**Session Management (https://pages.nist.gov/800-63-3/sp800-63b.html#sec7).**

Once an authentication event has taken place, it is often desirable to allow the subscriber to continue using the application across multiple subsequent interactions without requiring them to repeat the authentication event. This requirement is particularly true for federation scenarios — described in [SP 800-63C](https://pages.nist.gov/800-63-3/sp800-63c.html) — where the authentication event necessarily involves several components and parties coordinating across a network.

To facilitate this behavior, a session MAY be started in response to an authentication event, and continue the session until such time that it is terminated. The session MAY be terminated for any number of reasons, including but not limited to an inactivity timeout, an explicit logout event, or other means. The session MAY be continued through a re authentication event — described in [Section 7.2](https://pages.nist.gov/800-63-3/sp800-63b.html#sessionreauthn) — wherein the user repeats some or all of the initial authentication event, thereby re-establishing the session.

Session management is preferable over continual presentation of credentials as the poor usability of continual presentation often creates incentives for workarounds such as cached unlocking credentials, negating the freshness of the authentication event.

7.1 Session Bindings

A session occurs between the software that a subscriber is running — such as a browser, application, or operating system (i.e., the session subject) — and the RP or CSP that the subscriber is accessing (i.e., the session host). A session secret SHALL be shared between the subscriber’s software and the service being accessed. This secret binds the two ends of the session, allowing the subscriber to continue using the service over time. The secret SHALL be presented directly by the subscriber’s software or possession of the secret SHALL be proven using a cryptographic mechanism.

The secret used for session binding SHALL be generated by the session host in direct response to an authentication event. A session SHOULD inherit the AAL properties of the authentication event which triggered its creation. A session MAY be considered at a lower AAL than the authentication event but SHALL NOT be considered at a higher AAL than the authentication event.

Secrets used for session binding:

1. SHALL be generated by the session host during an interaction, typically immediately following authentication.
2. SHALL be generated by an approved random bit generator [[SP 800-90Ar1]](https://pages.nist.gov/800-63-3/sp800-63b.html#SP800-90Ar1) and contain at least 64 bits of entropy.
3. SHALL be erased or invalidated by the session subject when the subscriber logs out.
4. SHOULD be erased on the subscriber endpoint when the user logs out or when the secret is deemed to have expired.
5. SHOULD NOT be placed in insecure locations such as HTML5 Local Storage due to the potential exposure of local storage to cross-site scripting (XSS) attacks.
6. SHALL be sent to and received from the device using an authenticated protected channel.
7. SHALL time out and not be accepted after the times specified in [Sections 4.1.4](https://pages.nist.gov/800-63-3/sp800-63b.html#aal1reauth), [4.2.4](https://pages.nist.gov/800-63-3/sp800-63b.html#aal2reauth), and [4.3.4](https://pages.nist.gov/800-63-3/sp800-63b.html#aal3reauth), as appropriate for the AAL.
8. SHALL NOT be available to insecure communications between the host and subscriber’s endpoint. Authenticated sessions SHALL NOT fall back to an insecure transport, such as from https to http, following authentication.

URLs or POST content SHALL contain a session identifier that SHALL be verified by the RP to ensure that actions taken outside the session do not affect the protected session.

There are several mechanisms for managing a session over time. The following sections give different examples along with additional requirements and considerations particular to each example technology. Additional informative guidance is available in the OWASP *Session Management Cheat Sheet* [[OWASP-session]](https://pages.nist.gov/800-63-3/sp800-63b.html#OWASP-session).

7.1.1 Browser Cookies

Browser cookies are the predominant mechanism by which a session will be created and tracked for a subscriber accessing a service.

Cookies:

1. SHALL be tagged to be accessible only on secure (HTTPS) sessions.
2. SHALL be accessible to the minimum practical set of hostnames and paths.
3. SHOULD be tagged to be inaccessible via JavaScript (HttpOnly).
4. SHOULD be tagged to expire at, or soon after, the session’s validity period. This requirement is intended to limit the accumulation of cookies, but SHALL NOT be depended upon to enforce session timeouts.

7.1.2 Access Tokens

An access token — such as found in OAuth — is used to allow an application to access a set of services on a subscriber’s behalf following an authentication event. The presence of an OAuth access token SHALL NOT be interpreted by the RP as presence of the subscriber, in the absence of other signals. The OAuth access token, and any associated refresh tokens, MAY be valid long after the authentication session has ended and the subscriber has left the application.

7.1.3 Device Identification

Other methods of secure device identification — including but not limited to mutual TLS, token binding, or other mechanisms — MAY be used to enact a session between a subscriber and a service.

7.2 Re authentication

Continuity of authenticated sessions SHALL be based upon the possession of a session secret issued by the verifier at the time of authentication and optionally refreshed during the session. The nature of a session depends on the application, including:

1. A web browser session with a “session” cookie, or
2. An instance of a mobile application that retains a session secret.

Session secrets SHALL be non-persistent. That is, they SHALL NOT be retained across a restart of the associated application or a reboot of the host device.

Periodic re authentication of sessions SHALL be performed to confirm the continued presence of the subscriber at an authenticated session (i.e., that the subscriber has not walked away without logging out).

A session SHALL NOT be extended past the guidelines in Sections [4.1.3](https://pages.nist.gov/800-63-3/sp800-63b.html#aal1reauth), [4.2.3](https://pages.nist.gov/800-63-3/sp800-63b.html#aal2reauth), and [4.3.3](https://pages.nist.gov/800-63-3/sp800-63b.html#aal3reauth) (depending on AAL) based on presentation of the session secret alone. Prior to session expiration, the re authentication time limit SHALL be extended by prompting the subscriber for the authentication factor(s) specified in [Table 7-1](https://pages.nist.gov/800-63-3/sp800-63b.html#63bSec7-Table1).

When a session has been terminated, due to a time-out or other action, the user SHALL be required to establish a new session by authenticating again.

**Table 7-1 AAL Re authentication Requirements**

| **AAL** | **Requirement** |
| --- | --- |
| 1 | Presentation of any one factor |
| 2 | Presentation of a memorized secret or biometric |
| 3 | Presentation of all factors |

Note: At AAL2, a memorized secret or biometric, and not a physical authenticator, is required because the session secret is *something you have*, and an additional authentication factor is required to continue the session.

7.2.1 Re authentication from a Federation or Assertion

When using a federation protocol as described in [SP 800-63C](https://pages.nist.gov/800-63-3/sp800-63c.html), Section 5 to connect the CSP and RP, special considerations apply to session management and re authentication. The federation protocol communicates an authentication event between the CSP and the RP but establishes no session between them. Since the CSP and RP often employ separate session management technologies, there SHALL NOT be any assumption of correlation between these sessions. Consequently, when an RP session expires and the RP requires re authentication, it is entirely possible that the session at the CSP has not expired and that a new assertion could be generated from this session at the CSP without re authenticating the user.

An RP requiring re authentication through a federation protocol SHALL — if possible within the protocol — specify the maximum acceptable authentication age to the CSP, and the CSP SHALL re authenticate the subscriber if they have not been authenticated within that time period. The CSP SHALL communicate the authentication event time to the RP to allow the RP to decide if the assertion is sufficient for re authentication and to determine the time for the next re authentication event.

| **Security Parameter** | **NIST Guidelines Key Points** | **Weight Justification** |
| --- | --- | --- |
| **HttpOnly** (Cookie Protection) | SHOULD be tagged to be inaccessible via JavaScript (HttpOnly). | **Weight: 1.0** (XSS se bachne ke liye critical) |
| **Secure Flag (HTTPS only)** | SHALL be tagged to be accessible only on secure (HTTPS) sessions. | **Weight: 1.0** (Man-in-the-middle attack ka risk) |
| **HSTS (Strict Transport Security)** | SHALL NOT fall back to an insecure transport, such as HTTP after authentication. | **Weight: 0.8** (Force HTTPS zaroori hai) |
| **X-Content-Type-Options** | SHALL prevent MIME-type sniffing attacks. | **Weight: 0.3** (Kam risk but best practice) |
| **X-Frame-Options** (Clickjacking) | SHALL prevent UI redress attacks. | **Weight: 0.4** (Moderate risk) |
| **Session Identifier in URL** | SHALL NOT store session secrets in URLs or insecure storage. | **Weight: 1.0** (Highly risky - Session Hijacking) |
| **Session Timeout Policy** | SHALL be enforced based on authentication levels (AAL1, AAL2, AAL3). | **Weight: 0.9** (Session expiry se hijacking prevent hoti hai) |

OWASP's **HTTP Security Response Headers Cheat Sheet:**

OWASP's **HTTP Security Response Headers Cheat Sheet** provides detailed insights into the security impact of different HTTP headers. By analyzing this document, we can determine the weightage (or severity impact) of missing headers in your dataset based on their security importance. Here’s how we can assign weights:

**Weightage Analysis for Missing Security Headers**

| **Security Header** | **Impact Analysis (Based on OWASP)** | **Recommended Weight (0 - 1.0)** |
| --- | --- | --- |
| **Strict-Transport-Security (HSTS)** | Prevents protocol downgrade attacks & enforces HTTPS, reducing session hijacking risk. | **0.8 - 1.0** |
| **X-Content-Type-Options** | Prevents MIME sniffing attacks, reducing exposure to XSS & drive-by downloads. | **0.3 - 0.4** |
| **X-Frame-Options** | Protects against clickjacking attacks, ensuring site content isn’t embedded in malicious iframes. | **0.4 - 0.5** |
| **X-XSS-Protection** | Deprecated but historically mitigated reflected XSS attacks. Should be replaced with CSP. | **0.2 - 0.3** |
| **Content-Security-Policy (CSP)** | Strongest defense against XSS, preventing malicious scripts from executing. | **0.9 - 1.0** |
| **Referrer-Policy** | Prevents sensitive data from being leaked in referer headers. | **0.4 - 0.5** |
| **Permissions-Policy** | Limits access to sensitive browser features like camera, microphone, etc. | **0.4 - 0.6** |
| **Access-Control-Allow-Origin (CORS)** | Controls which domains can access resources, reducing the risk of cross-origin attacks. | **0.5 - 0.7** |

**Weightage Adjustments for Session Security**

Since your project focuses on **Session Hijacking Detection**, headers directly impacting **session security** should have **higher weightage**:

* **Missing Secure Cookie Flag** → **1.0** (Essential for preventing session hijacking over HTTP)
* **Missing HttpOnly Cookie** → **1.0** (Prevents JavaScript from accessing session tokens)
* **Session in URL** → **1.0** (High risk of session token exposure)
* **Missing SameSite Cookie Attribute** → **0.7 - 0.9** (Reduces CSRF attacks)
* **Missing HSTS** → **0.8 - 1.0** (Forces HTTPS, reducing MITM risk)
* **Session in Referer Header** → **0.7** (Possible token leakage)