# CWE Detail – CWE-14

## Description

Sensitive memory is cleared according to the source code, but compiler optimizations leave the memory untouched when it is not read from again, aka "dead store removal."

## Extended Description

This compiler optimization error occurs when: Secret data are stored in memory. The secret data are scrubbed from memory by overwriting its contents. The source code is compiled using an optimizing compiler, which identifies and removes the function that overwrites the contents as a dead store because the memory is not used subsequently.

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## Modes of Introduction

**•** Implementation: N/A

**•** Build and Compilation: N/A

## Common Consequences

**•** Impact: Read Memory, Bypass Protection Mechanism — Notes: This weakness will allow data that has not been cleared from memory to be read. If this data contains sensitive password information, then an attacker can read the password and use the information to bypass protection mechanisms.

## Potential Mitigations

**•** Implementation: Store the sensitive data in a "volatile" memory location if available. (Effectiveness: N/A)

**•** Build and Compilation: If possible, configure your compiler so that it does not remove dead stores. (Effectiveness: N/A)

**•** Architecture and Design: Where possible, encrypt sensitive data that are used by a software system. (Effectiveness: N/A)

## Applicable Platforms

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

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

## Demonstrative Examples

**•** The code in the example will behave correctly if it is executed verbatim, but if the code is compiled using an optimizing compiler, such as Microsoft Visual C++ .NET or GCC 3.x, then the call to memset() will be removed as a dead store because the buffer pwd is not used after its value is overwritten [18]. Because the buffer pwd contains a sensitive value, the application may be vulnerable to attack if the data are left memory resident. If attackers are able to access the correct region of memory, they may use the recovered password to gain control of the system.