PUSH AUTHORIZATION REQUEST

En .net 9

Acerca de quien les habla

Marcos Fabián Polischuk

- 37 años
- Ingeniero en sistema de información, egresado de la UTN FRRe
- Desarrollador fullstack (.net, react, SQL Server, Azure)
- Arquitecto de software
- Entusiasta del café
- Fanático de los mmorpg, el animé y cultura japonesa
- Catador de memes de programación

/marcos.polischuk.3
@MFPolischuk
/markfab182
/in/marcos-fabian-polischuk
/MPolischuk

Authentication vs Authorization

Authentication vs Authorization

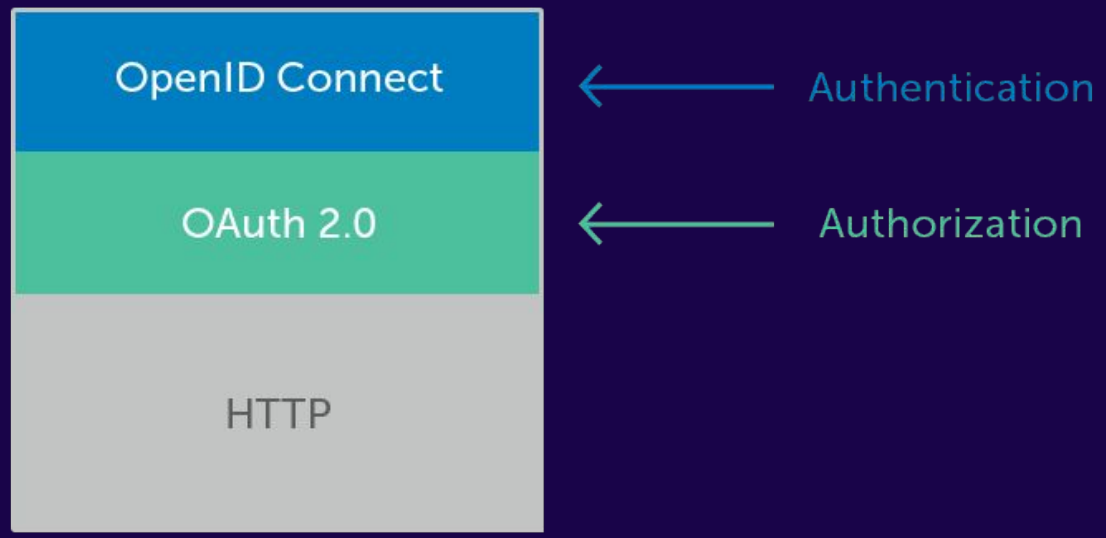
La **autenticación** es el proceso de verificar una identidad, es decir confirmar que una persona es quien dice ser. Normalmente, para constatar este hecho, el usuario usa algo que conoce para demostrar su identidad, como un usuario y una contraseña.

La **autorización** es el proceso de verificar lo que un usuario puede hacer. Por ejemplo, un usuario puede añadir canciones en una lista compartida de Spotify, pero no puede eliminar dicha lista. La autorización ocurre después de que un usuario se haya autenticado.

OAuth 2.0 y OIDC

OAuth 2.0 es el protocolo de **autorización** estándar de la industria. Se centra en la simplicidad del desarrollador del cliente y proporciona **flujos de autorización** específicos para aplicaciones web, aplicaciones de escritorio, teléfonos móviles y dispositivos de sala de estar.

OpenID Connect (OIDC) es un protocolo de **autenticación** interoperable basado OAuth 2.0.
Simplifica la forma de verificar la identidad de los usuarios en función de la autenticación realizada por un **servidor de autorización** y de obtener información del perfil del usuario de forma interoperable y similar a REST.

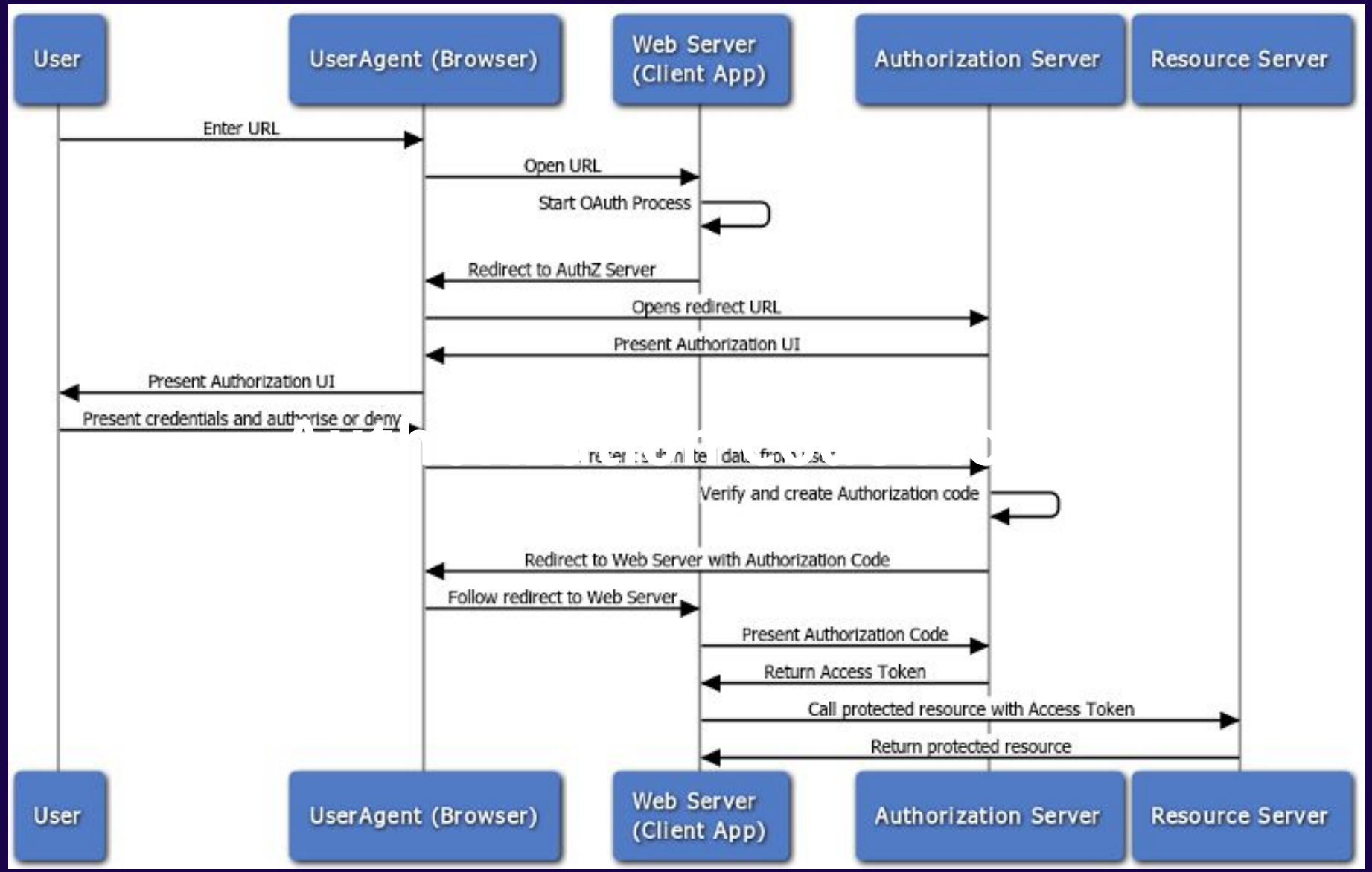


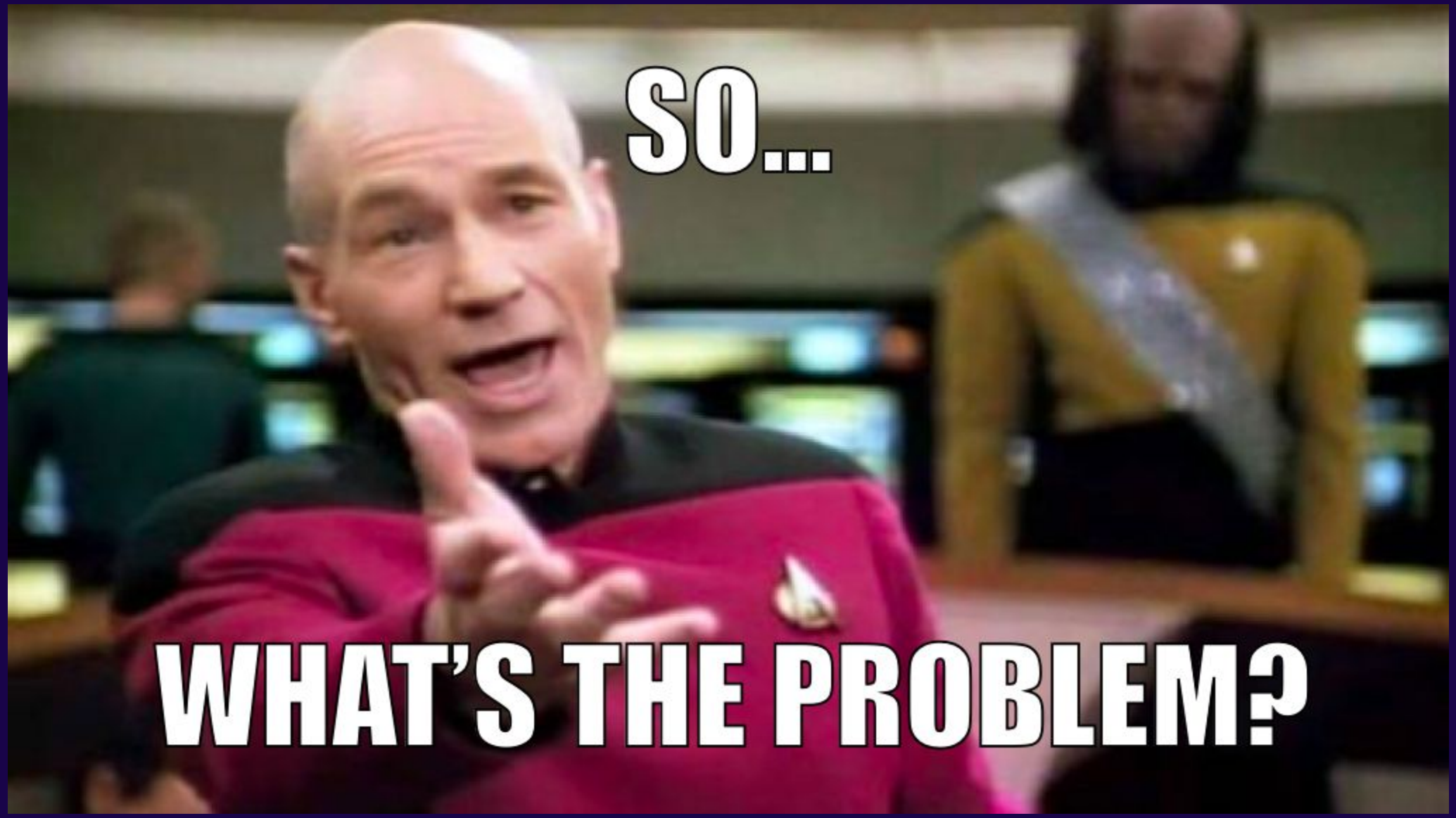
OAuth Grant Types

- Authorization Code
- PKCE
- Client Credentials
- Device Code
- Refresh Token

Legacy:

- Implicit Flow
- Password Grant


User
UserAgent (Browser)
Web Server (Client App)
Authorization Server
Resource Server
Enter URL
Open URL
Start OAuth Process
Redirect to AuthZ Server
Opens redirect URL
Present Authorization UI
Present Authorization UI
Present credentials and authorise or deny
Verify and create Authorization code
Redirect to Web Server with Authorization Code
Follow redirect to Web Server
Present Authorization Code
Return Access Token
Call protected resource with Access Token
Return protected resource
User
UserAgent (Browser)
Web Server (Client App)
Authorization Server
Resource Server


SO...
WHAT'S THE PROBLEM?

¿Cual es el problema con este flujo?

Ejemplo de solicitud inicial al auth server

```
https://your-authorization-server.com/authorize?
    client_id=1234567890&
    redirect_uri=https://your-app/callback&
    scope=openid profile email phone address&
    response_type=code&
    audience=https://myapi&
    state=12c1d32c1aba2cc14f9b907a9ec90
```

La solicitud de autorización no es más que una solicitud **GET HTTP** contra una URL con sus parámetros.

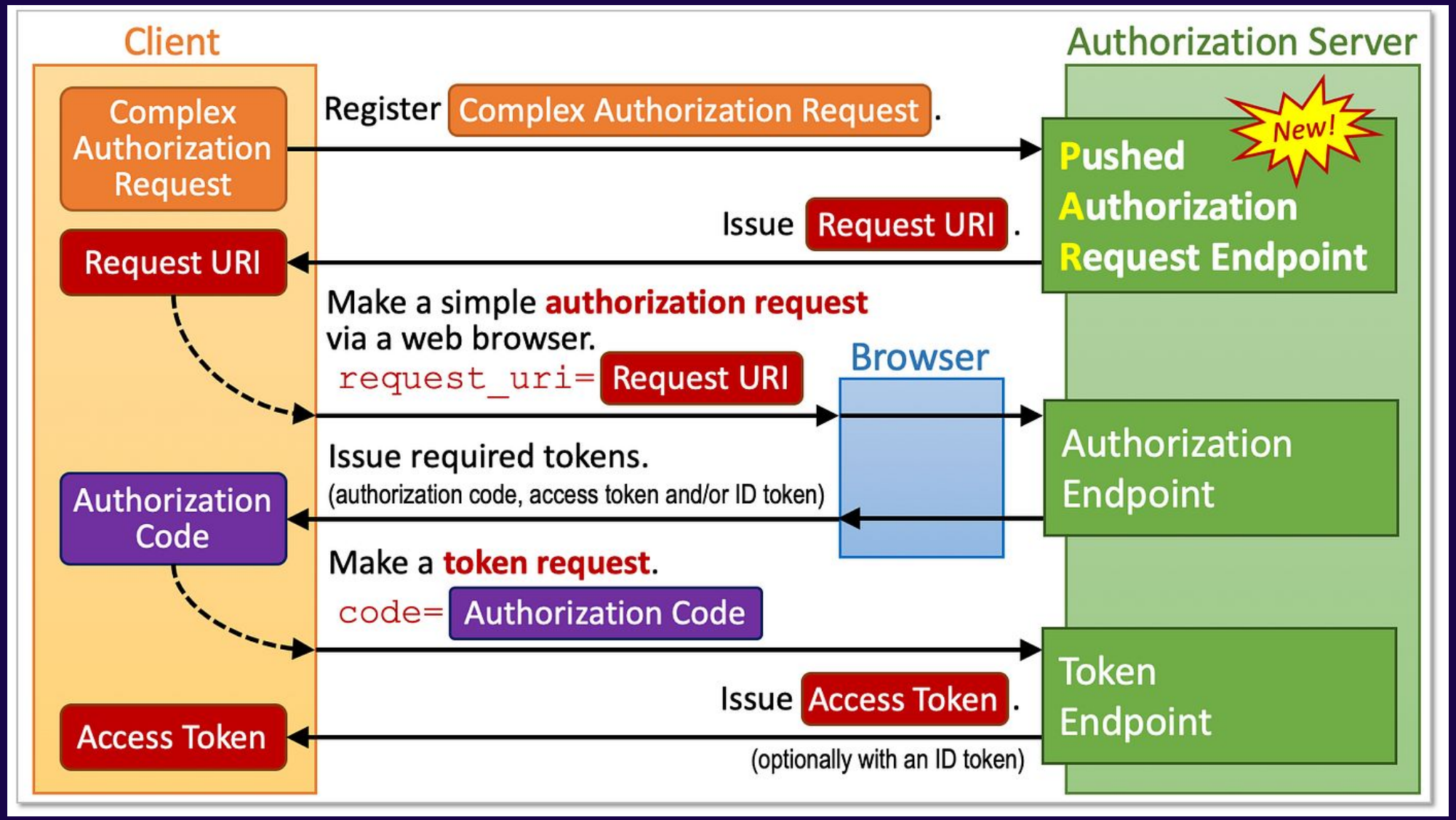
- **La solicitud de autorización puede modificarse.**
- **No hay garantía de la procedencia de la solicitud.**
- **No hay garantía de confidencialidad.**
- **Limitaciones del navegador.**

PAR

Pushed Authorization Request

- **Las solicitudes de autorización push (PAR)** son un estándar de OAuth relativamente nuevo que mejora la seguridad de los flujos de OAuth y OIDC al mover los parámetros de autorización del front channel al back channel. Es decir, mover los parámetros de autorización desde las URL de redireccionamiento en el navegador a llamadas http de máquina a máquina en el back-end.

- Esto evita que un atacante en el navegador pueda:
  - Ver parámetros de autorización, que podrían filtrar PII.
  - Manipular esos parámetros. Por ejemplo, el atacante podría cambiar el scope del acceso solicitado.


Client
Authorization Server
Complex Authorization Request
Register Complex Authorization Request .
New!
Pushed Authorization Request Endpoint
Issue Request URI .
Request URI
Make a simple authorization request via a web browser.
request_uri= Request URI
Browser
Authorization Endpoint
Issue required tokens.
(authorization code, access token and/or ID token)
Authorization Code
Make a token request.
code= Authorization Code
Token Endpoint
Issue Access Token .
Access Token
(optionally with an ID token)

Ventajas de usar PAR

- **Seguridad mejorada:**
  Dado que los parámetros de autorización se envían a través de una solicitud segura de canal secundario, ya no están expuestos en las URL del navegador, lo que reduce el riesgo de que se filtre información confidencial (como información de identificación personal o PII).

- **Resistencia a la manipulación:**
  Al enviar solicitudes de autorización a través del back-end, se evita que los atacantes modifiquen los parámetros, como los alcances solicitados o los niveles de acceso.

- **URL más cortas**
  En escenarios complejos de OAuth/OIDC (como cuando se utilizan solicitudes de autorización enriquecidas, las URL pueden volverse excesivamente largas, lo que puede causar problemas con algunos navegadores y sistemas de red.

Soporte para PAR en .net 9

Agregado por **Joe DeCock** de **Duende Software** (evolución de Identity Server) como una opción más en el **OpenIdConnectHandler**, el cual es un handler que funciona dentro del middleware de Autenticación de .net y se utiliza para implementar la autenticación basada en OpenID Connect (OIDC)

```
builder.Services
    .AddAuthentication(options =>
    {
        options.DefaultScheme = CookieAuthenticationDefaults.AuthenticationScheme;
        options.DefaultChallengeScheme = OpenIdConnectDefaults.AuthenticationScheme;
    })
    .AddCookie()
    .AddOpenIdConnect("oidc", oidcOptions =>
    {
        // Other provider-specific configuration goes here.

        // The default value is PushedAuthorizationBehavior.UseIfAvailable.

        // 'OpenIdConnectOptions' does not contain a definition for 'PushedAuthorizationBehavior'
        // and no accessible extension method 'PushedAuthorizationBehavior' accepting a first argument
        // of type 'OpenIdConnectOptions' could be found
        oidcOptions.PushedAuthorizationBehavior = PushedAuthorizationBehavior.Disable;
    });
```

Ejemplo práctico

Muchas gracias