# CWE Detail – CWE-366

## Description

If two threads of execution use a resource simultaneously, there exists the possibility that resources may be used while invalid, in turn making the state of execution undefined.

## Extended Description

N/A

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## Observed Examples (CVEs)

**•** CVE-2022-2621: Chain: two threads in a web browser use the same resource (CWE-366), but one of those threads can destroy the resource before the other has completed (CWE-416).

## Related Attack Patterns (CAPEC)

* CAPEC-26
* CAPEC-29

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Alter Execution Logic, Unexpected State — Notes: The main problem is that -- if a lock is overcome -- data could be altered in a bad state.

## Potential Mitigations

**•** Architecture and Design: Use locking functionality. This is the recommended solution. Implement some form of locking mechanism around code which alters or reads persistent data in a multithreaded environment. (Effectiveness: N/A)

**•** Architecture and Design: Create resource-locking validation checks. If no inherent locking mechanisms exist, use flags and signals to enforce your own blocking scheme when resources are being used by other threads of execution. (Effectiveness: N/A)

## Applicable Platforms

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

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

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

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

## Demonstrative Examples

**•** N/A