Alors bonjour à tous, je vais d'abord vous parler du protocole Open Id Connect et on va ensuite l'implémenter en direct avec Keycloak et php.

Tout d'abord, Open Id Connect qu'est ce que c'est ?

C'est un protocole d'authentification unique dit SSO.

Du coup, aujourd'hui, on va plus parler de sécurisation des applications que de sécurisation d'infrastructure.

L'authentification unique est, comme son nom l'indique, une méthode permettant à un utilisateur d'accéder à plusieurs applications informatiques en ne procédant qu'à une seule authentification.

Concrètement, c'est ça, on peut se connecter à Deezer grâce à Google qui implémente un provider d'identification Open Id Connect

Et aujourd'hui Oauth2 est le SSO le plus utilisé.
Alors pourquoi je vous parle d' Open Id Connect et pas d'Oauth2 ?

Tout simplement parce qu'Oauth2 n'est pas normalisé, il y a autant d'implémentation d'Oauth2 que de provider d'identification et c'est là qu'intervient Open Id Connect.
Click.

Open Id Connect normalise et ajoute une couche d'identification à Oauth2.

C'est à dire qu'entre 2 provider Open Id Connect vous ne changez pas le client tandis que pour 2 provider Oauth2, il y a 2 client différent.

Mais, Comment fonctionne Open Id Connect dans les fait ?

Ca fonctionne avec trois acteurs.
Le client, représenté ici par le navigateur web en gros le front end.
Le Service Provider, c'est le serveur qui possède le service auquel nous voulons accéder, ici Deezer
Et l'identity Provider, c'est le serveur qui possède la base de donnée d'utilisateurs/ mot de passe, ici Google, Apple ou Facebook

Le protocole fonctionne de la manière qui suit :

1) Le client demande un accès aux ressources du service provider
2) Le service provider redirige le Client vers l'identity provider avec un code autorisation de requête qui contient notamment les informations auxquelles le service provider souhaite accéder
3) Le client effectue un GET sur l'identity provider avec le code autorisation de requête
4) L'identity provider lui retourne une page de login
5) Le client rentre son nom d'utilisateur/mot de passe
6) L'identity provider donne un code d'autorisation au client
7) Le client demande un accès aux ressources du service provider avec le code d'autorisation
8)Le service provider et l'identity provider s'échange respectivement le code d'autorisation contre un access token et des informations sur le client
9) Le service provider donne accès aux ressources au client.

Niveau sécurité, les principales attaques sur ce protocole se font par falsification d'access token que ce soit sur le service provider pour avoir accès aux ressources sans s'authentifier ou sur l'identity provider pour obtenir l'accès à d'autre service provider ou pour obtenir une élévation de privilège.

Nous allons passer à la partie démo

Keycloak est un logiciel libre fait par Red Hat il y a d'autre identity provider comme microsoft azure, google identity ou auth0 dans le libre

Keycloak fonctionne de la manière suivante :

Chaque realm ou royaume possèdent leur base de donnée d'utilisateurs, du coup si vous avez plusieurs entreprise clientes dans vos entreprises elles seront toute administré à partir de la même page, dans mon entreprise c'est comme ça qu'on fonctionne.

Les Clients représentes les différents Service providers du Realm

Les roles sont fait pour attribuer différent droit aux utilisateurs et accès aux services providers

Et enfin les users ce sont utilisateurs des différents services providers

Le Realm Master est fait pour administrer Keycloak

On va créer un nouveau royaume pour notre application

Keycloak est entièrement paramétrable de A à Z ici par exemple on peut modifier le thème de la page de connection ainsi que celle de l'administration, on peut aussi ajouter d'autre langue, mais ce n'est pas ce qui nous intéresse aujourd'hui

on va juste ajouter l'enregistrement d'utilisateur

On créé un client qui représenteras notre application

Et voila c'est fini pour la partie identity provider

C'est à dire que maintenant on a un serveur d'identité

Maintenant on va paramétrer le service provider from scratch pour voir toutes les étapes du protocole.

Nous avons 2 pages, index.php et connect.php

index.php est la page où il y a un lien pour se connecter

Et connect.php et la page auquelle on peut accéder une fois connecté.

Le but de notre application sera de voir une image de panda en se connectant.

On commence par les étapes 1 et 2

1) Le client demande un accès aux ressources du service provider
2) Le service provider redirige le Client vers l'identity provider avec un code autorisation de requête qui contient notamment les informations auxquelles le service provider souhaite accéder

ici la redirection vers l'identity provider avec le type de réponse qu'on attend, le client-id, les informations qui nous intéressent ainsi que l'url de redirection

et voilà nous avons dans l'url le code d'authorisation

on l'affiche à l'écran

nous sommes donc à l'étape 6

6) L'identity provider donne un code d'autorisation au client

Et donc on va échanger cette variable auprès de keycloak pour avoir un access token.

c'est à dire l'étape 7

On fait donc un curl vers keycloak auquel on passe en paramètre le code ainsi qu'une url de redirection

On récupère la réponse et si tout se passe bien on l'affiche

On récupère ici un access token que l'on va utiliser pour récupérer les donnée de l'utilisateur

Ici on peut voir l'access token, sa durée avant expiration

Un refresh token avec sa durée avant expiration

et ici les information que l'on veut récupérer auprès de l'identity provider

On va récupérer donc les récupérer, c'est l'étape 9

On fait donc un curl auquel on passe en paramètre l' access token ainsi qu'une url de redirection

On récupère la réponse et si tout se passe bien on l'affiche

Et Enfin on affiche l'image de panda

on voit que l'email n'est pas vérifié , le nom de l'utilisateur son username, ainsi que son email.

Et voila en moins de 10 minutes on a sécuriser une application grâce à oidc en partant de rien.

Je vais maintenant vous parler des access token ou JWT

on va donc récupérer le token et le décoder.

Un jwt est composé de 3 partie :

Un header, utilisé pour décrire le jeton. Il s'agit d'un objet JSON.

Un payload qui représente les informations embarquées dans le jeton. Il s'agit également d'un objet JSON.

Une *signature* numérique.

Pour obtenir la signature, on encode séparément le header et le payload en base 64. Ensuite, on les concatène ensemble en les séparant avec un point. On obtient la signature de ce résultat avec l'algorithme choisi. Cette signature est ajoutée au résultat de la même manière

C’est donc des information très peu sécurisée auxquelles on ne peut pas faire confiance. Ainsi si le service provider ou l’identity provider sont mal programmé on peut très vite faire des dégats.

Il y a 4 challenge sur root me pour ces failles de sécurités.

Merci de votre attention, je suis disponible pour répondre à vos questions.