# CWE Detail – CWE-567

## Description

The product does not properly synchronize shared data, such as static variables across threads, which can lead to undefined behavior and unpredictable data changes.

## Extended Description

Within servlets, shared static variables are not protected from concurrent access, but servlets are multithreaded. This is a typical programming mistake in J2EE applications, since the multithreading is handled by the framework. When a shared variable can be influenced by an attacker, one thread could wind up modifying the variable to contain data that is not valid for a different thread that is also using the data within the variable. Note that this weakness is not unique to servlets.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Related Attack Patterns (CAPEC)

* CAPEC-25

## Attack TTPs

**•** T1499.004: Application or System Exploitation (Tactics: impact)

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Read Application Data, Modify Application Data, DoS: Instability, DoS: Crash, Exit, or Restart — Notes: If the shared variable contains sensitive data, it may be manipulated or displayed in another user session. If this data is used to control the application, its value can be manipulated to cause the application to crash or perform poorly.

## Potential Mitigations

**•** Implementation: Remove the use of static variables used between servlets. If this cannot be avoided, use synchronized access for these variables. (Effectiveness: N/A)

## Applicable Platforms

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

## Demonstrative Examples

**•** Consider when two separate threads, Thread A and Thread B, concurrently handle two different requests: