# Session Management Testing

## 1. Introduction

Session management testing focuses on the security of the mechanisms used to manage user sessions, including the generation, maintenance, and destruction of session tokens. A secure session management implementation is critical to preventing attackers from hijacking user sessions and gaining unauthorized access to the application. This testing category examines cookie attributes, session fixation, session timeout, and other session-related vulnerabilities.

## 2. CVSS Score Summary

| Test Case | Vulnerability | CVSS v3.1 Score | CVSS Vector | Severity |
|---|---|---|---|---|
| OTG-SESS-001 | Testing for Bypassing Session Management Schema | 0.0 | N/A | Informational |
| OTG-SESS-002 | Insecure Cookie Attributes | 5.3 | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N | Medium |
| OTG-SESS-003 | Testing for Session Fixation | 0.0 | N/A | Informational |
| OTG-SESS-004 | Testing for Exposed Session Variables | 0.0 | N/A | Informational |
| OTG-SESS-005 | Testing for Cross Site Request | 0.0 | N/A | Informational |

| | | | | |
|---|---|---|---|---|
| | Forgery (CSRF) | | | |
| OTG-SESS-006 | Testing for logout functionality | 0.0 | N/A | Informational |
| OTG-SESS-007 | Test Session Timeout | 0.0 | N/A | Informational |
| OTG-SESS-008 | Testing for Session puzzling | 0.0 | N/A | Informational |

## 3. Summary of Findings

The most significant finding in this category is the use of insecure cookie attributes. The session cookies (`PHPSESSID` and `security`) are missing the `Secure` and `HttpOnly` flags, and have a weak `SameSite` policy. This makes the application more vulnerable to session hijacking attacks, such as Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF).

A key challenge during this assessment was the inability to test for session fixation and other session-related vulnerabilities due to the lack of a valid session cookie before login. While the tests for bypassing the session management schema, exposed session variables, CSRF, logout functionality, and session puzzling did not reveal any vulnerabilities, the inability to fully test for session fixation leaves a potential gap in the assessment of the application's session management security.

## 4. General Recommendation/Remediation

- **Secure Cookie Attributes:** The `Secure` and `HttpOnly` flags should be set for all session cookies. The `SameSite` attribute should be set to `Strict` or `Lax` to prevent CSRF attacks.
- **Regenerate Session IDs:** Session IDs should be regenerated after any privilege level change, such as a user logging in.
- **Implement Session Timeouts:** The application should enforce a reasonable session timeout to prevent session hijacking.
- **Do Not Expose Session Variables:** Session variables should not be exposed in the client-side code.