OAuth 2.0 Security Best Current Practice

Torsten Lodderstedt, John Bradley, Andrey Labunets, Daniel Fett
Refines and enhances security guidance for OAuth 2.0 implementers

Updates, but does not replace:
- OAuth 2.0 Threat Model and Security Considerations (RFC 6819)
- OAuth 2.0 Security Considerations (RFC 6749 & 6750)

- Updated, more comprehensive Threat Model
- Description of Attacks and Mitigations
- Simple and actionable recommendations
Changes Since IETF-104 (-12..-13)
Discourage use of
Resource Owner Password Credentials Grant

- → R.O.P.C.G. MUST NOT be used
- Exposes credentials to the client
- Increased attack surface
- Not or not easily adaptable to modern authentication methods
  - 2FA
  - WebAuthn
  - WebCrypto
  - Multi-step authentication
Client impersonating Resource Owner

- Input from Neil Madden
- Confusion between “sub” used for client in client credentials grant and “sub” for a resource owner in auth code grant
- E.g.: client uses dynamic registration and can influence its “sub” value such that it becomes identical to a “sub” of a resource owner
- → client SHOULD NOT be able to select “sub” value
PKCE

- Encourage use of PKCE mode “S256” (instead of PLAIN)
  - “... SHOULD use PKCE code challenge methods that do not expose the PKCE verifier in the authorization request”
- AS MUST support PKCE
- AS SHOULD publish PKCE support
- PKCE MAY replace state for CSRF protection
  - ... under certain conditions!
  - → see later
Open Questions
Make Metadata Mandatory?

- Clients can **rely on PKCE** only when they know that AS supports PKCE
  - In particular, clients need to know if the AS supports PKCE when they want to drop other CSRF countermeasures
- Current status: AS **SHOULD** use metadata to announce support for PKCE
- “**MUST**” would make RFC8414 (AS Metadata) mandatory for ALL implementations
PKCE Chosen Challenge Attack

- Prerequisites:
  - Attacker can read authorization response (through a leaked/logged URI, Mix-Up, …)
  - Attacker can bring his victim to visit a URI and authorize “honest RP” (e.g., malicious app, phishing website, …)
1) Attacker starts flow with RP

2) User authorizes RP

3) Attacker uses access token via RP
What can we do about this?

- Use Token Binding (lack of support)
- Use Form Post Response Mode (relatively big change)
- Check Origin/Referer header at AS (lack of support; spec not suitable)
- ???
- IVAR!
After receiving authz request, AS checks with client if
- the request came from the client’s session with the user,
- and whether it was manipulated.
IVAR Protocol

1. The client signals in its metadata that it supports IVAR and publishes its IVAR URI.
2. The client stores the authorization request URI in the user browser’s web storage.
3. AS opens the IVAR URI in an iframe and sends the authz URI in a postMessage.
4. JavaScript at IVAR URI checks web storage and answers “ok” if match for authz URI is found.
IVAR

- Provides a fallback if JavaScript is disabled.
- Checks the integrity/origin of state, nonce, request_uri, ..., and redirect_uri!
- Thus protects against
  - PKCE Chosen Challenge Attack
  - Attacks using manipulated redirect URIs
  - A variant of the Mix-Up attack
  - ...

Feedback welcome!
https://tools.ietf.org/html/draft-fett-oauth-ivar-00
Ready for Publication?
Q & A