The Problem

Users are being tracked around the web without their consent and without any control.
Common forms of Tracking on the Web

- Link Decoration
- Network Addresses
- Third-Party Cookies
- Fingerprinting
Common forms of Tracking on the Web

- Link Decoration / Bounce Tracking
- Third-Party Cookies
- Network Addresses
- Fingerprinting
Common forms of Tracking on the Web

- Link Decoration / Bounce Tracking
- Third-Party Cookies
- Fingerprinting

Adapted from Tim Cappalli's OSW 2024 Presentation • tcslides.link/osw24-fedcm101
Third-Party Cookies

example.com

tracker.example

example.org

tracker.example

tracker.example
OIDC Front-Channel Logout

Identity Provider

Relying Party A
iframe
tcslides.link/osw24-fedcm101
example.com

Relying Party B
iframe
tcslides.link/osw24-fedcm101
example.net

Relying Party C
iframe
tcslides.link/osw24-fedcm101
example.org

identity-provider.net

Adapted from Tim Cappalli's OSW 2024 Presentation • tcslides.link/osw24-fedcm101
iframe silent token refresh
Link Decoration / Bounce Tracking

example.com

user=1234

tracker.example?user=1234

user=1234

example.org?user=1234

user=1234

Adapted from Tim Cappalli’s OSW 2024 Presentation • tcslides.link/osw24-fedcm101
OAuth/OIDC/SAML rely on redirects

Adapted from Tim Cappalli's OSW 2024 Presentation • tcslides.link/osw24-fedcm101
OAuth/OIDC/SAML redirects

These redirects are virtually indistinguishable from link decoration tracking, especially multi-hop federation

https://authorization-server.com/authorize
?response_type=code
&client_id=3VzsQHnFhALpExMrhSg2E1Qx8zAA1QW
&code_challenge=o_MKLFeQIBGDm26GltsBsoZBre2fiJ0xmHNdzUxKo
&code_challenge_method=S256
&scope=offline_access
&state=dbe329477e158127ea52
&redirect_uri=https%3A%2F%2Fexample-app.com%2Fclient

Adapted from Tim Cappalli’s OSW 2024 Presentation • tcslides.link/osw24-fedcm101
The Solution?

**Federated Identity Community Group @ W3C**

[https://www.w3.org/community/fed-id/](https://www.w3.org/community/fed-id/)

Started in 2021

**Federated Identity Working Group @ W3C**

[https://www.w3.org/groups/wg/fedid/](https://www.w3.org/groups/wg/fedid/)

Started in June 2024

Anyone can join

Limited to W3C Member Companies
Two Proposals

Federated Credential Management API (FedCM)
Cross-Site Cookie Access Credential (CSCAC?)
FedCM

Provides a browser API to manage federated identity flows on the web.

Ensures there is *no information disclosed* between RP and IdP until the user consents.
FedCM

- User starts at RP/Client
- The RP initiates the FedCM API
  - The user sees a prompt and consents to use an account at an IdP
- The browser (not the RP/Client!) sends requests to the IdP endpoints
- The browser returns data from the IdP to the RP/Client

Note: I use RP/Client interchangeably here. RP is OIDC/FedCM terminology, Client is OAuth terminology.
Webmention.io

Webmention.io is a hosted service created to easily receive webmentions on any web page.

You might also be interested in reading about this project on the IndieWeb wiki.

Web Sign-In

https://example.com

Sign in
const identityCredential = await navigator.credentials.get({
    identity: {
        context: "signin",
        providers: [
            {
                configURL: "https://authorization-server.com/fedcm/config.json",
                clientId: "C1234567890",
                params: {...}
            },
        ],
    },
}).catch(e => {
    console.log(e);
});
Browser fetches FedCM IdP Config

No cookies, no Origin, no Referer

GET /fedcm/config.json
Host: authorization-server.com
Accept: application/json
Sec-Fetch-Dest: webidentity

{
    "accounts_endpoint": "/fedcm/accounts.php",
    "client_metadata_endpoint": "/fedcm/client_metadata.php",
    "id_assertion_endpoint": "/fedcm/assertion.php",
    "disconnect_endpoint": "/fedcm/disconnect.php",
    "revocation_endpoint": "/fedcm/revoke.php",
    "login_url": "/fedcm/login.php",
    "branding": {
        "background_color": "blue",
        "color": "0xFFEEAA",
        "icons": [{
            "url": "https://authorization-server.com/fedcm/icon.ico",
            "size": 25
        }]
    }
}"
Browser fetches Accounts endpoint

IdP cookies, no Origin, no Referer, no redirects

GET /fedcm/accounts.php
Host: authorization-server.com
Accept: application/json
Cookie: IdPCookie123456
Sec-Fetch-Dest: webidentity

{
    "accounts": [
        {
            "id": "1",
            "given_name": "Aaron",
            "name": "Aaron Parecki",
            "email": "aaron@parecki.com",
            "picture": "https://aaronparecki.com/images/profile.jpg"
        },
        ...
    ]
}
Browser fetches Client Metadata endpoint at IdP

No cookies, RP Origin, no redirects

GET /fedcm/client_metadata.php?client_id=C1234567890
Host: authorization-server.com
Accept: application/json
Sec-Fetch-Dest: webidentity

{
    "privacy_policy_url": "https://example-app.com/privacy_policy.html",
    "terms_of_service_url": "https://example-app.com/terms_of_service.html"
}

Yes, this seems backwards.

This is the browser fetching the client metadata that has been pre-registered at the IdP.

See [Issue #581](https://example-app.com/issue-581) to allow RPs to provide their own metadata when they are not pre-registered.
Browser fetches Identity Assertion endpoint

IdP cookies, RP Origin, no redirects

POST /fedcm/assertion.php
Host: authorization-server.com
Accept: application/json
Cookie: IdPCookie123456
Origin: https://example-app.com/
Content-type: application/x-www-form-urlencoded
Sec-Fetch-Dest: webidentity

account_id=1&
client_id=C1234567890&
disclosure_text_shown=true&
(custom params included, TBD)

{
   "token": "(ARBITRARY STRING)"
}
Cross-Site Cookie Access Credential

Provides a browser API to enable a website to make a string available to another origin after prompting the user.
IdP

navigator.credentials.store({
  identity: {
    id: "user1",
    effectiveQueryURL: ["https://example-app.com"],
    uiHint: {
      name: "Aaron Parecki",
      iconURL: "https://authorization-server.com/photos/user1.jpg"
    }
  }
  token: dataToBeSharedWithRP,
});
let credential = await navigator.credentials.get({
    identity: {
        providers: [
            {
                loginURL: "https://authorization-server.com/login"
            }
        ]
    }
});

// User sees prompt

console.log(credential.token)
OAuth Profile for FedCM

https://github.com/aaronpk/oauth-fedcm-profile

Not yet an I-D, this is more of an implementation guide until FedCM stabilizes and adds the necessary features.

Has been prototyped by myself and Filip.

OAuth Profile for Cross-Site Cookie Access Credential

Not yet shipped with UI in Firefox Nightly, and
did not have time to prototype this before IETF 120.
OAuth Profile for FedCM

- Send OAuth request parameters in “params”
- Receive an authorization code from the Identity Assertion endpoint
- Exchange authorization code for access token as you normally would, outside of FedCM

Notes

- This pattern enables confidential clients and extensions like PAR
- No redirects, roughly equivalent to prompt=none
const identityCredential = await navigator.credentials.get({
  identity: {
    context: "signin",
    providers: [
      {
        configURL: "https://authorization-server.com/fedcm/config.json",
        clientId: "C1234567890",
        params: {
          "response_type": "code",
          "scope": "photos:read photos:write",
          "code_challenge": "0T8ahiSvW_c6g35YPG2Jz9X8t1ZbHfNFCVuAft94bB0",
          "code_challenge_method": "S256"
        }
      },
      ...
    ].catch(e => {
      console.log(e);
    });
Browser fetches Identity Assertion endpoint

IdP cookies, RP Origin, no redirects

POST /fedcm/assertion.php
Host: authorization-server.com
Accept: application/json
Cookie: IdPCookie123456
Origin: https://example-app.com/
Content-type: application/x-www-form-urlencoded
Sec-Fetch-Dest: webidentity

account_id=1&
client_id=C1234567890&
disclosure_text_shown=true&
param_response_type=code&
param_scope=photos:read+photos:write&
param_code_challenge=OT8ahiSvW_c6g35YPG2Jz9X8tlZbHfNFCVuAft94bB0&
param_code_challenge_method=S256

{
    "token": "(OAUTH AUTHORIZATION CODE)"
}
JS client or app backend fetches Token Endpoint

No restrictions on cross-origin requests at this point

POST /oauth/token
Host: authorization-server.com
Accept: application/json
Content-type: application/x-www-form-urlencoded

grant_type=authorization_code&
client_id=C1234567890&
code=(authorization code)&
code_verifier=a6128783714cfda1d388e2e98b6ae8221ac31aca31959e59512c59f5

{
    "access_token": "2YotnFZFEjr1zCsicMWpAA",
    "token_type": "Bearer",
    "expires_in": 3600,
    "refresh_token": "tGzv3JOkF0XG5Qx2T1KWIA"
}
Recap

- Send OAuth request parameters in “params”
- Receive an authorization code from the Identity Assertion endpoint
- Exchange authorization code for access token as you normally would, outside of FedCM

Notes

- This pattern enables confidential clients and extensions like PAR
- No redirects, roughly equivalent to prompt=none

https://github.com/aaronpk/oauth-fedcm-profile
const loginChallengeResponse = await fetch("/auth/fedcm-start", {
    method: "POST"
});
const loginChallenge = await loginChallengeResponse.json();

const identityCredential = await navigator.credentials.get({
    identity: {
        context: "signin",
        providers: [
            {
                configURL: "https://authorization-server.com/fedcm/config.json",
                clientId: loginChallenge.client_id,
                params: {
                    "request": loginChallenge.request
                }
            }
        ],
    },
}).catch(e => {
    console.log(e);
});
FedCM Open Questions/Issues

- **#319** Multiple IdP Support
- **#240** “Any” IdP Support - enable RPs to accept any IdP of specific types
- **#581** Allow RPs to provide their own ToS and privacy policy URLs
- **#580** Allow IdPs to verify using DNS instead of .well-known at eTLD+1
- **#578** Allow IdPs to return arbitrary JSON instead of just a “token” string
- **#556** Passing arbitrary parameters to the assertion endpoint
Feedback Request

W3C WG wants feedback on the "params" feature (issue #556)
let {token} = await navigator.credentials.get({
  identity: {
    providers: [{
      configURL: "https://idp.example/fedcm.json",
      clientId: "63993123b3b56",
      loginHint: "previous@user.com",
      // Parameters that need to be passed from the RP to the IdP
      // but that don't play any role with the browser.
      params: {
        "response_type": "code",
        "scope": "photos:read photos:write",
        "code_challenge": "OT8ahiSvW_c6g35YPG2Jz9X8t1ZbHfNFCVuAft94bB0",
        "code_challenge_method": "S256"
      }
    }]
  }
});
Option 1 - Prefix Custom Parameters *(Current Implementation)*

POST /fedcm_assertion_endpoint HTTP/1.1
Host: idp.example
Origin: https://rp.example/
Content-Type: application/x-www-form-urlencoded
Cookie: 0x23223
Sec-Fetch-Dest: webidentity

account_id=123
&client_id=63993123b3b56
&disclosure_text_shown=false

&param_response_type=code
&param_scope=photos:read+photos:write
&param_code_challenge=OT8ahiSvW_c6g35YPG2Jz9X8t1ZbHfNFCVuAft94bB0
&param_code_challenge_method=S256
Option 2 - Prefix FedCM Parameters

POST /fedcm_assertion_endpoint HTTP/1.1
Host: idp.example
Origin: https://rp.example/
Content-Type: application/x-www-form-urlencoded
Cookie: 0x23223
Sec-Fetch-Dest: webidentity

fedcm_account_id=123
&fedcm_client_id=63993123b3b56
&fedcm_disclosure_text_shown=false
&response_type=code
&scope=photos:read+photos:write
&code_challenge=OT8ahiSvW_c6g35YPG2Jz9X8t1ZbHfNFCVuAft94bB0
&code_challenge_method=S256
Option 3 - JSON Request

POST /fedcm_assertion_endpoint HTTP/1.1
Host: idp.example
Origin: https://rp.example/
Content-Type: application/json
Cookie: 0x23223
Sec-Fetch-Dest: webidentity

```
{
  "fedcm": {
    "account_id": "123",
    "client_id": "63993123b3b56",
    "disclosure_text_shown": false
  },
  "oauth": {
    "response_type": "code",
    "scope": "photos:read photos:write",
    "code_challenge": "OT8ahiSwW_c6g35YPG2Jz9X8tlZbHfNFCVuAft94bB0",
    "code_challenge_method": "S256"
  }
}
```

FedCM Parameters

Custom Parameters

Note: This will trigger a CORS pre-flight check!