RECOMMENDATIONS

FOR DEVELOPERS OF SINGLE SIGN-ON

1. REQUEST MINIMAL PERMISSIONS

Developers should request only the permissions necessary for their application's functionality. This principle of least privilege minimizes the risk of exposing sensitive user data unnecessarily. Over-requesting permissions can erode user trust and may increase scrutiny during security assessments.

2. USE INCREMENTAL AUTHORIZATION

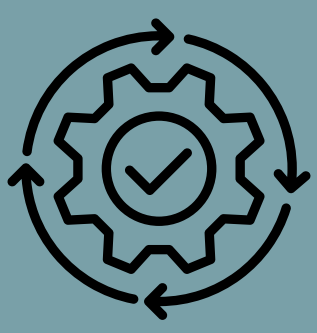
Incremental authorization allows your app to request permissions gradually as they are needed, instead of all at once during the initial login. This approach reduces the initial barrier for users to start using your app while allowing you to maintain security and privacy as additional features are accessed.

3.REQUEST PERMISSIONS IN CONTEXT

Providing clear and contextual explanations when requesting permissions helps users understand why your app needs specific access. This builds trust and improves user consent rates.

4. VERIFY TOKENS RECEIVED FROM A THIRD-PARTY IDP

Always validate the tokens you receive from an identity provider (IdP) by checking their signature, expiration, and intended audience. This ensures the token is legitimate, unaltered, and issued for your application, preventing misuse by malicious actors.



5.USE SECURE BROWSERS

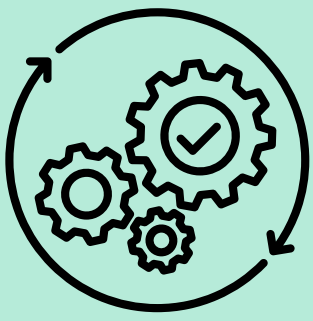
Ensure that the SSO flow operates only in secure and up-to-date browsers to protect against vulnerabilities such as man-in-the-middle (MITM) attacks.
Do not redirect the request through embedded browsing environments, including webviews on mobile platforms, such as WebView on Android or WKWebView on iOS.

RECOMMENDATIONS

FOR DEVELOPERS OF SINGLE SIGN-ON

6. ENFORCE EXACT PATHS IN OAUTH PROVIDERS CONFIGURATION

Configure your OAuth providers to enforce exact redirect URIs, preventing attackers from intercepting tokens through similar or mismatched paths. This ensures that tokens are only sent to the intended destination



7. USE PROOF KEY FOR CODE EXCHANGE (PKCE)

PKCE enhances OAuth 2.0 flows by mitigating CSRF and interception attacks. It introduces an additional verification step using a code challenge and code verifier, ensuring that authorization codes cannot be reused or intercepted by attackers.

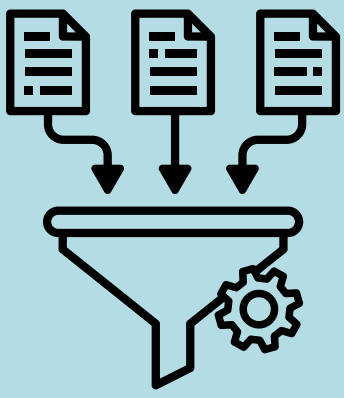
8. HANDLE TOKENS SECURELY

Tokens, such as access and refresh tokens, are critical to the SSO process and must be handled with utmost care.

Always store them securely at rest and ensure they are transmitted over HTTPS to prevent interception.

Never include your App Secret in client-side or decompilable code, as this exposes sensitive information to attackers.

Revoke tokens as soon as they are no longer needed and delete them permanently from your systems.



9. AVOID THE IMPLICIT OAUTH GRANT TYPE

The implicit grant type is less secure due to its exposure of tokens directly in the browser. Instead, use the authorization code grant type, which keeps tokens off the front-end and leverages server-side token handling for better security.