**SESSIONS & COOKIES VS. LOCALSTORAGE**

**More On Session Security**

Users and web developers should be concerned with [session hijacking](https://owasp.org/www-community/attacks/Session_hijacking_attack), an attack in which an attacker steals session identifiers and gains access to the web server as a different person.

Why hijack a session? If a user is in an authenticated session on their bank’s website, the attacker could transfer funds from a user’s bank account. Another scenario is an attacker could hijack the session of an authorized admin on an organization’s website and steal data.

Below, we will introduce a couple of secure practices for implementing sessions that prevents hijacking attacks.

**Define Session Expiry**

The shorter a session is, the less time an attacker has to hijack a session. This is usually done by setting an expiry on the session cookie. It’s also important to implement an automatic session expiration on the backend.

A timeout dictates how long a session can stay open. The session timeout after an idle period is a common feature on bank websites! Other environments that require high security even implement an absolute timeout where a user’s session ends regardless of activity.

**Make Session IDs Difficult to Hack**

Session IDs are just like passwords — the longer and more random, the better. According to [OWASP](https://owasp.org/www-community/vulnerabilities/Insufficient_Session-ID_Length), session identifiers should be at least 128 bits long. This helps prevent brute-force attacks where a hacker uses multiple bots to guess IDs.

In order to make the session ID random, ensure it does not contain personally identifying information and that the algorithm to generate an ID doesn’t follow a predefined pattern that makes it easier to guess.

**Securing Cookies**

Session cookies can be made more secure if they expire. This decreases the timeframe where an attacker could steal the session identifier.

You should also use the [Secure](https://owasp.org/www-community/controls/SecureCookieAttribute), [HttpOnly](https://owasp.org/www-community/HttpOnly" \t "_blank), and [SameSite](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie/SameSite" \t "_blank) attributes in the Set-Cookie HTTP header.

**Enforce HTTPS**

Ideally, all sessions-based web applications should enforce HTTPS for all communication! This prevents common web attacks that could give the attacker access to the session.