# CWE Detail – CWE-1341

## Description

The product attempts to close or release a resource or handle more than once, without any successful open between the close operations.

## Extended Description

Code typically requires "opening" handles or references to resources such as memory, files, devices, socket connections, services, etc. When the code is finished with using the resource, it is typically expected to "close" or "release" the resource, which indicates to the environment (such as the OS) that the resource can be re-assigned or reused by unrelated processes or actors - or in some cases, within the same process. API functions or other abstractions are often used to perform this release, such as free() or delete() within C/C++, or file-handle close() operations that are used in many languages. Unfortunately, the implementation or design of such APIs might expect the developer to be responsible for ensuring that such APIs are only called once per release of the resource. If the developer attempts to release the same resource/handle more than once, then the API's expectations are not met, resulting in undefined and/or insecure behavior. This could lead to consequences such as memory corruption, data corruption, execution path corruption, or other consequences. Note that while the implementation for most (if not all) resource reservation allocations involve a unique identifier/pointer/symbolic reference, then if this identifier is reused, checking the identifier for resource closure may result in a false state of openness and closing of the wrong resource. For this reason, reuse of identifiers is discouraged.

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## Observed Examples (CVEs)

**•** CVE-2019-13351: file descriptor double close can cause the wrong file to be associated with a file descriptor.

**•** CVE-2006-5051: Chain: Signal handler contains too much functionality (CWE-828), introducing a race condition that leads to a double free (CWE-415).

**•** CVE-2004-0772: Double free resultant from certain error conditions.

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: DoS: Crash, Exit, or Restart — Notes:

## Potential Mitigations

**•** Implementation: Change the code's logic so that the resource is only closed once. This might require simplifying or refactoring. This fix can be simple to do in small code blocks, but more difficult when multiple closes are buried within complex conditionals. (Effectiveness: N/A)

**•** Implementation: It can be effective to implement a flag that is (1) set when the resource is opened, (2) cleared when it is closed, and (3) checked before closing. This approach can be useful when there are disparate cases in which closes must be performed. However, flag-tracking can increase code complexity and requires diligent compliance by the programmer. (Effectiveness: N/A)

**•** Implementation: When closing a resource, set the resource's associated variable to NULL or equivalent value for the given language. Some APIs will ignore this null value without causing errors. For other APIs, this can lead to application crashes or exceptions, which may still be preferable to corrupting an unintended resource such as memory or data. (Effectiveness: Defense in Depth)

## Applicable Platforms

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

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

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

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

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

## Demonstrative Examples

**•** There are multiple possible fixes. This fix only has one call to fclose(), which is typically the preferred handling of this problem - but this simplistic method is not always possible.

**•** Double free vulnerabilities have two common (and sometimes overlapping) causes:

## Notes

**•** Terminology: The terms related to "release" may vary depending on the type of resource, programming language, specification, or framework. "Close" has been used synonymously for the release of resources like file descriptors and file handles. "Return" is sometimes used instead of Release. "Free" is typically used when releasing memory or buffers back into the system for reuse.