While short-lived cookies can reduce the "window of opportunity" for an attacker, they are **not a sufficient primary defense** against Cross-Site Request Forgery (CSRF).

Why They Are Not a Complete Measure

- **Automatic Submission:** The fundamental vulnerability of CSRF is that browsers automatically include all relevant cookies for a domain in every request, regardless of whether that request was intentional or malicious. As long as a cookie is valid—even for just a few minutes—an attacker can trigger a request during that window.
- **Attack Timing:** Most CSRF attacks are executed immediately when a user visits a malicious site or clicks a link. If the user's short-lived session is currently active, the attack will succeed.

Better Primary Defenses

Instead of relying on cookie lifespan, modern security standards recommend these industry-proven methods:

- **SameSite Cookie Attribute:** Setting cookies to `SameSite=Strict` or `SameSite=Lax` prevents the browser from sending them during cross-site requests, which is the most direct way to block CSRF.
- **Anti-CSRF Tokens (Synchronizer Token Pattern):** The server generates a unique, cryptographically strong token for the session. This token must be included in the body or headers of state-changing requests (like POST or DELETE). Since a malicious site cannot read this token due to the Same-Origin Policy (SOP), it cannot forge a valid request.
- **Double Submit Cookies:** A stateless alternative where the CSRF token is sent in both a cookie and a request parameter. The server then verifies that they match.
- **Custom Request Headers:** Since malicious cross-site forms cannot set custom HTTP headers, requiring a header like `X-Requested-With` or a custom token header effectively blocks CSRF.

Where Short Lifespans *Do* Help

Short-lived tokens or cookies are effective as **defense-in-depth**. They limit the damage if a token is somehow leaked (e.g., via a separate XSS vulnerability), ensuring that an attacker can only use the stolen credential for a very brief period.

For more comprehensive guidance, refer to the OWASP CSRF Prevention Cheat Sheet.