# CWE Detail – CWE-384

## Description

Authenticating a user, or otherwise establishing a new user session, without invalidating any existing session identifier gives an attacker the opportunity to steal authenticated sessions.

## Extended Description

Such a scenario is commonly observed when: A web application authenticates a user without first invalidating the existing session, thereby continuing to use the session already associated with the user. An attacker is able to force a known session identifier on a user so that, once the user authenticates, the attacker has access to the authenticated session. The application or container uses predictable session identifiers. In the generic exploit of session fixation vulnerabilities, an attacker creates a new session on a web application and records the associated session identifier. The attacker then causes the victim to associate, and possibly authenticate, against the server using that session identifier, giving the attacker access to the user's account through the active session.

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## Observed Examples (CVEs)

**•** CVE-2022-2820: Website software for game servers does not proprerly terminate user sessions, allowing for possible session fixation

## Related Attack Patterns (CAPEC)

* CAPEC-196
* CAPEC-21
* CAPEC-31
* CAPEC-39
* CAPEC-59
* CAPEC-60
* CAPEC-61

## Attack TTPs

**•** T1539: Steal Web Session Cookie (Tactics: credential-access)

**•** T1134.002: Create Process with Token (Tactics: defense-evasion, privilege-escalation)

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

**•** T1528: Steal Application Access Token (Tactics: credential-access)

**•** T1134.003: Make and Impersonate Token (Tactics: defense-evasion, privilege-escalation)

**•** T1606: Forge Web Credentials (Tactics: credential-access)

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

**•** T1134: Access Token Manipulation (Tactics: defense-evasion, privilege-escalation)

## Modes of Introduction

**•** Architecture and Design: N/A

**•** Implementation: N/A

## Common Consequences

**•** Impact: Gain Privileges or Assume Identity — Notes:

## Potential Mitigations

**•** Architecture and Design: Invalidate any existing session identifiers prior to authorizing a new user session. (Effectiveness: N/A)

**•** Architecture and Design: For platforms such as ASP that do not generate new values for sessionid cookies, utilize a secondary cookie. In this approach, set a secondary cookie on the user's browser to a random value and set a session variable to the same value. If the session variable and the cookie value ever don't match, invalidate the session, and force the user to log on again. (Effectiveness: N/A)

## Applicable Platforms

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

## Demonstrative Examples

**•** In order to exploit the code above, an attacker could first create a session (perhaps by logging into the application) from a public terminal, record the session identifier assigned by the application, and reset the browser to the login page. Next, a victim sits down at the same public terminal, notices the browser open to the login page of the site, and enters credentials to authenticate against the application. The code responsible for authenticating the victim continues to use the pre-existing session identifier, now the attacker simply uses the session identifier recorded earlier to access the victim's active session, providing nearly unrestricted access to the victim's account for the lifetime of the session. Even given a vulnerable application, the success of the specific attack described here is dependent on several factors working in the favor of the attacker: access to an unmonitored public terminal, the ability to keep the compromised session active and a victim interested in logging into the vulnerable application on the public terminal.

**•** N/A

## Notes

**•** Other: Other attack vectors include DNS poisoning and related network based attacks where an attacker causes the user to visit a malicious site by redirecting a request for a valid site. Network based attacks typically involve a physical presence on the victim's network or control of a compromised machine on the network, which makes them harder to exploit remotely, but their significance should not be overlooked. Less secure session management mechanisms, such as the default implementation in Apache Tomcat, allow session identifiers normally expected in a cookie to be specified on the URL as well, which enables an attacker to cause a victim to use a fixed session identifier simply by emailing a malicious URL.