OAuth 2.0 for Browser Based Apps

OAuth Security Workshop 2019

David Waite, Principal Technical Architect, Ping Identity

Purpose

- Best Current Practices Document
- Builds mostly on
  - RFC 8252 - OAuth 2.0 for Native Apps
  - I-D oauth-security-topics - OAuth 2.0 Security BCP

What is a Browser-Based App

- Application code delivered to and run inside a browser
  - Single Page Application (SPA)
  - Progressive Web Application (PWA)
- This BCP is around how to support these applications as OAuth Public Clients

Similarities to AppAuth

- Public Client
- Authorization Code flow
- Recommend against Implicit
- PKCE
- Ownership of HTTPS URLs
  - justification to persist consent

How do they differ?

- Running sandboxed inside user agent
  - No reason to prefer an external user agent
  - Actually: external user agent conflicts with multiple local browsers
- Same Origin security model - require exceptions using CORS
- Greater potential for malicious code injection and traffic capture
  - XSS, third party script loading, site compromise, service workers
  - Recommend Content-Security-Policy

No Implicit

- In addition to the Security Topics BCP:
  - Supporting the implicit flow requires additional code and a different set of security considerations
  - Browser and Native apps versions/packaging should not require two different OAuth flows
  - id_token in front channel must be signature verified, which clients might skip. Delivery over backchannel is more secure by default.

URL Ownership as (weak) Client Authentication

- Using the Authorization Code Flow
- Code can only be delivered to redirect URL owned by the application
- Used to justify:
  - Cached consent
- Requires HTTPS URL schemes only
  - Client cannot have custom scheme or localhost redirect URIs

Refresh Tokens

- Current guidance is that the AS should not issue refresh tokens
  - due to fears of token theft
- Would like to change to give more (and possibly general) guidance
- Lack of token binding (or alternatives) hurts implementors

Architectural Alternatives

- “When you could avoid using OAuth”
  - Difficult line for guidance - telling people to punt on well-defined security recommendations/considerations for simplicity
  - May use cookie instead of access tokens - HttpOnly, SameSite
- Same domain, first-party apps
- OAuth-handling component
  - Back-end API for doing OAuth flow
  - Front-end reverse proxy authenticating and applying OAuth

Eventual Wishes

- Sender constrained / PoP mechanism for browsers
- General (non-FUD) guidance on token lifetime policy
- General guidance on policy geared toward public / confidential clients
  - not based on client implementation technology
- Guidance against localhost/custom URL
- External User Agent UX