# CWE Detail – CWE-765

## Description

The product unlocks a critical resource more times than intended, leading to an unexpected state in the system.

## Extended Description

When the product is operating in a concurrent environment and repeatedly unlocks a critical resource, the consequences will vary based on the type of lock, the lock's implementation, and the resource being protected. In some situations such as with semaphores, the resources are pooled and extra calls to unlock will increase the count for the number of available resources, likely resulting in a crash or unpredictable behavior when the system nears capacity.

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## Observed Examples (CVEs)

**•** CVE-2009-0935: Attacker provides invalid address to a memory-reading function, causing a mutex to be unlocked twice

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: DoS: Crash, Exit, or Restart, Modify Memory, Unexpected State — Notes:

## Potential Mitigations

**•** Implementation: When locking and unlocking a resource, try to be sure that all control paths through the code in which the resource is locked one or more times correspond to exactly as many unlocks. If the product acquires a lock and then determines it is not able to perform its intended behavior, be sure to release the lock(s) before waiting for conditions to improve. Reacquire the lock(s) before trying again. (Effectiveness: N/A)

## Notes

**•** Maintenance: An alternate way to think about this weakness is as an imbalance between the number of locks / unlocks in the control flow. Over the course of execution, if each lock call is not followed by a subsequent call to unlock in a reasonable amount of time, then system performance may be degraded or at least operating at less than peak levels if there is competition for the locks. This entry may need to be modified to reflect these concepts in the future.