Securing Servers Using OIDC and Keycloak

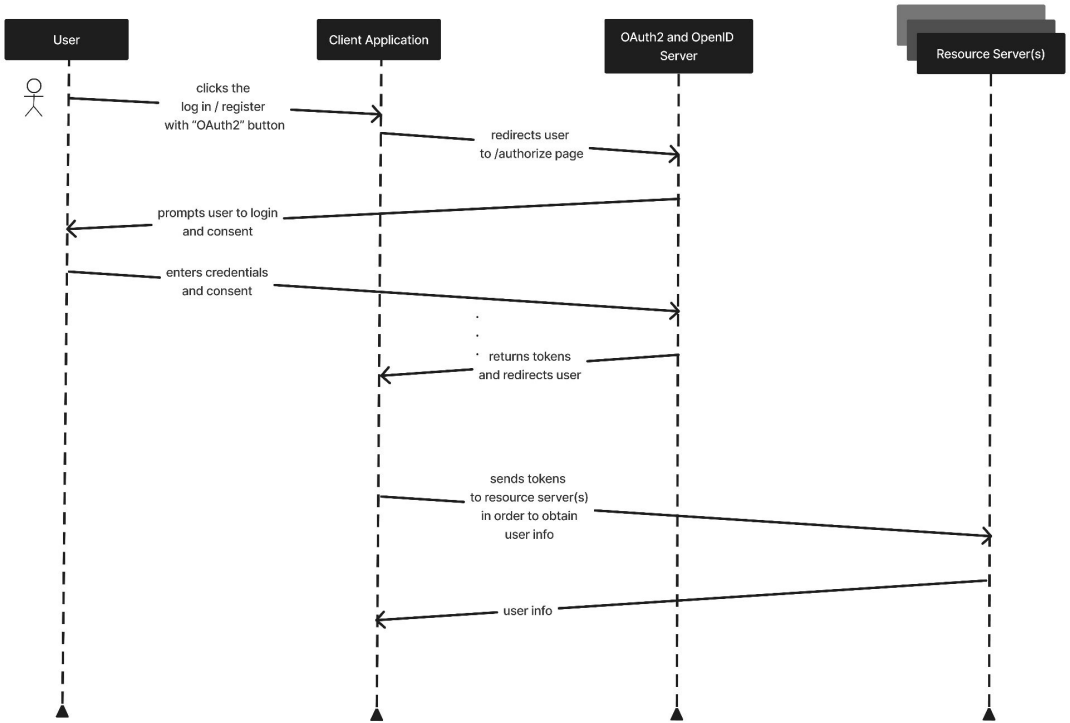
- Overview of OpenID Connect
- Introduction to Keycloak
- Securing a demo application

stzagkarak 19/02/24

What is OAuth 2.0 ?

- An authorization framework that enables applications to access **user resources** from **resource servers** over HTTP.
- Actors:
  - Resource owner: **Users** (real people)
  - **OAuth2 clients** (The application that wants to use oauth2 to retrieve information about its users)
  - **Resource Servers** (A server holding sensitive user information)
  - **Authorization Server** (A server capable of providing access tokens that can be consumed by resource servers)

OAuth Flow (over-simplified)



What is OpenID Connect ?

- An **extension of the OAuth2 framework** used in order **authenticate a user**, and **receive information about the user's identity** and session.
- Essentially, **it is a token that contains user data** that the client application can immediately use (**no need to access a resource server**). Such data can be **used** by the application in order **to apply custom logic** (e.g display the username on a site, create an authenticated session between provider and application).

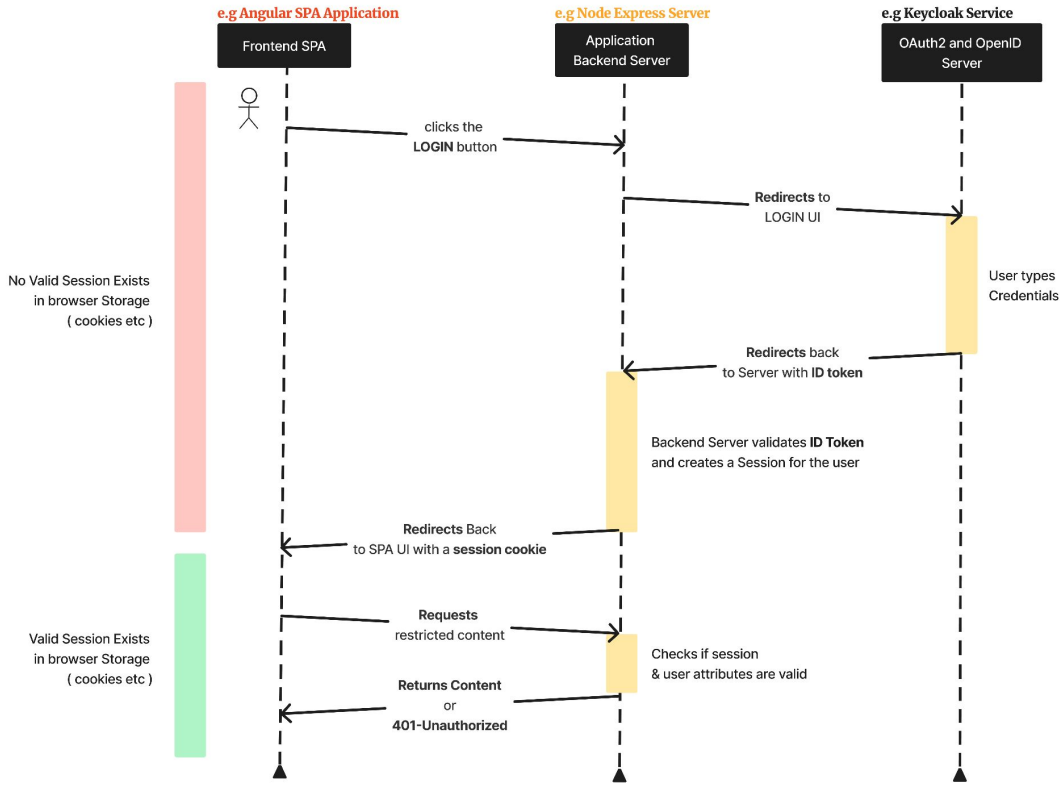
What is Keycloak ?

- An open-source **standalone authentication service** that provides **Single Sign-On**, **Identity** and Access **Management**.
- Supports **OAuth2**, **OpenID Connect** and SAML protocols.
- Can easily be managed and configured using the **admin dashboard**.
- In our discussion, **keycloak provides ID tokens** to requesting **client applications**. Those applications can be numerous and may have the same user base.

Securing SPA Applications with OIDC and Keycloak

- There is **NO SECURE** way of **DIRECTLY** **acquiring and storing ID Tokens in SPA applications**.
- That's why **acquire and store** the **ID tokens** in our **backend server(s)** and **create secure sessions between our backend(s) and SPA application(s)**.
- **We require valid sessions in order to access backend resources**. (e.g letting a client create a post)

Acquiring ID Tokens from an OpenID Server and creating a Secure Session



Demo time

- We will deploy keycloak locally. Using the admin dashboard, we will create a client application and then link it with our backend server. Then, using our SPA client, we will initiate Login actions in order to create secure sessions.
- You can replicate the demo using this repo: https://github.com/stzagkarak/angular-express-keycloak-demo.git

Understanding Keycloak Realms & Clients

- In keycloak, **a realm manages a set of users**, credentials, roles, and groups. **A user belongs to and logs into a realm**. **Realms are isolated** from one another and can only manage and authenticate the users that they control.
- **Inside realms we can create clients**. Each client has access to the users that are inside that realm.

In Production

- You **MUST** change the keycloak configuration for production environments. This repo contains a production-ready configuration that you can easily tweak and deploy in a web server.
    - *Note that the repo is not fully documented yet.*
- The instructions inside `deploying_in_production.md` and the appropriate deployment file may give you an example of how to deploy a full stack application securely in production.

Further Reading

Please study the provided code and read through the provided presentation.
Additionally, I highly encourage you to check the below resources (random order):

1. https://medium.com/@prashantramnyc/node-js-with-passport-authentication-simplified-76ca65ee91e5
2. https://medium.com/keycloak/keycloak-express-openid-client-fabea857f11f
3. https://dev.to/zachgoll/the-ultimate-guide-to-passport-js-k2l
4. https://dev.to/cristain/how-to-set-up-typescript-with-nodejs-and-express-2023-gf
5. https://medium.com/@ramanamuttana/custom-attribute-in-keycloak-access-token-831b4be7384a