# CWE Detail – CWE-1325

## Description

The product manages a group of objects or resources and performs a separate memory allocation for each object, but it does not properly limit the total amount of memory that is consumed by all of the combined objects.

## Extended Description

While the product might limit the amount of memory that is allocated in a single operation for a single object (such as a malloc of an array), if an attacker can cause multiple objects to be allocated in separate operations, then this might cause higher total memory consumption than the developer intended, leading to a denial of service.

## Threat-Mapped Scoring

Score: 1.9

Priority: P3 - Important (Medium)

## Observed Examples (CVEs)

**•** CVE-2020-36049: JavaScript-based packet decoder uses concatenation of many small strings, causing out-of-memory (OOM) condition

**•** CVE-2019-20176: Product allocates a new buffer on the stack for each file in a directory, allowing stack exhaustion

**•** CVE-2013-1591: Chain: an integer overflow (CWE-190) in the image size calculation causes an infinite loop (CWE-835) which sequentially allocates buffers without limits (CWE-1325) until the stack is full.

## Related Attack Patterns (CAPEC)

* CAPEC-130

## Attack TTPs

**•** T1499.003: Application Exhaustion Flood (Tactics: impact)

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: DoS: Resource Consumption (Memory) — Notes: Not controlling memory allocation can result in a request for too much system memory, possibly leading to a crash of the application due to out-of-memory conditions, or the consumption of a large amount of memory on the system.

## Potential Mitigations

**•** Implementation: Ensure multiple allocations of the same kind of object are properly tracked - possibly across multiple sessions, requests, or messages. Define an appropriate strategy for handling requests that exceed the limit, and consider supporting a configuration option so that the administrator can extend the amount of memory to be used if necessary. (Effectiveness: N/A)

**•** Operation: Run the program using system-provided resource limits for memory. This might still cause the program to crash or exit, but the impact to the rest of the system will be minimized. (Effectiveness: N/A)

## Applicable Platforms

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

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

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

## Demonstrative Examples

**•** Since this uses alloca(), it allocates memory directly on the stack. If end\_limit is large enough, then the stack can be entirely consumed.