Doing OAuth the secure way

DARKO KUKOVEC

JUNE 2023

neverstop

INFINUM

Who am I

Darko Kukovec
Head of Frontend Department @ Infinum

- working with frontend (professionaly) for 11 years
- working with OAuth2 for 5-ish years
- helped two enterprise clients roll out their own OAuth2 based identity federation

01

Why is SSO/OAuth2 great?

Improved User Experience

SSO eliminates the need for users to remember and manage multiple sets of login credentials, making it easier and more convenient for them to access the systems and applications they need.

Increased Security

SSO reduces the risk of password reuse and sharing, which can lead to security breaches.

Additionally, SSO can also improve security by requiring users to go through a stronger authentication process, such as multi-factor authentication, before granting access.

Reduced IT Costs

SSO can reduce the costs associated with managing multiple sets of login credentials and resetting forgotten passwords.

Better Compliance

SSO can help organizations meet compliance requirements by providing a centralized location for managing and auditing access to sensitive information.

02

What's wrong with OAuth2?

02

What's wrong with OAuth2 in a JS app?

Cookies vs localStorage

~~Cookies~~ vs localStorage

localStorage XSS issue scenarios

Dependencies

A rogue dependency stealing user data

User input

User input that is not correctly escaped

External script

A script that someone added without checking what the script does
e.g. via GTM or a browser extension

~~Cookies~~ vs ~~localStorage~~ vs ???

In-memory storage

In-memory storage

Better security

Sensitive data can be saved in JS closures

Bad UX

Log in on every page refresh or tab open

Insecure

Bad actors can still intercept network requests

03

The solutions...

Workgroup: Web Authorization Protocol
Internet-Draft:
draft-ietf-oauth-browser-based-apps-13
Published: 13 March 2023
Intended Status: Best Current Practice
Expires: 14 September 2023

A. Parecki
Okta
D. Waite
Ping Identity

OAuth 2.0 for Browser-Based Apps

Abstract

This specification details the security considerations and best practices that must be taken into account when developing browser-based applications that use OAuth 2.0.

https://datatracker.ietf.org/doc/html/draft-ietf-oauth-browser-based-apps

What are our options?

6.2 Backend For Frontend (BFF) Proxy

In this architecture, commonly referred to as "backend for frontend" or "BFF", the JavaScript code is loaded from a BFF Proxy server (A) that has the ability to execute code and handle the full OAuth flow itself. This enables the ability to keep the request to obtain an access token outside the JavaScript application.

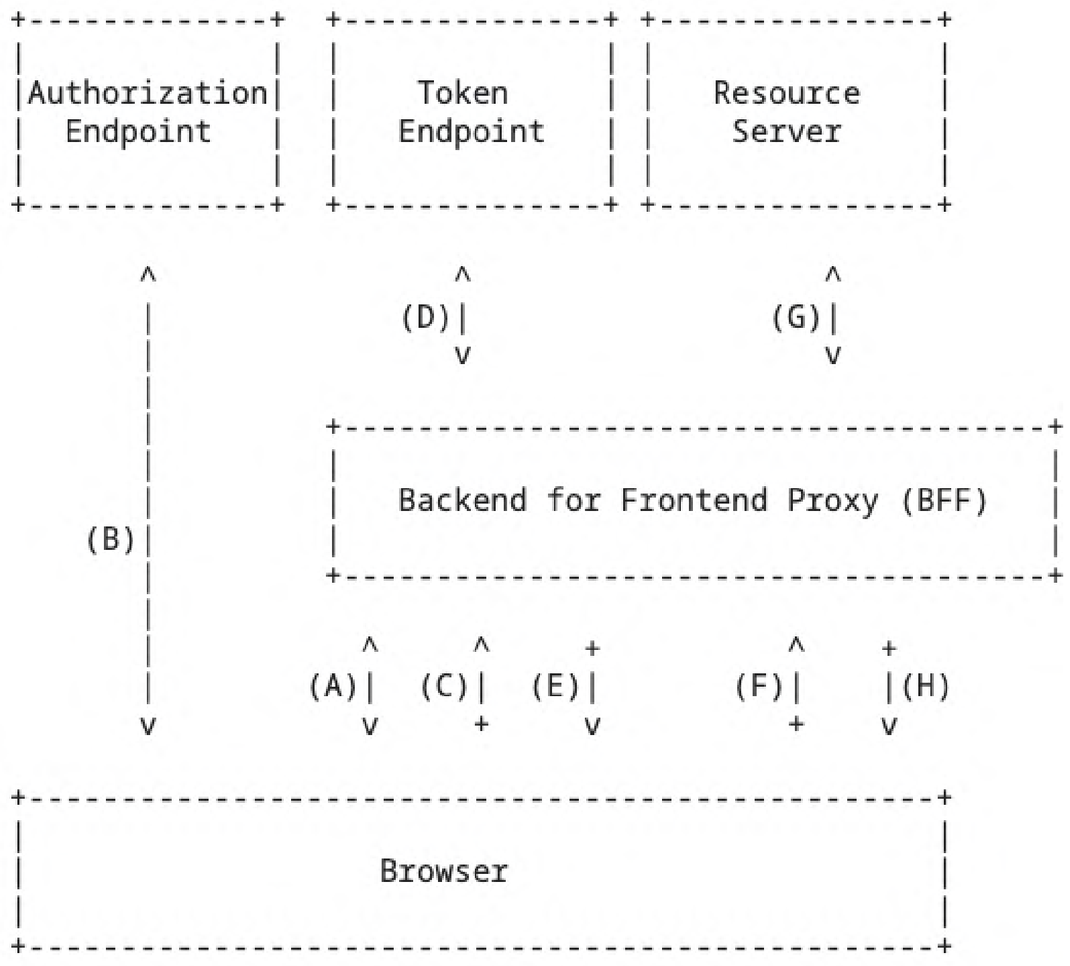
6.3 Token-Mediating Backend

An alternative to a full BFF where all resource requests go through the backend is to use a token-mediating backend which obtains the tokens and then forwards the tokens to the browser.

6.4 JS Apps obtaining tokens directly

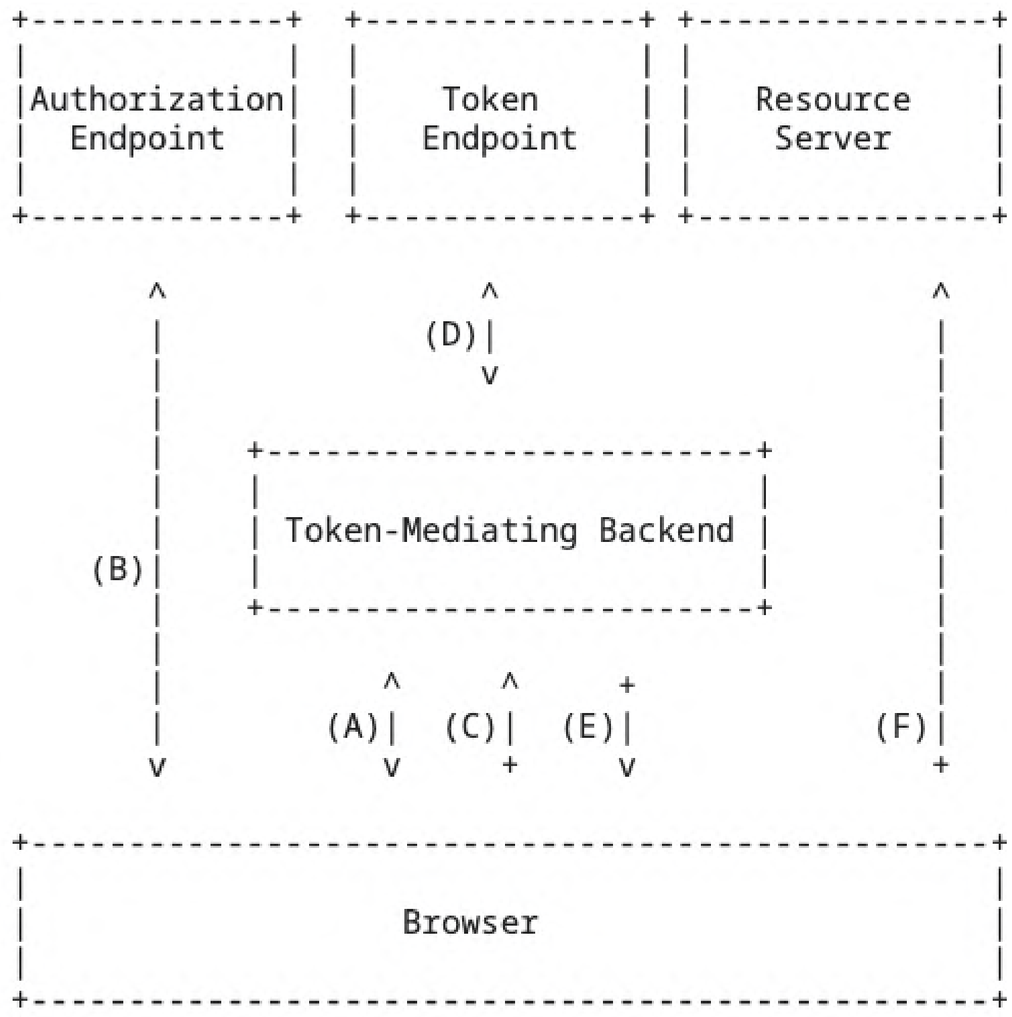
This section describes the architecture of a JavaScript application obtaining tokens from the authorization server itself, with no intermediate proxy server and no backend component.

Backend For Frontend (BFF) Proxy



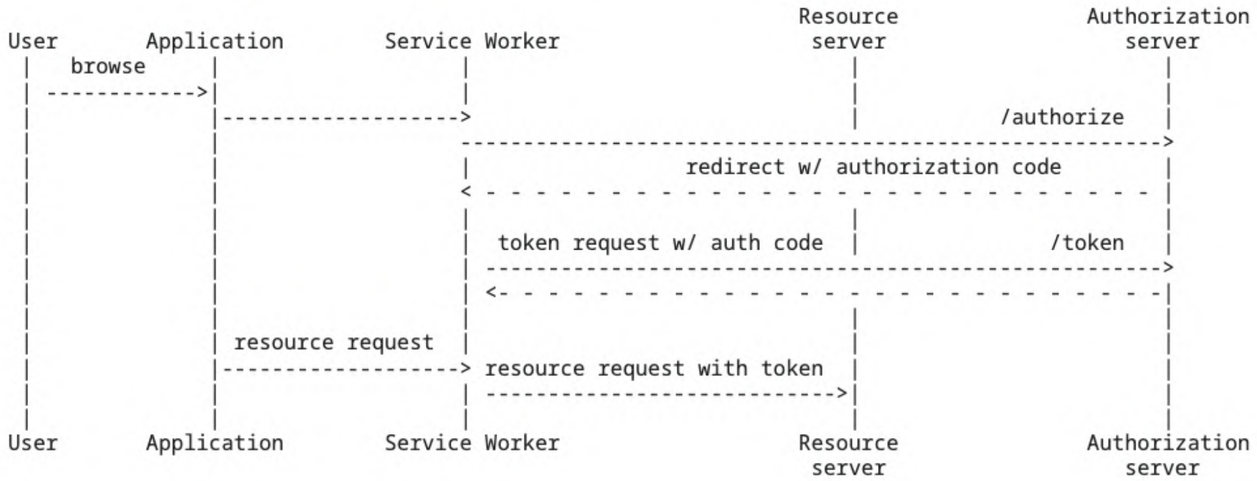
- http-only cookies or backend storage
- BFF is handling the conversion between sessions

Token-Mediating Backend



- custom backend session
- frontend is only using the custom session
- resource server also needs to support it

JavaScript Applications obtaining tokens directly



- Service Workers
  - Intercept network requests in separate context
  - Long(er) living - in-memory data lives longer than a page or tab
- No good way to persist the data while beeing 100% secure
  - Encryption is a good 1st step, but the key is still public

What are our options?

6.2 Backend For Frontend (BFF) Proxy

In this architecture, commonly referred to as "backend for frontend" or "BFF", the JavaScript code is loaded from a BFF Proxy server (A) that has the ability to execute code and handle the full OAuth flow itself. This enables the ability to keep the request to obtain an access token outside the JavaScript application.

6.3 Token-Mediating Backend

An alternative to a full BFF where all resource requests go through the backend is to use a token-mediating backend which obtains the tokens and then forwards the tokens to the browser.

6.4 JS Apps obtaining tokens directly

This section describes the architecture of a JavaScript application obtaining tokens from the authorization server itself, with no intermediate proxy server and no backend component.

04

How are others doing it?

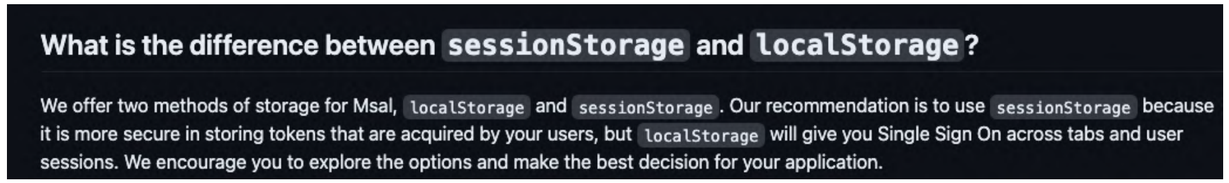
Auth0

```
16    if (location.search.includes("state=") &&
17        (location.search.includes("code=") ||
18        location.search.includes("error="))) {
19      await auth0Client.handleRedirectCallback();
20      window.history.replaceState({}, document.title, "/");
21    }
```

- auth0-spa-js
- App can read auth code
- Cookies / Local Storage / Session Storage

❌ Not following the best practices

Microsoft Authentication Library (MSAL)



- msal-browser
- Manual redirect handling
- Cookies / Local Storage / Session Storage

❌ Not following the best practices

Others

- Looked at a bunch of most popular SPA libs
- Not one of them is following the best practices 🤦‍♂️
- Service Workers
  - Hard to work with - tooling is still not good enough
  - Can cause some caching issues when updating
  - Browser support is fine, only outlier is Firefox in Private Mode (for now)

Our solution

- auth-worker
  - Service Workers
  - Optional persisting with encyption
  - Still WIP

05

Bonus: Callback URL validation

Booking.com + Facebook OAuth

https://salt.security/blog/traveling-with-oauth-account-takeover-on-booking-com

Two main issues

Facebook domain validation

Facebook is validating only the callback domain, not the whole path

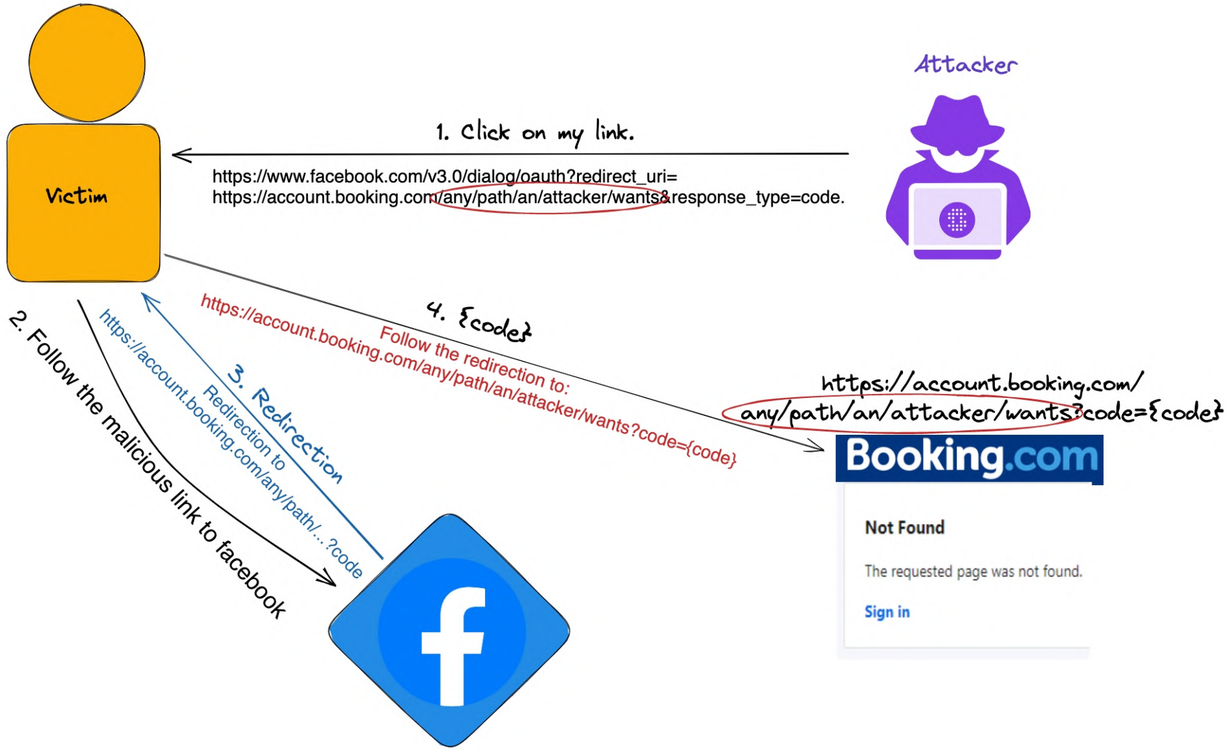
Booking.com page with redirects

Booking.com had a page that allowed redirects to any URL without verifying the domain

Some smaller problems

e.g. the Booking redirect was encoded so it was not immediatelly obvious what is happening

How to exploit



- Construct a custom login link
- Callback url is to the redirect page redirecting to an external page
- Query params are passed to the external page

How to mitigate

Domain validation

Make sure the provider is validating the whole path, not just the domain

No unexpected redirects

If you're redirecting users based on a parameter (basically an user input), make sure it's of an expected value (domain, path)

Thanks!

neverstop

Instagram Facebook Twitter Dribbble

INFINUM