

# ITÉRATION 6

## Le protocole OAuth 2

### Modalités

- Travail individuel en autonomie
- 2 jours en présentiel

### Objectifs

Comprendre les fondamentaux de la sécurité des intégrations d'applications tierces avec des APIs en utilisant le protocole OAuth 2. Ceci comprend plusieurs types d'applications: Web, Mobiles/Natives, Embarquées (TV, terminaux, etc), et/ou des applications Back Office. Nous apprenons les différentes manières d'utiliser le protocole OAuth2 pour sécuriser l'authentification et les autorisations de ces multitudes d'applications avec des APIs protégées. Nous pratiquons sur des systèmes existants et nous développons aussi des requêtes simulant de vraies applications.

### Compétences

- S'assurer que les transactions sont sécurisées

#### Livrables

Pour cette itération, et afin de valider la seule compétence (ci-avant citée), vous devez compléter et retourner au formateur le document joint **"Cybersécurité\_Kit-6-Exercices"** complété avec vos réponses et solutions aux différents exercices.

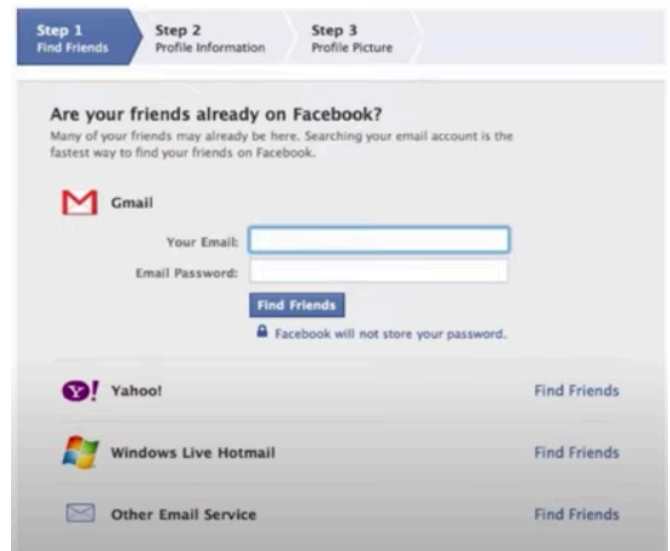
## 1.1 — Pourquoi OAuth 2 ?

1h — Présentiel

Pour vous expliquer pourquoi OAuth 2 est nécessaire, revenons un peu en arrière. Imaginons que vous souhaitiez vous inscrire sur Facebook. À un certain moment de l'enregistrement, on vous demande votre compte et votre mot de passe Gmail pour récupérer la liste de vos contacts. Facebook cherche alors à savoir si l'un d'entre eux est déjà membre de son réseau social, afin d'établir un lien entre les deux comptes.

*C'est quoi le problème majeur de cette fonctionnalité ?*

Vous l'aurez compris, cette méthode de connexion est risquée, car elle implique de fournir à Facebook vos identifiants Gmail, qui pourraient être utilisés à des fins malveillantes. C'est pourquoi OAuth 2 a été mis en place : il permet à Facebook de récupérer les informations dont il a besoin sans avoir accès à votre mot de passe Gmail, garantissant ainsi la sécurité de votre compte.



C'est pourquoi nous étudions ce protocole très largement utilisé. Vous aurez probablement l'occasion, dans votre parcours DevOps, de vous intégrer avec une API externe et le protocole à utiliser sera OAuth2.

Commencez par voir la vidéo suivante (dans les ressources ci-après) pour avoir une toute première compréhension de ce protocole. Répondez ensuite aux questions 1.1 dans le document d'exercices.

### RESSOURCES

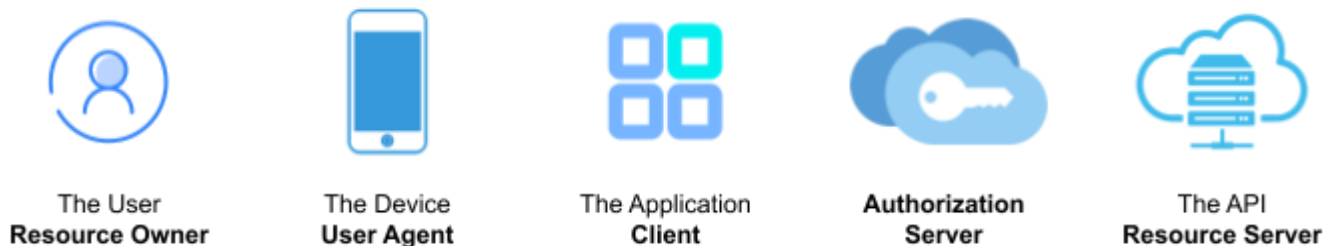
- [1] Qu'est-ce qu'OAuth et pourquoi est-ce important ?  
<https://www.youtube.com/watch?v=KT8ybowdyr0>
- [2] Le document d'exercices à cloner et compléter. Commencez à répondre aux questions de la section 1.1 pour le moment.

## 1.2 – Les rôles OAuth 2

1h – Présentiel

OAuth2 est un protocole d'autorisation qui permet à des utilisateurs de donner l'accès à leurs données à des tiers de manière sécurisée. Dans OAuth2, il existe différents rôles qui jouent un rôle important dans le processus d'autorisation:

1. Tout d'abord, il y a le propriétaire de la ressource (Resource Owner), qui est l'utilisateur final qui possède les données qu'il souhaite partager avec des tiers.
2. Ensuite, il y a le client, qui est l'application tierce qui souhaite accéder à ces données.
3. L'application tourne dans un équipement (User Agent) qui peut être le navigateur Web ou le téléphone portable de l'utilisateur.
4. Le serveur d'autorisation (Authorization Server) est le service qui autorise le client à accéder aux données de l'utilisateur.
5. Enfin, il y a le serveur de ressources (Resource Server, ou l'API), qui fournit les données auxquelles le client a été autorisé à accéder.



Comprendre les différents rôles dans OAuth2 est essentiel pour comprendre comment fonctionne le processus d'autorisation et comment les données sont partagées en toute sécurité entre les différents acteurs impliqués.

Continuez votre compréhension en lisant l'article suivant (dans les ressources ci-après). Répondez ensuite aux questions 1.2 dans le document d'exercices.

### RESSOURCES

- OAuth 2.0 Roles : <https://medium.com/@greekykhs/ro-acd8cb4cc0f4>

## 1.3 – Les types d'applications (clients)

1h – Présentiel

Il existe deux types d'applications (Clients) dans OAuth 2: les clients confidentiels et les clients publics.

- Les clients confidentiels (**Confidential Clients**) sont des applications qui peuvent protéger de manière fiable leurs informations d'identification client, telles que les clés d'API et les secrets partagés, contre une exposition accidentelle ou malveillante. Ces applications sont

souvent des serveurs web sécurisés qui peuvent stocker en toute sécurité des informations d'identification client et qui communiquent directement avec le fournisseur d'authentification. Les clients confidentiels sont souvent utilisés pour les applications commerciales ou à haute sécurité, telles que les services bancaires en ligne ou les services gouvernementaux.

- Les clients publics (**Public Clients**), en revanche, sont des applications qui ne peuvent pas garantir la protection de leurs informations d'identification client contre une exposition accidentelle ou malveillante. Ces applications sont souvent des applications mobiles ou des scripts côté client qui sont exécutés sur des navigateurs Web. Les clients publics ne stockent pas les informations d'identification client de manière sécurisée et doivent donc utiliser des mécanismes de redirection pour communiquer avec le fournisseur d'authentification. Les clients publics sont souvent utilisés pour les applications grand public, telles que les réseaux sociaux et les applications de messagerie instantanée.

En comprenant la différence entre les clients confidentiels et les clients publics dans OAuth 2, les développeurs peuvent choisir le type de client approprié pour leur application en fonction de la sécurité et des fonctionnalités requises. Ils peuvent également mettre en place des mesures de sécurité supplémentaires pour les clients publics afin de réduire les risques de sécurité associés à l'utilisation d'un flux d'autorisation moins sécurisé.

**À faire :** complétez la section 1.3 du document d'exercices en classifiant plusieurs exemples d'applications en tant que clients confidentiels ou clients publics.

---

## 1.4 – Enregistrement des Applications (Clients OAuth2)

1h – Présentiel

Avant de commencer le processus OAuth, vous devez d'abord enregistrer une nouvelle application auprès du service. Lors de l'enregistrement d'une nouvelle application, vous enregistrez généralement des informations de base telles que le nom de l'application, le site web, un logo, etc. De plus, vous devez enregistrer un URI de redirection vers l'application afin de transmettre des informations nécessaires pour continuer le flux d'autorisation. Comme résultat d'enregistrement de l'application, vous aurez un identifiant unique généré pour l'application (Client\_ID) et éventuellement un secret (Client\_Secret). L'ID client est considéré comme une information publique et est utilisé pour créer des URLs de connexion, ou inclus dans le code source Javascript d'une page. Le secret client doit être gardé confidentiel. Si une application déployée ne peut pas garder le secret confidentiel, comme les applications Javascript monopage ou les applications natives, alors le secret n'est pas utilisé et idéalement le service ne devrait pas émettre de secret pour ces types d'applications en premier lieu.

**Exercice 1:**

Auth0.com est un service générique qui fournit un serveur d'authentification et d'autorisation pour nos applications. Dans cet exercice, vous allez vous inscrire pour obtenir un compte développeur sur [Auth0.com](https://auth0.com) (Authorization Server) et enregistrer une application de type "Client Confidentiel" (Application Web serveur par exemple). Notez sur votre document de réponse aux exercices (section 1.4) la preuve de création de cette application OAuth2 dans Auth0 (il n'est pas question ici de coder!).

**Exercice 2:**

Dans ce deuxième exercice nous allons plutôt essayer de s'intégrer avec un serveur d'autorisation existant qui protège des ressources de la même société. Nous expérimentons avec Dropbox qui est un service de stockage de fichiers en ligne qui fournit une API utilisable par des applications tierces. Créer un compte gratuit dans [Dropbox.com](https://dropbox.com) et enregistrer une application fictive dans la console développeur de Dropbox <https://www.dropbox.com/developers/apps> (le Guide complet par ici <https://developers.dropbox.com/oauth-guide>).

## 1.5 – Flux d'autorisations

3h – Présentiel

Commencer par voir la vidéo suivante: <https://www.youtube.com/watch?v=CPbvxxsIDTU>

OAuth 2.0 fournit plusieurs flux d'autorisation pour différents cas d'utilisation.

1. **Client Credentials:** pour les applications qui [ne nécessitent pas l'autorisation](#) d'un utilisateur mais interrogent l'API en tant que telles.
2. **Password:** pour autoriser des applications du même éditeur que le serveur d'autorisation (1st party applications) en utilisant [le compte et mot de passe](#) de l'utilisateur directement.
3. **Authorization Code:** pour des applications s'exécutant dans des [serveurs web](#), dans [le navigateur](#), ou des [applications mobiles](#).
4. **Implicit:** cela était précédemment recommandé pour les clients/applications publics (qui ne peuvent pas garantir la confidentialité du Client\_Secret), mais a été remplacé par l'utilisation de l'**Authorization Code avec PKCE**.

### 1. Client Credentials Grant flow

Dans certains cas, les applications peuvent avoir besoin d'un jeton d'accès pour agir en leur propre nom plutôt qu'au nom d'un utilisateur. Par exemple, le service peut fournir un moyen pour l'application de mettre à jour ses propres informations telles que son URL de site Web ou son icône, ou elle peut souhaiter obtenir des statistiques sur les utilisateurs de l'application. Dans ce

cas, les applications ont besoin d'un moyen d'obtenir un jeton d'accès pour leur propre compte, en dehors du contexte de tout utilisateur spécifique. OAuth fournit le type d'autorisation "**client\_credentials**" à cette fin. Pour l'utiliser, effectuez une requête POST comme suit :

```
POST https://api.authorization-server.com/token
grant_type=client_credentials&
client_id=CLIENT_ID&
client_secret=CLIENT_SECRET
```

Plus de détail par ici: <https://www.oauth.com/oauth2-servers/access-tokens/client-credentials/>

## 2. Flux de mot de passe (Password flow)

OAuth 2 propose également un type d'autorisation qui peut être utilisé pour échanger directement un nom d'utilisateur et un mot de passe contre un jeton d'accès. Comme cela nécessite évidemment que l'application collecte le mot de passe de l'utilisateur, il ne doit être utilisé que par des applications créées par le service lui-même. Par exemple, l'application Twitter native pourrait utiliser ce type d'autorisation pour se connecter sur des applications mobiles ou de bureau.

Pour utiliser ce type d'autorisation par mot de passe, il suffit de faire une requête POST comme suit:

```
POST https://api.authorization-server.com/token
grant_type=password&
username=USERNAME&
password=PASSWORD&
client_id=CLIENT_ID
```

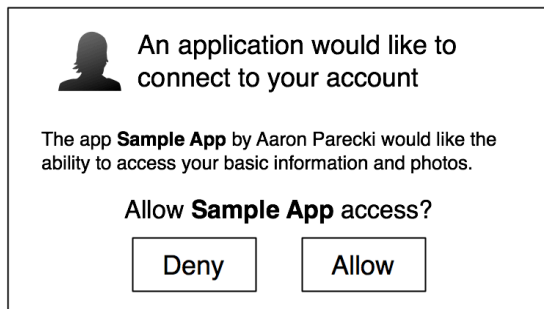
Plus de détail par ici: <https://www.oauth.com/oauth2-servers/access-tokens/password-grant/>

## 3. Flux d'octroi d'autorisation (Authorization Code Grant Flow)

Ce flux fonctionne permet à l'utilisateur de donner son consentement au client/application tierce pour accéder à ses ressources protégées, généralement en fournissant ses informations d'identification, telles qu'un nom d'utilisateur et un mot de passe sur une page d'authentification fournie par le serveur d'autorisation.

```
https://authorization-server.com/auth?response_type=code&
client_id=CLIENT_ID&redirect_uri=REDIRECT_URI&scope=photos&state=1234zyx
```

Le serveur d'autorisation vérifie les informations d'identification de l'utilisateur et, si elles sont valides, demande le consentement de l'utilisateur pour accorder l'accès aux ressources demandées. Si l'utilisateur donne son consentement, le serveur d'autorisation émet un jeton d'accès au client, qui peut ensuite être utilisé pour accéder aux ressources protégées.



```
https://example-app.com/cb?code=AUTH_CODE_HERE&state=1234zyx
```

Ce code d'autorisation peut être échangé contre un token d'accès qui sera quant à lui utilisé pour accéder à l'API.

```
POST https://api.authorization-server.com/token
grant_type=authorization_code&
code=AUTH_CODE_HERE&
redirect_uri=REDIRECT_URI&
client_id=CLIENT_ID&
client_secret=CLIENT_SECRET
```

On vous propose de s'entraîner sur des vrais exemples ci-après pour bien appréhender ces différents flux.

### Exercice 1:

Faire une première expérience dans le playground suivant :

<https://www.oauth.com/playground/index.html>

### Exercice 2:

ETAPE 1: préparer votre compte auth0.com pour l'exercice:

<https://oauth.school/exercise/introduction/>

ETAPE 2: faire l'exercice sur OAuth pour les applications web:

<https://oauth.school/exercise/web/>

### RESSOURCES

- <https://www.youtube.com/watch?v=CPbvxxslDTU>
- <https://zestedesavoir.com/articles/1616/comprendre-oauth-2-0-par-l'exemple/>

- <https://aaronparecki.com/oauth-2-simplified>


## 1.6 – Lab 1: Lister des fichiers dans Google Drive

2h – Présentiel

Google fournit un outil aux développeurs pour tester l'ensemble de ses APIs directement depuis une interface Web. Tous les APIs Google sont protégés par OAuth2.

Comme exercice, on vous demande d'utiliser cet outil pour pratiquer les différentes étapes du protocole OAuth2 afin d'appeler au final l'API de Google Drive et récupérer la liste de vos fichiers stockés.

<https://developers.google.com/oauthplayground>

Cherchez l'API  Drive API v3 dans la liste, et choisissez l'API

<https://www.googleapis.com/auth/drive.readonly>

Déclencher le flux d'autorisation OAuth jusqu'à l'obtention d'un JSON contenant la liste de vos fichiers dans Google Drive (utiliser un compte personnel et non pas celui du Campus numérique, ainsi créer quelques fichiers dans Drive pour avoir un résultat).

Notez le JSON de résultat dans votre fichier de réponse aux exercices (section 1.6)

## 1.7 – Lab 2: Intégration Travis CI et GitHub

2h – Présentiel

Travis CI est un service d'intégration continue hébergé, utilisé pour tester et construire des projets logiciels, ainsi que pour automatiser le déploiement de ces projets sur des plateformes d'hébergement et dans le cloud.

Dans cet exercice, vous allez intégrer Travis-CI (Client OAuth2) avec GitHub (Resource Server) en utilisant votre compte GitHub (GitHub comme Authorization Server).

1. Rendez-vous sur [Travis-ci.com](https://travis-ci.com) et inscrivez-vous avec GitHub.
2. Acceptez l'autorisation de Travis CI. Vous serez redirigé vers GitHub. Noter les paramètres passés dans l'URL vers GitHub depuis Travis. Noter aussi les scopes requis par Travis-CI. Pouvez-vous les noter et expliquer sur une fiche?
3. Cliquez sur votre photo de profil en haut à droite de votre tableau de bord Travis, cliquez sur Paramètres, puis sur le bouton Activer en vert, vous serez redirigé vers GitHub à nouveau pour activer votre application avec des options d'autorisation plus fines. Est ce que cette étape fait partie du protocole OAuth2?

## RESSOURCES

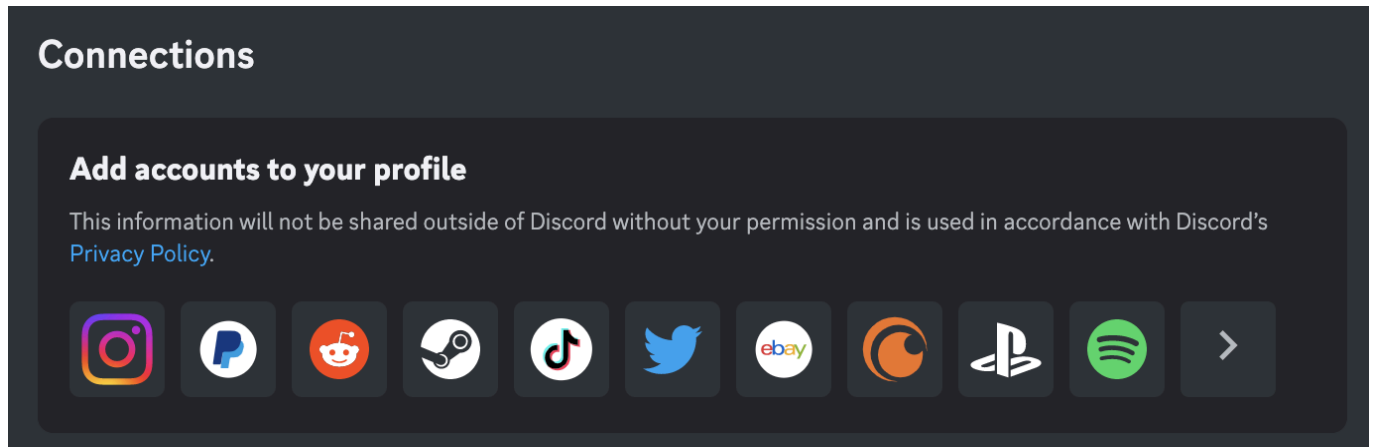


- <https://docs.travis-ci.com/user/tutorial#to-get-started-with-travis-ci-using-gitlab>

## 1.8 — Lab 3: Discord et ses connexions

1h — Présentiel

Discord support des intégrations avec d'autres outils externes via OAuth 2. Voici la liste des connexions à l'état actuelle:



Comme exercice, vous allez choisir une de ses connexions et analyser les différentes étapes et les paramètres passés dans l'URL.

### RESSOURCES

- Spotify connexion  
<https://support.discord.com/hc/en-us/articles/360000167212-Discord-Spotify-Connection>

## 1.9 — Lab 4: Implémenter soi-même une application cliente OAuth2 (optionnel)

2h — Présentiel

En utilisant le langage que vous maîtrisez le mieux, et vous choisissez le serveur d'autorisation et l'API que vous voulez (ils respectent tous le standard OAuth2).

*HINT:* Auth0.com génère des codes exemples pour plusieurs langages de programmation et pour les différents types d'applications supportées. Vous pouvez l'essayer !

## 1.10 – Lab 5: S'authentifier dans Grafana en utilisant notre compte GitLab (optionnel)

Un cas intéressant pour utiliser OAuth2 comme un moyen d'externaliser l'authentification.

Démarrer Grafana (avec Docker):

<https://grafana.com/docs/grafana/latest/setup-grafana/configure-docker/>

```
docker run -d \  
-p 80:3000 \  
--name=grafana \  
-e "GF_INSTALL_PLUGINS=grafana-clock-panel,grafana-simple-json-datasource" \  
grafana/grafana-oss
```

Enregistrer dans GitLab et Configurer Grafana

Config:

<https://grafana.com/docs/grafana/latest/setup-grafana/configure-grafana/#override-configuration-with-environment-variables>

GitLab OAuth2

<https://grafana.com/docs/grafana/latest/setup-grafana/configure-security/configure-authentication/gitlab/>

```
GF_<SectionName>_<KeyName>
```