# CWE Detail – CWE-226

## Description

The product releases a resource such as memory or a file so that it can be made available for reuse, but it does not clear or "zeroize" the information contained in the resource before the product performs a critical state transition or makes the resource available for reuse by other entities.

## Extended Description

When resources are released, they can be made available for reuse. For example, after memory is de-allocated, an operating system may make the memory available to another process, or disk space may be reallocated when a file is deleted. As removing information requires time and additional resources, operating systems do not usually clear the previously written information. Even when the resource is reused by the same process, this weakness can arise when new data is not as large as the old data, which leaves portions of the old data still available. Equivalent errors can occur in other situations where the length of data is variable but the associated data structure is not. If memory is not cleared after use, the information may be read by less trustworthy parties when the memory is reallocated. This weakness can apply in hardware, such as when a device or system switches between power, sleep, or debug states during normal operation, or when execution changes to different users or privilege levels.

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## Observed Examples (CVEs)

**•** CVE-2019-3733: Cryptography library does not clear heap memory before release

**•** CVE-2003-0001: Ethernet NIC drivers do not pad frames with null bytes, leading to infoleak from malformed packets.

**•** CVE-2003-0291: router does not clear information from DHCP packets that have been previously used

**•** CVE-2005-1406: Products do not fully clear memory buffers when less data is stored into the buffer than previous.

**•** CVE-2005-1858: Products do not fully clear memory buffers when less data is stored into the buffer than previous.

**•** CVE-2005-3180: Products do not fully clear memory buffers when less data is stored into the buffer than previous.

**•** CVE-2005-3276: Product does not clear a data structure before writing to part of it, yielding information leak of previously used memory.

**•** CVE-2002-2077: Memory not properly cleared before reuse.

## Related Attack Patterns (CAPEC)

* CAPEC-37

## Attack TTPs

**•** T1005: Data from Local System (Tactics: collection)

**•** T1552.004: Private Keys (Tactics: credential-access)

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Read Application Data — Notes:

## Potential Mitigations

**•** Architecture and Design: During critical state transitions, information not needed in the next state should be removed or overwritten with fixed patterns (such as all 0's) or random data, before the transition to the next state. (Effectiveness: High)

**•** Architecture and Design: When releasing, de-allocating, or deleting a resource, overwrite its data and relevant metadata with fixed patterns or random data. Be cautious about complex resource types whose underlying representation might be non-contiguous or change at a low level, such as how a file might be split into different chunks on a file system, even though "logical" file positions are contiguous at the application layer. Such resource types might require invocation of special modes or APIs to tell the underlying operating system to perform the necessary clearing, such as SDelete (Secure Delete) on Windows, although the appropriate functionality might not be available at the application layer. (Effectiveness: High)

## Applicable Platforms

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

## Demonstrative Examples

**•** Suppose a device is transitioning from state A to state B. During state A, it can read certain private keys from the hidden fuses that are only accessible in state A but not in state B. The device reads the keys, performs operations using those keys, then transitions to state B, where those private keys should no longer be accessible.

**•** There is an attempt to scrub the sensitive data from memory, but realloc() is used, so it could return a pointer to a different part of memory. The memory that was originally allocated for cleartext\_buffer could still contain an uncleared copy of the data.

**•** The above code snippet [REF-1402] illustrates an instance of a vulnerable implementation of the AES
 wrapper module, where p\_c[i] registers are cleared at reset. Otherwise, p\_c[i]registers either
 maintain their old values (if reglk\_ctrl\_i[3]is true) or get filled through the AXI signal wdata. Note
 that p\_c[i]registers can be read through the AXI Lite interface (not shown in snippet). However,
 p\_c[i] registers are never cleared after their usage once the AES engine has completed the encryption
 process of the message. In a multi-user or multi-process environment, not clearing registers may result
 in the attacker process accessing data left by the victim, leading to data leakage or unintentional
 information disclosure. To fix this issue, it is essential to ensure that these internal registers are cleared in a timely manner after
 their usage, i.e., the encryption process is complete. This is illustrated below by monitoring the assertion
 of the cipher text valid signal, ct\_valid [REF-1403].

## Notes

**•** Relationship: There is a close association between CWE-226 and CWE-212. The difference is partially that of perspective. CWE-226 is geared towards the final stage of the resource lifecycle, in which the resource is deleted, eliminated, expired, or otherwise released for reuse. Technically, this involves a transfer to a different control sphere, in which the original contents of the resource are no longer relevant. CWE-212, however, is intended for sensitive data in resources that are intentionally shared with others, so they are still active. This distinction is useful from the perspective of the CWE research view (CWE-1000).

**•** Maintenance: This entry needs modification to clarify the differences with CWE-212. The description also combines two problems that are distinct from the CWE research perspective: the inadvertent transfer of information to another sphere, and improper initialization/shutdown. Some of the associated taxonomy mappings reflect these different uses.

**•** Research Gap: This is frequently found for network packets, but it can also exist in local memory allocation, files, etc.