# CWE Detail – CWE-488

## Description

The product does not sufficiently enforce boundaries between the states of different sessions, causing data to be provided to, or used by, the wrong session.

## Extended Description

Data can "bleed" from one session to another through member variables of singleton objects, such as Servlets, and objects from a shared pool. In the case of Servlets, developers sometimes do not understand that, unless a Servlet implements the SingleThreadModel interface, the Servlet is a singleton; there is only one instance of the Servlet, and that single instance is used and re-used to handle multiple requests that are processed simultaneously by different threads. A common result is that developers use Servlet member fields in such a way that one user may inadvertently see another user's data. In other words, storing user data in Servlet member fields introduces a data access race condition.

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## Related Attack Patterns (CAPEC)

* CAPEC-59
* CAPEC-60

## Attack TTPs

**•** T1134.001: Token Impersonation/Theft (Tactics: defense-evasion, privilege-escalation)

**•** T1550.004: Web Session Cookie (Tactics: defense-evasion, lateral-movement)

## Modes of Introduction

**•** Implementation: REALIZATION: This weakness is caused during implementation of an architectural security tactic.

## Common Consequences

**•** Impact: Read Application Data — Notes:

## Potential Mitigations

**•** Architecture and Design: Protect the application's sessions from information leakage. Make sure that a session's data is not used or visible by other sessions. (Effectiveness: N/A)

**•** Testing: Use a static analysis tool to scan the code for information leakage vulnerabilities (e.g. Singleton Member Field). (Effectiveness: N/A)

**•** Architecture and Design: In a multithreading environment, storing user data in Servlet member fields introduces a data access race condition. Do not use member fields to store information in the Servlet. (Effectiveness: N/A)

## Applicable Platforms

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

## Demonstrative Examples

**•** While this code will work perfectly in a single-user environment, if two users access the Servlet at approximately the same time, it is possible for the two request handler threads to interleave in the following way: Thread 1: assign "Dick" to name Thread 2: assign "Jane" to name Thread 1: print "Jane, thanks for visiting!" Thread 2: print "Jane, thanks for visiting!" Thereby showing the first user the second user's name.