TMI-BFF
Token Mediating and session Information Backend For Frontend

Vittorio Bertocci
Brian Campbell
TL;DR

• Mechanism for a JS frontend to delegate token requests and storage to its backend, while retaining the ability to invoke API directly from frontend code

• Formalization of a common practice
Important Note

• Some people interpret BFF as “the backend does everything, including routing API calls”

• That’s NOT how we use the term here. We call that “Full BFF”

• In this document we use BFF to indicate that the backend exists and does some of the work

• BFF TMI does NOT describe a Full BFF

• BFF TMI does NOT try to replace a Full BFF
Agenda

• Why TMI-BFF
• Main Flow
• Discussion
Why TMI BFF (1/3) – vs Full BFF, Reverse Proxy

• Whenever possible, it is more secure to keep tokens out of the browser
  • Eg JS frontend accessing API via reverse proxy on backend
• But it is not always possible
  • Performance
  • Costs
  • Backend limitations
  • Misc requirements
    • Regions, etc
Why TMI BFF (2/3) – vs Code+PKCE

• Code+PKCE from JS is viable, but complex
  • Config settings
  • Many moving parts
  • Requirements not satisfied by every AS (RT rotation, CORS)
Why TMI BFF (3/3) – Common Workaround

• Developers use confidential code flows on the backend to obtain ATs...
• ...and pass them back to their JS to perform API calls from the browser
• Issues
  • Custom code
  • Custom frontend-backed protocol
  • No threat model
  • No interop, every app/dev stack reinvents the frontend-backend relationship
TMI-BFF Elements

• Two new endpoints on the app
  • bff-token – for the frontend to ask for ATs from the backend
  • bff-sessioninfo - for the frontend to ask the backend session info (e.g. username)
• Message format for requesting ATs, session info
• Error messages
• Security considerations
Main Flow (1/3) - Prerequisites

1. Classic web sign on the app, establish session
2. Get access token (AT), any other artifact (RTs?)
Main Flow (2/3) - getting an AT

Frontend:
1. requests an AT from the backend
2. receives an AT
3. calls the API

GET /.well-known/bff-token?scope=buy+sell &resource=https%3A%2F%2Fapi.example.org%2Fstocks HTTP/1.1
Host: myapp.example.com
Cookie: super-secure-session=hVQvkyX2I0j36fqIoUQF1BeALbh

HTTP/1.1 200 OK
Content-Type: application/json
X-Content-Type-Options: nosniff
Cache-Control: no-cache, no-store
Cross-Origin-Resource-Policy: same-origin
Content-Security-Policy: sandbox
Cross-Origin-Opener-Policy: same-origin
X-Frame-Options: DENY

{ "access_token":"4bWc8ESC9aCc77LTc8EjR1pCFe4WxfNg", "expires_in":3596, "scope":"buy sell" }
Main Flow (3/3) - Getting Session Info

GET /.well-known/bff-sessioninfo HTTP/1.1
Host: myapp.example.com
Cookie: super-secure-session=hVQvkyX2I0j36fQIoUQFlBeALbh

HTTP/1.1 200 OK
Content-Type: application/json
X-Content-Type-Options: nosniff
Cache-Control: no-cache, no-store

{  
  "iss": "https://as.example.com",
  "sub": "24400320",
  "exp": 1311281970,
  "auth_time": 1311280969,
  "preferred_username": "johnny",
  "emailVerified": "johnny@example.com",
  "given_name": "Jonathan",
  "family_name": "Swift"
}
Advantages

• JS is ultrasimple (no config whatsoever)
• The new endpoints can be easily added to existing middleware
• API calls are performed from the user-agent
  • Less burden on backend, better perf
• Works with any AS, including old implementations
• Easy mix & match interop between frontend stacks (react, angular etc) and backend stacks (Node, Ruby, ASP.NET etc)
• Works with any sign in tech (as long as it results in a session cookie)
• Easy testing, mocking, etc
Changes in draft -01

• Extra security measures in the HTTP headers when returning ATs
• Removal of all claims of security benefits
• Clarified “BFF” vs “Full BFF”
• Recommendation to go Full BFF when viable
• More thorough explanation of prerequisites (preexisting session, tokens)
• Clarified relationship with the Browser BCP

TL;DR 2

- People are doing this today, without any guidance. Two possibilities:
  - We find reasons for which this is so insecure NO ONE should do this. We articulate that super clearly and start a campaign against it
  - We find the approach acceptable. In that case, leaving developers to their own device without guidance is less than ideal
Open Issues

• Should we handle interactive token acquisition case?