* + 1. SSO is an authentication method that enables users to securely authenticate with multiple applications and websites by using just one set of credentials.

True-SSO has the user authenticate with the Authentication Service Provider, which in turn notifies the Service Provider of the user’s authentication status.

With pseudo-SSO, the user must authenticate with each SP separately, using the predefined protocols of the pseudo-SSO component.

Local SSO occurs when the ASP/SSO component is local to the user platform. Proxy-based SSO occurs when the ASP/SSO component is offered as a service by an external entity.

* + 1. Local pseudo-SSO:

Local true-SSO: Unix user login

Remote pseudo-SSO:

Local pseudo-SSO: SSH/OAuth

* 1. The HttpOnly flag ensures that the cookie can only be accessed by the browser for the purpose of being sent to the server via HTTP requests, and cannot be accessed by a client-side script. This is designed to mitigate XSS attacks.
     1. POST is more appropriate here.
* With POST, the form data is not included in the URL, so if e.g., the user copies the URL and navigates to it again, it does not result in resubmission of the form.
* Semantically, POST is more appropriate here because data is being sent to the server, not queried from it.
* POST ensures that simply clicking on a hyperlink (to a page hosted on the correct domain) cannot trigger a form submission.
  + 1. This mitigates Cross-Site Request Forgery. Suppose an attacker created a website which, upon visiting, submitted an HTTP request to the real server on behalf of the user, sending along their session cookie if they happen to be logged in to the true site. If the server requires a valid CSRF token, then the attacker’s website would (ideally) have no way of knowing how to generate such a valid token.
    2. The server could only accept requests which come from the same domain (e.g., by setting the SameSite attribute on the session cookie). However, this is possible for an attacker to spoof.

The server could require the user to enter their password again to perform critical actions. This reduces the difference between the time of check and time of use of credentials, which is what CSRF exploits.

The server can make use of custom HTTP headers. By default, most browsers do not allow client-side scripts to make cross-origin HTTP requests with custom headers.