid to be regarded as equivalent to "0" in the conditional, allowing the original user to perform administrator actions. Even if the attacker cannot directly influence the state data, unexpected errors could cause incorrect privileges to be assigned to a user just by accident.

**•** This might seem innocent enough, but str was not initialized, so it contains random memory. As a result, str[0] might not contain the null terminator, so the copy might start at an offset other than 0. The consequences can vary, depending on the underlying memory.

**•** 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: