PKCE in the Security BCP

CSRF Protection

PKCE provides better CSRF protection than state and nonce:

- PKCE can be enforced by the AS
  - the client can’t “forget” to check
  - forces client to maintain a session
- With SHA256: CSRF protection even if authorization request leaks

Protection Layer for Authorization Codes

Misuse of authorization codes was a major driver for the security BCP.

(Log files, mobile devices, intermediaries, browser history, etc.)

For confidential clients:

Attackers need to use the code in a second flow to circumvent the client authentication (“code injection”). **PKCE and nonce (via c_hash) prevent this.**

For public clients:

Attackers can directly call the token endpoint. **Only PKCE prevents this.**

Client Type	**Response Type**	**Attacker Calls Token Endpoint** Prevented by:	**Auth Code Injection** Prevented by:	**CSRF** Prevented by:
Confidential	code	Client Authentication	PKCE	PKCE *or* nonce
Confidential	code+id_token	Client Authentication	PKCE *or* nonce+c_hash	PKCE *or* nonce
Public	code	PKCE	PKCE	PKCE *or* nonce
Public	code+id_token	PKCE	PKCE *or* nonce+c_hash	PKCE *or* nonce

Clients MUST prevent injection (replay) of authorization codes into the authorization response by attackers.* Public clients MUST use PKCE [RFC7636] to this end. For confidential clients, the use of PKCE [RFC7636] is RECOMMENDED. With additional precautions, described in Section 4.5.3.2, confidential clients MAY use the OpenID Connect nonce parameter and the respective Claim in the ID Token [OpenID] instead. In any case, the PKCE challenge or OpenID Connect nonce MUST be transaction-specific and securely bound to the client and the user agent in which the transaction was started.

Note: Although PKCE was designed as a mechanism to protect native apps, this advice applies to all kinds of OAuth clients, including web applications.

When using PKCE, clients SHOULD use PKCE code challenge methods that do not expose the PKCE verifier in the authorization request. Otherwise, attackers that can read the authorization request (cf. Attacker A4 in Section 3) can break the security provided by PKCE. Currently, S256 is the only such method.

Authorization servers MUST support PKCE [RFC7636].

* better: “Clients MUST prevent *misuse of authorization codes* and injection (replay) of authorization codes into the authorization response …”

Compatible to Existing Implementations?

Adding a protection layer for authorization codes and robust CSRF protection is not 100% compatible with OIDC Core:

- state is so far not required
- nonce is so far not required in response_type=code flows
- PKCE is so far not required for public clients

Same goes for other defenses, like precise redirect URI matching.

Further Thoughts…

- Security BCP makes AS-support for PKCE mandatory
- Enforcing PKCE on the AS is an easy way to prevent PKCE Downgrade attacks

Summary of Discussion 2022-03-23 @ IETF113

- Current rules are fine
  - Obligations for clients to use PKCE, not for servers to enforce PKCE
- Spec needs more explanation