Simplified Integration of OAuth into JavaScript Applications

IETF 101

Travis Spencer – Curity.io

@travisspencer

Problem

- Front-end developers need dirt-simple integration
  - Reduce number of parameters required
  - Provide ready APIs (JavaScript libs) and examples
  - Avoid client management of multiple tokens
- The SPA problem must be solved inside the OAuth server

The Assisted Token Flow

Example: User is already authenticated

1. Open hidden iframe and do GET on the assisted token endpoint with client_id
2. Assisted token endpoint serves page that performs postMessage to parent frame
3. Parent page receive event with AT and closes iframe.

Success postMessage

- The success postMessage should at least contain:

```
{
   status: "success",
   access_token: “ABCDEFGH”,
   expires_in: 1499,
   scope: ”read write”
}
```


Add ID token if scope includes openid

- Together with target domain:

```
channel.postMessage(data, ”https://origin_of_client");
```

Authenticating

- If the user doesn’t have a session
    1. The Client must be told, so it can take action
    2. Must work without the need for OIDC
    3. Client cannot inspect child frame to find location, browser prohibits.

- Options
    1. Let the client TIMEOUT
    2. prompt=none (respond with error if no session)
    3. postMessage(“authenticating”,…

The Assisted Token Flow

Example: Authenticating user

```
<html>
<script>
  window.parent.postMessage("authenticating", …)
</script>
<form>
  <input name=username>
  <input name=password>
</form>
</html>
```

1. Open hidden iframe window and do GET on the assisted token endpoint with client_id
2. Assisted token endpoint serves page that performs postMessage to parent frame informing about the authentication that needs to take place
3. The Client library can then close the hidden iframe, and restart the flow with a visible frame or child window.
4. RO interacts with the content of the visible frame/window

The Assisted Token Endpoint

- Why a new endpoint
  - No overloading = fewer required parameters
  - Clear separation from existing protocol
- Equivalent to new grant type

- Takes 1 parameter mandatory:
  - `client_id`

- Optional parameters:
  - `scope`
  - `for_origin (see table)`
  - `reuse`
  - *`forceAuthN`*, *`freshness`* `(OIDC overlay)`

scope	Issue all scopes configured for client if empty
for_origin	Required if more than one origin is configured on client
reuse	Reuse existing session, default true
forceAuthN, freshness	Require new authentication to take place (Open ID overlay)

for_origin

- Not the same as redirect_uri
  - The framer isn’t necessary the one to be called during implicit flow.
- A client is REQUIRED to be configured with an allowed origin.
  - The domain from which the client is framed.
  - If more than one is configured, the client needs to send for_origin together with client_id.
- The AS can then ensure secure framing
  - Using X-FRAME-OPTIONS (can only handle one origin)
  - Also, server MUST respond with CSP headers (supports multiple origins)

scope

- If no scope is requested:

- **ALL configured scopes are returned**

  - Different from regular authorization + token endpoints
  - Reason: make typical requests easier – no extra param

info@curity.io

@jacobideskog

jacob.ideskog@curity.io