Etude des services d'authentification

FDI-LEEX

Période d'expérimentation :

Date de début : Date de fin :

Membres du groupe :

- **feldma_l / dev**
- **places_m / devops**

Contexte d'expérimentation :

Architecture Microservices sur Kubernetes (backend) et 2 sites clients (une boutique et un site applicatif). Infrastructure utilisée dans le cadre du GPE de places_m & feldma_l en IOT. Un focus particulier a été voulu sur la sécurité du produit web car affichant des informations sensibles (notamment de la géolocalisation).

Objet du protocole :

Comparaison de protocoles d'authentification OAuth2 proposés par différents Cloud providers et d'une solution auto-hébergée pour sécuriser des microservices de manière robuste et fiable.

Objectifs détaillés :

L'objectif est de mettre en place certaines des solutions les plus répandues pour avoir un réel retour d'expérience et nous permettre de choisir au mieux celle qui nous convient.
Les différentes comparaisons mises en place porteront principalement sur la facilité de mise en place, les performances en conditions réelles et le coût. La facilité de mise en place et d'administration de chaque solution sera aussi un critère non négligeable pour le choix final de la solution.

Environnement de test :

- Protocole OAuth 2.0 sur une application React Js
- Google, Microsoft (et Keycloak hébergé sur notre cluster)
- Test automatisé en Python avec Sélénium

Protocole d'expérimentation :

Produire 2 POC (le POC keycloak n'ayant pas pu être réalisé à temps) : 2 applications React JS utilisant respectivement les librairies et services SSO de Microsoft Azure AD B2C et de Google Cloud Platform (fia Firebase).
Nous discuterons de la facilité de mise en place du SSO au sein des applications, de l'administration des services SSO, de la modularité des différentes solutions, des coûts des infrastructures associées.
Nous réaliserons des mini "stress tests" pour évaluer les performances à l'aide de Sélénium et d'un script Python. Le script ouvrira les applications un certain nombre de fois et y réalisera une authentification afin de calculer un temps moyen pour une authentification automatique.

Mise en place de l'infrastructure :

La création du cluster hébergeant nos applications ne rentre pas dans le sujet. Nous ne détaillerons pas sa mise en place ni le déploiement des applications clientes.

Azure AD B2C :

1. Création d'un locataire Azure
2. Aller sur le portail Azure AD B2C et inscrire une application (http://localhost:3000 pour le développement local et https://<nom de domaine de l'application> pour l'application déployée)
3. Créer le flux utilisateur (Sign In and Register pour permettre à l'utilisateur de s'enregistrer lui-même)
4. Renseigner l'url de redirection pour l'application (http://<nom de domaine de l'application>/authentication/login-callback)
5. Récupérer les données de configuration fournies pour l'application (ClientId, Authority, …)

Firebase Google Cloud Platform :

1. Créer un instance Firebase Authentication via Google Cloud Platform
2. Créer une instance de Firestore Database sur la console Firebase
3. Inscrire une application dans les paramètres de la console Firebase
4. Récupérer les données de configuration fournies pour l'application (apiKey, appId, …)

Keycloak

1. Déployer une instance de Keycloak et une base de donnée (ici PostgreSQL)
2. Se connecter à la console d'administration
3. Créer un "Realm" dédié aux applications clientes (par exemple "my-realm") et autoriser les utilisateurs à s'enregistrer via l'onglet "Login"
4. Créer un client et renseigner les url nécessaires
5. Récupérer les informations dans l'onglet "Installation" (clientId, auth-url, …)

Implémentation du SSO dans une application cliente :

Nous utiliserons au maximum les librairies officielles fournies afin de déployer une application React Js pour chaque solution.

Azure AD B2C :

Librairie : @azure/msal-browser et @azure/msal-react

Implémentation : Les librairies nous permettent de nous loguer via une méthode de redirection ou une méthode de PopUp. En plus de ces méthodes, les librairies nous fournissent des composants/templates permettant d'afficher ou non du contenu si l'utilisateur est authentifié.

Firebase Google Platform :

Librairie : firebase et react-firebase-hooks

Implémentation : Les librairies nous fournissent différentes méthodes que nous pouvons utiliser pour effectuer des appels vers le serveur d'authentification. Nous avons dû développer le formulaire de connexion ainsi que le formulaire d'inscription de l'utilisateur.

Keycloak :

Librairie : keycloak-js

Implémentation : La librairie est peu documentée pour une application React Js. Nous n'avons pas eu le temps d'expérimenter afin d'arriver à un résultat convenable.

Documentation :

Concernant le systèmes de single sign_on :

https://fr.wikipedia.org/wiki/Authentification_unique

À propos des différents systèmes à tester et à mettre en place :

https://oauth.net/

https://oauth.net/code/python/

https://blog.authlib.org/2020/fastapi-google-login

https://www.keycloak.org/

https://developers.google.com/identity/protocols/oauth2

https://docs.microsoft.com/fr-fr/azure/active-directory/develop/active-directory-v2-protocols

Performance mesurée des différents systèmes :

À l'aide d'un script de test automatique utilisant Selenium pour Python, nous avons simulé l'utilisation des différentes pages d'authentification. Sur chacune, notre test se connectait 100 fois et calculait le temps pris pour être redirigé vers le service voulu. Voici les valeurs mesurées.

	Moyenne de temps (ms)	Temps le plus faible (ms)	Temps le plus long (ms)
Microsoft Azure Active Directory	1663	2165	1404
Firebase Google Cloud Platform	936	1268	713

En analysant, ces chiffres semblent peu représentatifs. La majorité du temps est prise par l'utilisateur pour rentrer ses informations de connexion. Passer de 0.9 à 1.6 secondes ne semble pas important et que peu impactant, nous avons donc voulu écarter cette composante de notre prise de décision finale.

Prix :

Azure AD B2C :

Le service est gratuit pour moins de 50k utilisateurs par mois.
Au delà de 50k utilisateurs par mois : 0,002925 € par utilisateurs actifs mensuels. Prix max : 1681875€ par mois.

Firebase Google Cloud Platform :

Gratuit pour moins de 25k utilisateurs par mois.
Au delà de 25k utilisateurs par mois : pas de prix fixé par utilisateurs actifs mensuels. Prix max : 1157,80€ par mois.

Keycloak :

Le prix dépend du nombre d'instances de Keycloak à déployer. Plus nous avons d'instances d'applications utilisant le SSO, plus nous devons déployer d'instances de Keycloak et augmenter la taille et le nombre de réplicats de la base de données.

Conclusion :

Après prise en compte des éléments impactant, nous avons pu faire un choix, et ce sera Microsoft Azure Active Directory. Voici les points qui ont motivé notre décision :

- Le fait que la première installation soit fastidieuse n'était pas important pour nous. La configuration initiale étant faite, elle n'est plus à faire. Cela ne rentre donc pas en compte dans la problématique de choix pour notre GPE.
- Après l'avoir fait une première fois, nous savons déployer une nouvelle application et il sera donc simple de continuer dans cette voie.
- Malgré des performances quasiment deux fois meilleures pour GCP, nous ne prenons pas en compte ce paramètre car il n'est que trop peu changeant d'un système à l'autre.
- Pour le prix, MAAD étant gratuit pour les 50k premiers utilisateurs, cette solution est pour le moment bien suffisante dans notre cas de figure.

Finalement et en ayant pesé le pour et le contre de chaque solution, nous pensons que MAAD est plus adapté et plus pratique à utiliser dans notre situation.