



AI Mode All Short videos Videos Forums More ▾



should I use csrf cookie header with jwt access and refresh token auth system



Yes, if you store your JWTs in **HTTP-only cookies**, you absolutely need to implement a CSRF protection mechanism, such as using an anti-CSRF token, because the browser will automatically attach cookies to all requests, even those from malicious sites. [🔗](#)

Why CSRF Protection is Needed with Cookie-Stored JWTs

- **Browser Behavior:** Cookies, including HTTP-only ones, are automatically sent by the browser with every request to the domain, regardless of the request's origin (the website the user is currently visiting).
- **CSRF Vulnerability:** A malicious site can trick a user's browser into making an unwanted request to your application (e.g., a form submission to change a password or transfer funds). Since the browser automatically includes the authentication JWT cookie, the server would treat this forged request as legitimate if there were no further

Ask anything



AI Mode All Short videos Videos Forums More ▾



1. Server-Side:

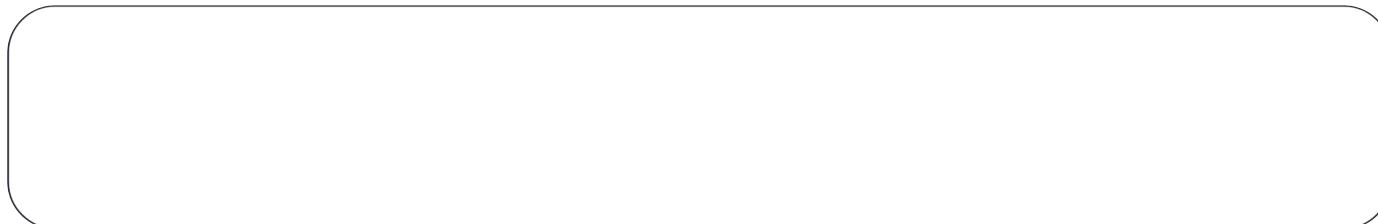
1. Upon login, issue the JWT (or refresh token) in a secure, **HTTP-only** cookie (e.g., `accessToken` or `refreshToken`). This prevents XSS.
2. Also, issue a separate, non-HTTP-only cookie containing a randomly generated CSRF token (e.g., `X-CSRF-Token`). This cookie is accessible by client-side JavaScript.

2. Client-Side:

1. For all state-changing requests (POST, PUT, DELETE), your client-side JavaScript reads the value from the non-HTTP-only CSRF cookie.
2. It then sends this value back to the server in a custom HTTP header (e.g., `X-CSRF-Token`).

3. Server-Side Validation:

1. The server receives the request and performs two checks:
 - It validates the JWT from the HTTP-only cookie to authenticate the user.

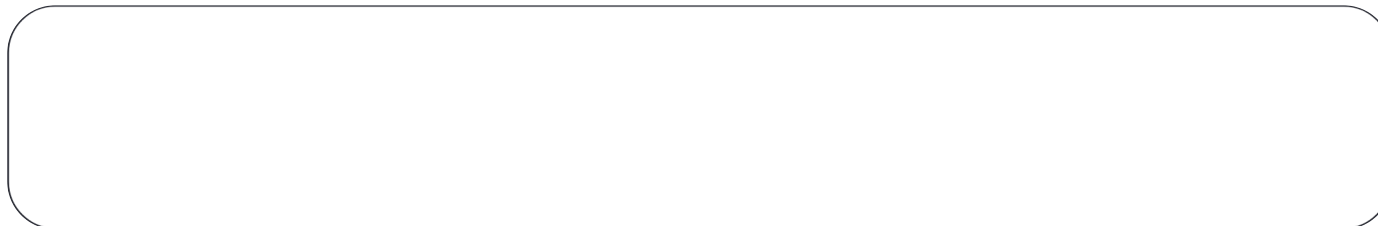


AI Mode All Short videos Videos Forums More ▾



Alternative Mitigation

Modern browsers also offer significant CSRF protection through the `SameSite` cookie attribute, particularly with `SameSite=Strict` or `SameSite=Lax`. Using `SameSite=Strict` is a powerful defense, but the anti-CSRF token mechanism offers defense-in-depth, especially for compatibility with older browsers or complex cross-domain setups. [🔗](#)



AI Mode All Short videos Videos Forums More ▾



 18 sites



JWTs & CSRF Tokens - bv Buddika Gunawardena - Medium

