Wstęp do OAuth2

OAuth 2.0 to otwarty protokół autoryzacji, który umożliwia aplikacjom na dostęp do danych użytkownika. Na początek, celem wprowadzenia, życiowy przykład: Jurek wyjeżdża na dłużej i potrzebuje kogoś, kto będzie podlewał mu kwiatki w domu i karmił rybki. Prosi więc koleżankę o przysługę i daje jej klucz do swojego domu. Jakie to rodzi problemy? Całkiem sporo:

1. Koleżanka może **zgubić klucz** do domu Jurka. Jeśli ktoś niepowołany znajdzie klucz, kłopoty gotowe. Nie można unieważnić klucza. Trzeba wymieniać zamki.

2. Ktoś może bez jej wiedzy **skopiować klucz**, będący w jej posiadaniu. Problemy jak w punkcie 1.

3. Posiadaczka klucza może być nieuczciwa i **sama skopiować sobie klucz**.

4. Oprócz dostępu do kwiatków i rybek koleżanka ma **dostęp do całego mieszkania**. Może spowodować inne szkody.

5. Nasza opiekunka rybek może **przyprowadzić kogoś znajomego**, nawet w dobrej wierze, ale ta trzecia osoba może narobić nam szkód.

6. Jeśli przekazaliśmy klucz np. przez kogoś, nie wiemy **czy dotarł do właściwej osoby**. Ponieważ mówimy tak naprawdę o sieci, wszystko może odbywać się zdalnie.

7. **Zmiana zamków,** gdy ktoś zgubił klucz unieważnia wszystkie klucze. Trzeba na nowo wykonać kopie kluczy i rozdać je uprawnionym osobom.

Kto jest kim w OAuth2, czyli trochę o **rolach**:

Właściciel zasobu (*resource owner*) to ktoś, kto rozporządza dostępem. W naszym przypadku jest to Jurek – właściciel mieszkania.
Klient (*client*) jest tym, który chce uzyskać dostęp do zasobu. W przykładzie to nasza koleżanka podlewająca kwiatki.
Zasób (*resource server*) to aplikacja lub serwis, do której klient chce uzyskać dostęp.
W powyższym przykładzie chodzi oczywiście o mieszkanie, a konkretnie kwiatki i rybki w nim obecne.
Serwer autoryzacji (*authorization server*) to aplikacja, która zarządza przydzielaniem kluczy, tokenów i tymczasowych kodów dostępu do zasobu oraz upewnia się, że właściwa osoba dostaje dostęp. W przykładzie z domem nie ma nikogo takiego. Moglibyśmy sobie wyobrazić portiera, ciecia i ochroniarza w jednej osobie. Taki portier sprawdzałby dokumenty każdej wchodzącej osoby i wpuszczał tylko tych, którzy mają prawo wejść.

Mamy też dwa **ważne** pojęcia, które pojawiają się w specyfikacji OAuth2:

Zgoda na dostęp (*authorisation grant*) to umowne potwierdzenie, że klient ma prawo dostępu do określonego zasobu. Zgoda ma formę fizyczną tzn jest kodem, dokumentem lub czymkolwiek co można wydać klientowi.
Token (*access token*) to fizyczny klucz, kod, karta dostępu lub inna forma pozwalająca faktycznie dostać się do wybranych fragmentów naszego zasobu.

Jak przebiega autoryzacja?

Przykład z dostępem do mieszkania nie zapewniał bezpieczeństwa. Jeśli pomyślimy o małej firmie mieszczącej się w nowoczesnym biurowcu będziemy bliżej OAuth2.

Szef firmy (właściciel zasobu) zatrudnia nowego **pracownika** (klienta). Pracownik po rozmowie kwalifikacyjnej podpisuje umowę. Szef decyduje w jakim dziale nowa osoba ma pracować, a więc do jakich **fragmentów zasobu** ma mieć dostęp.

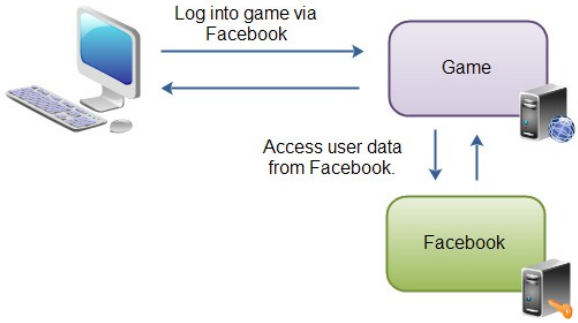
Pracownik z podpisaną umową idzie do **zarządcy** (serwera autoryzacji). Zarządca to ktoś, kto wydaje klucze lub karty dostępu do biur, przydziela numery szafek i inne ważne w firmie dostępy.

Pracownik pokazuje mu dokument **tożsamości i umowę** (dane autoryzacyjne). Po sprawdzeniu zarządca wydaje **kartę do drzwi** (token), a następnie kieruje do odpowiedniego budynku, piętra, korytarza itp.

Pracownik idzie więc z kartą (tokenem) do miejsca, gdzie został skierowany. Trafia na **portiera** (zasób). Portier może być elektroniczny (np. czytnik kart otwierający drzwi). Portier sprawdza token i wpuszcza pracownika do jego nowego miejsca pracy.

Praktyczny przykład:

1.



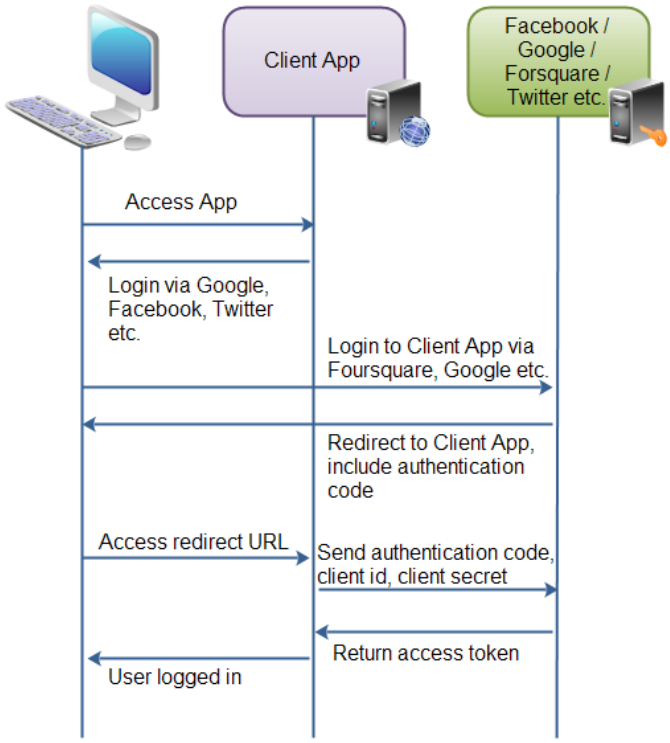
Użytkownik uruchamia grę internetową. Gra “prosi” użytkownika o zalogowanie poprzez Facebook. Użytkownik loguje się do Facebooka i jest odsyłany z powrotem do gry. Gra ma teraz dostęp do wybranych danych użytkownika (tych z Facebooka) i może wywoływać funkcje w Facebooku w imieniu użytkownika, np. uaktualnienia statusu.

2. Diagram poniżej pokazuje najczęstsze zastosowanie OAuth2 dla ww. przypadku.

a) użytkownik uruchamia webową **aplikację kliencką** (*client web application*). W tej apce jest przycisk np. "Login via Facebook" (lub poprzez Google lub Twitter).

b) Gdy użytkownik klika przycisk logowania jest przekierowywany do **aplikacji autentykującej** (np. do Facebooka). Użytkownik loguje się do aplikacji autentykującej i jest pytany czy chce dać dostęp webowej apce klienckiej do jego danych znajdujących się w aplikacji autentykującej (Facebooku). Użytkownik się zgadza.

c) Aplikacja autentykująca przekierowuje użytkownika do **adresu przekierowania** (*redirect URI*), który webowa apka kliencka dostarczyła aplikacji autentykującej.



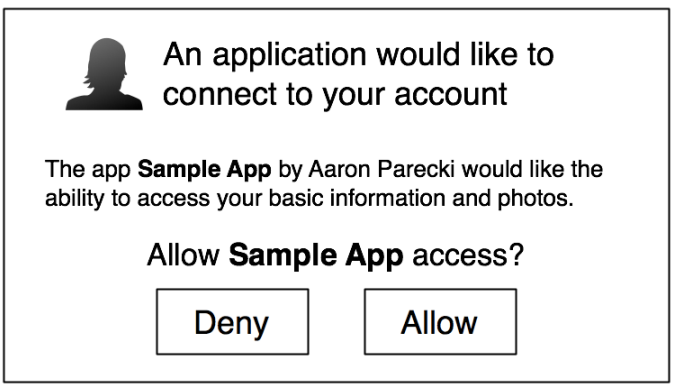
Dostarczanie adresu przekierowania (redirect URI) najczęściej jest realizowane poprzez rejestrację apki klienckiej w aplikacji autentykującej. W czasie procesu rejestracji właściciel apki klienckiej "rejestruje" (zapisuje) adres przekierowania (redirect URI). Również w czasie rejestracji aplikacja autentykująca (np. Facebook) dostarcza aplikacji klienckiej **ID klienta** (*client id*), które jest dostępne publicznie oraz **hasło klienta** (*client password* lub *client secret*).

Przykładowy link "ukryty" pod "Log in with Facebook" może wyglądać tak:

https://oauth2server.com/auth?response_type=code&
client_id=CLIENT_ID&redirect_uri=REDIRECT_URI&scope=photos

, gdzie oauth2server.com może być np. Facebookiem.

Po kliknięciu przycisku logowania możemy zobaczyć coś w stylu jak poniżej:



Po kliknięciu "Zezwól" użytkownik jest przekierowywany do swojej strony (dzięki *redirect URI*)

wraz z kodem autentykacji (authentication code), który jest dołączany do URI, np.:

https://oauth2client.com/cb?code=AUTH_CODE_HERE

d) Użytkownik w aplikacji klienckiej uzyskuje dostęp do strony określonej w redirect URI. Aplikacja kliencka wysyła do aplikacji autentykującej ID klienta (client id), hasło klienta (client password) i kod autoryzacji odebrany wcześciej:

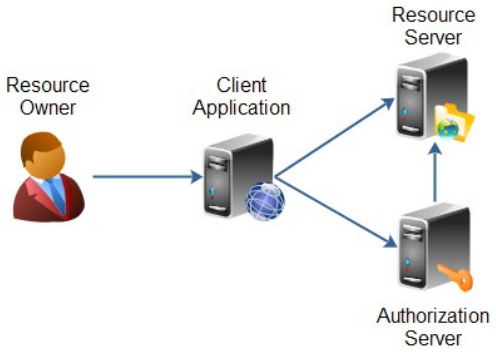
POST https://api.oauth2server.com/token grant_type=authorization_code& code=AUTH_CODE_HERE& redirect_uri=REDIRECT_URI& client_id=CLIENT_ID& client_secret=CLIENT_SECRET

Aplikacja autentykująca przesyła w drugą stronę token dostępu (access token), np.:

{ "access_token":"RsT5OjbzRn430zqMLgV3Ia" }

Po uzyskaniu tokenu dostępu przez aplikację kliencką, może być on użyty do uzyskania dostępu do zasobów zalogowanego użytkownika w Facebooku, Google, Twitterze.

Jeszcze raz o rolach w OAuth2:



Resource owner to osoba (najczęściej) lub aplikacja (Oauth 2.0 dopuszcza taką opcje), która posiada dane. Dane te moga zostać udostępnione. Np. użytkownik na FB lub Google może być *resource owner*'em. Zasoby, które posiada to po prostu jego dane.

Resource server to serwer, na którym znajdują się zasoby. Np. Facebook lub Google to *resource server'y* (lub je posiadają).

Client application to aplikacja oczekująca dostępu do zasobów zgromadzonych na *resource serverze*. Zasobów, które są własnością *resource owner*'a. Client application może być np. grą żądającą dostępu konta użytkownika na FB.

Authorization server to serwer autoryzujący do dostępu do zasobów, które należą do *resource owner*'a. *Authorization server* i *resource server* mogą być tym samym serwerem, ale nie muszą. Specyfikacja OAuth 2.0 nic nie mówi o komunkacji tych serwerów, gdy są rozdzielone. To wewnętrzna decyzja projektanta systemu.