# CWE Detail – CWE-609

## Description

The product uses double-checked locking to access a resource without the overhead of explicit synchronization, but the locking is insufficient.

## Extended Description

Double-checked locking refers to the situation where a programmer checks to see if a resource has been initialized, grabs a lock, checks again to see if the resource has been initialized, and then performs the initialization if it has not occurred yet. This should not be done, as it is not guaranteed to work in all languages and on all architectures. In summary, other threads may not be operating inside the synchronous block and are not guaranteed to see the operations execute in the same order as they would appear inside the synchronous block.

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Modify Application Data, Alter Execution Logic — Notes:

## Potential Mitigations

**•** Implementation: While double-checked locking can be achieved in some languages, it is inherently flawed in Java before 1.5, and cannot be achieved without compromising platform independence. Before Java 1.5, only use of the synchronized keyword is known to work. Beginning in Java 1.5, use of the "volatile" keyword allows double-checked locking to work successfully, although there is some debate as to whether it achieves sufficient performance gains. See references. (Effectiveness: N/A)

## Applicable Platforms

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

## Demonstrative Examples

**•** The programmer wants to guarantee that only one Helper() object is ever allocated, but does not want to pay the cost of synchronization every time this code is called.