Një Hyrje në OAuth 2

Prezantimi

OAuth 2 është një authorization framework që u mundëson aplikacioneve të marrin qasje të kufizuar në llogaritë e përdoruesve në një shërbim HTTP. Funksionon duke deleguar autentikimin e përdoruesit tek shërbimi që mban llogarinë e përdoruesit dhe autorizon aplikacionet e palëve të treta për të hyrë në llogarinë e përdoruesit. OAuth 2 siguron rrjedhat e autorizimit për aplikacionet në internet dhe desktop, dhe pajisjet mobile.

Ky udhëzues informativ është i orientuar drejt zhvilluesve të aplikacioneve dhe ofron një pasqyrë të OAuth 2.

Le të fillojmë me rolet OAuth!

Rolet e OAuth (OAuth Roles)

OAuth përcakton katër role:

- Zotërues Burimesh (Resource Owner)
- Klient (Client)
- Server Burimesh (Resource Server)
- Server Autorizimi (Authorization Server)

Do të detajojmë secilin rol në nënseksionet e mëposhtme.

Resource Owner: *User*

Zotëruesi i burimeve është *përdorues* i cili autorizon një aplikacion për të hyrë në llogarinë e tij. Qasja e aplikacionit në llogarinë e përdoruesit është e kufizuar në "fushëveprimin" (scope) e autorizimit të dhënë (p.sh. lexim ose shkrim).

Resource / Authorization Server: *API*

Serveri i burimeve mban llogaritë e përdoruesve të mbrojtur dhe serveri i autorizimit verifikon identitetin e *përdoruesit* dhe pastaj lëshon argumentet e qasjes (access tokens) për *aplikacionin*.

Nga një pikëpamje e zhvilluesit të aplikacionit, një shërbimi API përmbush rolin e serverit të burimeve dhe autorizimit. Do t'i referohemi të dy këtyre roleve të kombinuara, si roli i *Shërbimit* ose i *API*-t.

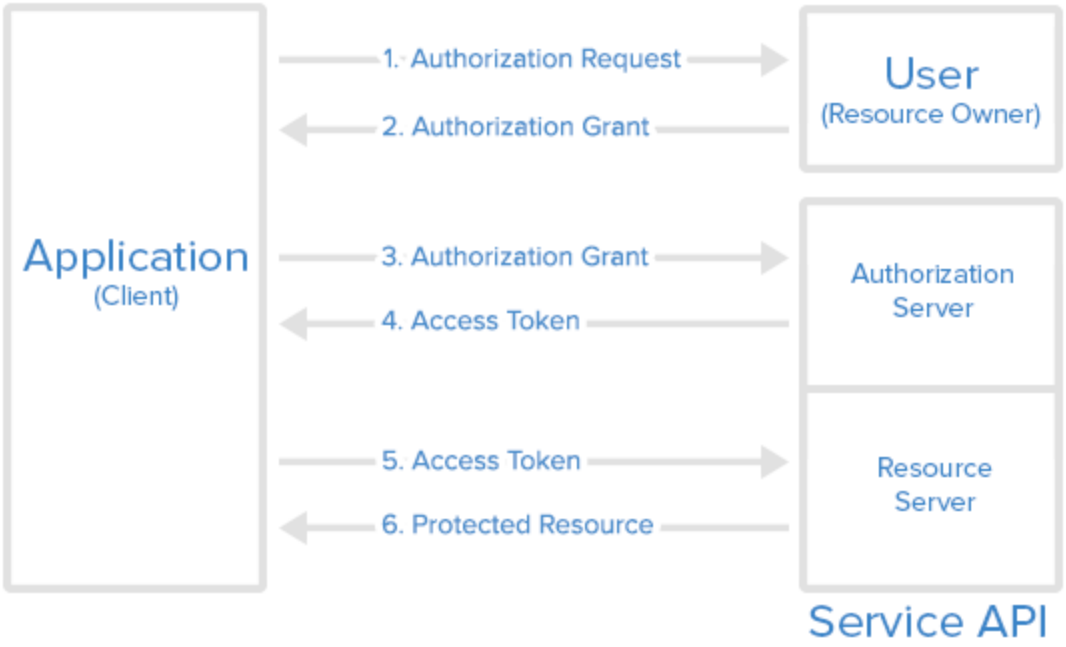
Client: *Application*

Klienti është *aplikacioni* që dëshiron të hyjë në llogarinë e *përdoruesit*. Para se të mund ta bëjë këtë, duhet të autorizohet nga përdoruesi dhe autorizimi duhet të vërtetohet nga API.

Rrjedha Abstrakte e Protokollit

Tani që ju keni një ide se cilat janë rolet OAuth, le të shohim në diagramën e më poshtme se si zakonisht ndërveprojnë ato me njëri-tjetrin:

Rrjedha Abstrakte e Protokollit



Këtu është një shpjegim më i hollësishëm i hapave në diagramë:

1. *Aplikacioni* kërkon autorizimin për të përdorur burimet e shërbimit nga *përdoruesi*
2. Nëse *përdoruesi* autorizon kërkesën, *aplikacioni* merr autorizim
3. *Aplikacioni* kërkon një access token nga *serveri i autorizimit* (API) duke paraqitur vërtetimin e identitetit të vet dhe dhënien e autorizimit.
4. Nëse identiteti i aplikacionit vërtetohet dhe e drejta e autorizimit është e vlefshme, *serveri i autorizimit* (API) dërgon një access token te aplikacioni. Autorizimi është kryer.
5. *Aplikacioni* kërkon burime nga *serveri i burimeve* (API) dhe paraqet access token-in për autentikim
6. Nëse access token-i është i vlefshëm, *serveri i burimeve* (API) i shërben burimet *aplikacionit*

Rrjedha aktuale e këtij procesi do të ndryshojë në varësi të llojit të dhënjes së autorizimit në përdorim, por kjo është ideja e përgjithshme. Do të shqyrtojmë lloje të ndryshme të dhënies së autorizimit në një seksion të mëvonshëm.

Regjistrimi i Aplikacionit

Para se të përdorni OAuth në aplikacionin tuaj, duhet të regjistroni aplikacionin tuaj te shërbimi. Kjo bëhet përmes një formulari regjistrimi në pjesën e "zhvilluesit" ose faqes së internetit të shërbimit, "API"-t , ku do të jepni informacionin e mëposhtëm (dhe ndoshta detajet rreth aplikacionit tuaj):

- Emri e Aplikacionit
- Website-n e Aplikacionit
- URI-ja përcjellëse ose URL e rikthimit

URI-ja përcjellëse (Redirect URI) është ajo ku shërbimi do të përcjellë përdoruesin pasi të autorizojë (ose të mohojë) aplikacionin tuaj, dhe për këtë është pjesa e aplikacionit tuaj që do të trajtojë authorization codes ose access tokens.

Client ID dhe Client Secret

Pasi aplikacioni juaj të regjistrohet, shërbimi do të lëshojë "kredencialet e klientit" në formën e një *identifikuesi të klientit* dhe një *vagu sekret karakteresh të klientit*. ID-ja e klientit është një varg i ekspozuar publikisht që përdoret nga API i shërbimit për të identifikuar aplikacionin dhe përdoret gjithashtu për të ndërtuar URL-të e autorizimit që u janë paraqitur përdoruesve. Vagu sekret i karaktereve të klientit përdoret për të vërtetuar identitetin e aplikacionit tek API i shërbimit kur aplikacioni kërkon të hyjë në një llogari të një përdoruesi dhe duhet të mbahen të fshehta ndërmjet aplikacionit dhe API-t.

Dhënja e Autorizimit

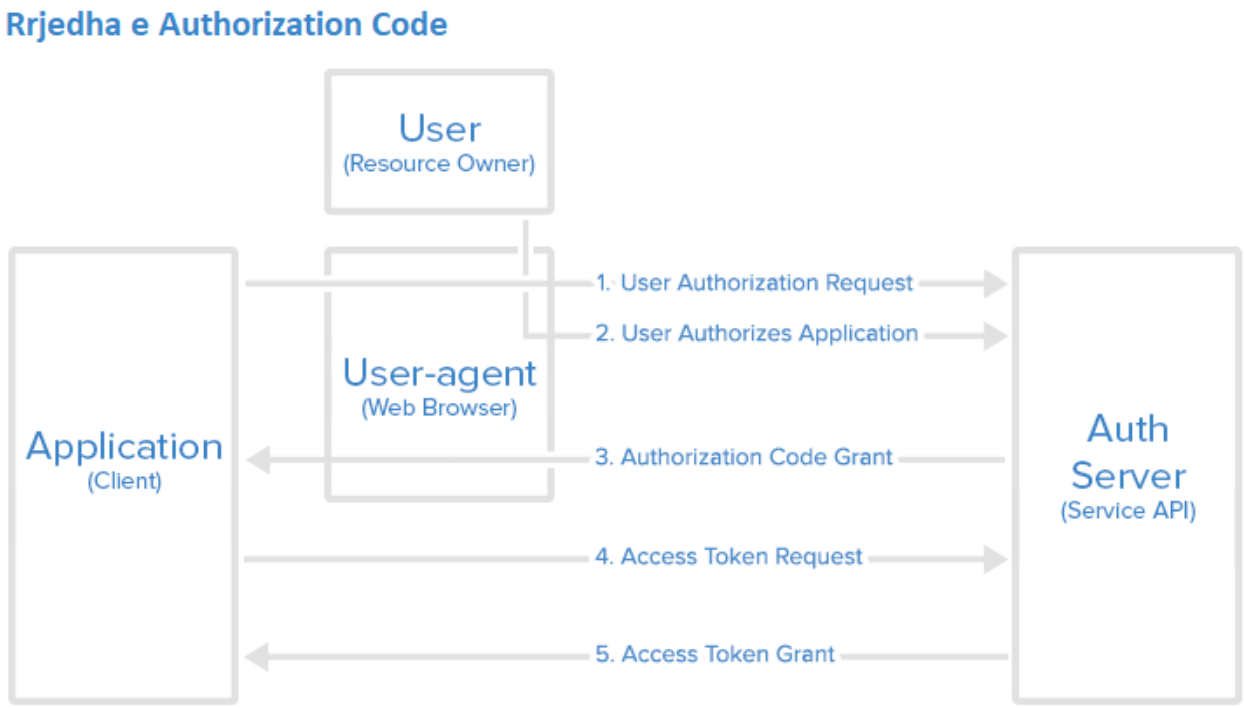
Në *rrjedhën abstrakte të protokollit* më sipër, katër hapat e parë mbulojnë marrjen e një të drejte për autorizimi dhe një access token-i. Lloji i dhënjes së autorizimit (authorization grant) varen nga metoda e përdorur nga aplikacioni për të kërkuar autorizim dhe llojet e mbështetura nga API. OAuth 2 përcakton katër lloje dhënje të autorizimit, secila prej tyre është e dobishme në raste të ndryshme:

- Authorization Code: përdoret me aplikacionet në anën e serverit
- Implicit: përdoret me aplikacionet mobile ose web(aplikacionet që përdoren në pajisjen e përdoruesit)
- Resource Owner Password Credentials: përdoret me aplikacionet e besueshme, të tilla si ato në pronësi të vetë shërbimit
- Client Credentials: përdoret me aplikacionet qe përdorin qasje në API

Tani ne do të përshkruajmë llojet e granteve në mënyrë më të hollësishme, në seksionet vijuese.

Grant Type: Authorization Code

Tipi authorization code grant është më i zakonshmi, sepse është i optimizuar për *aplikacionet në anën e serverit*, ku kodi burimor nuk është i ekspozuar publikisht dhe konfidencialiteti i kredincialit *Client Secret* mund të mbahet. Ky është një rrjedhë e bazuar në ridrejtim, që do të thotë se aplikacioni duhet të jetë i aftë të bashkëveprojë me agjentin e përdoruesit (*user-agent*:dmth. shfletuesin e përdoruesit) dhe marrjen e kodeve të autorizimit të API-t që janë dërguar përmes agjentit të përdoruesit.

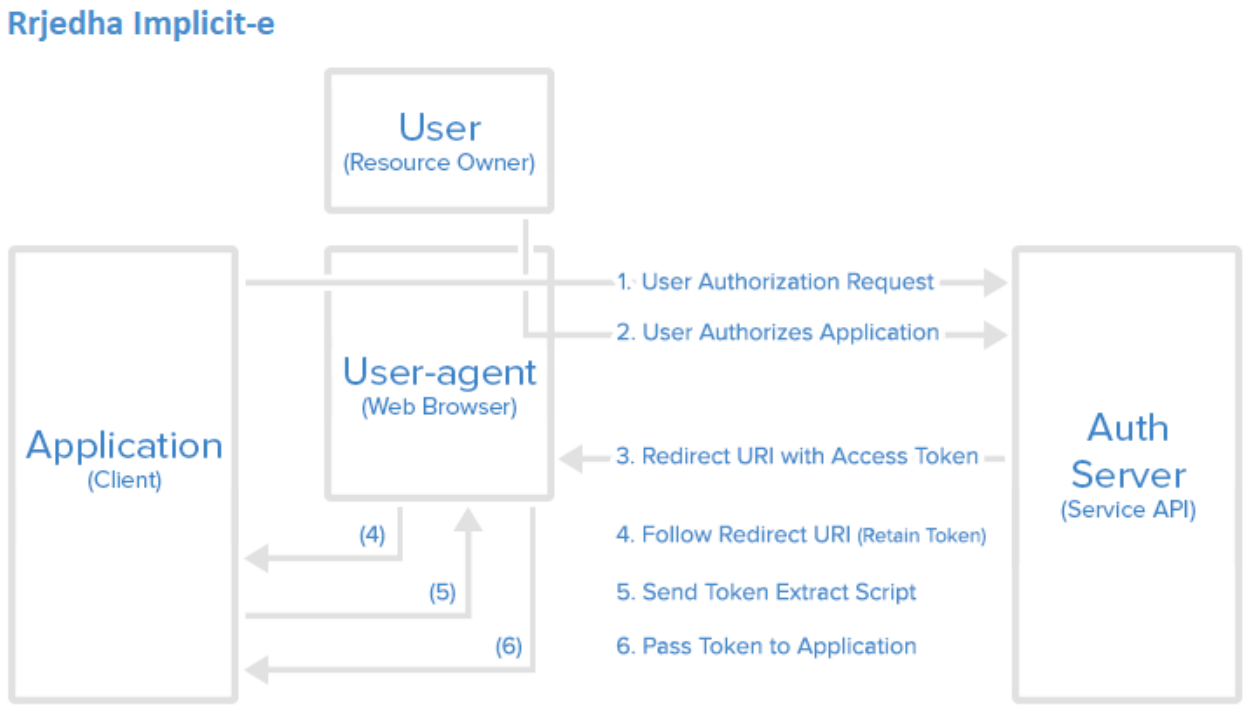


Grant Type: Implicit

Lloji implicit grant përdoret për aplikacionet celulare dhe web (dmth. aplikacionet që funksionojnë në një shfletues interneti), ku konfidencialiteti *client secret* nuk është i garantuar. Tipi implicit grant është gjithashtu një rrjedhë e bazuar në ridrejtim, por access token i jepet agjentit të përdoruesit (user-agent) për ta përcjellë te aplikacioni, kështu që mund të jetë i ekspozuar ndaj përdoruesit dhe aplikacioneve e tjera në pajisjen e përdoruesit. Gjithashtu, kjo rrjedhë nuk vërteton identitetin e aplikacionit dhe mbështetet në URI-në e ridrejtuar (që ishte regjistruar në shërbim) për t'i shërbyer këtij qëllimi.

Lloji implicit grant nuk suporton refresh tokens.

Rrjedha e implicit grant në thelb funksionon si në vijim: përdoruesit i kërkohet të autorizojë aplikacionin, atëherë serveri i autorizimit kalon token-in e qasjes përsëri te agjenti i përdoruesit, i cili e kalon në aplikacion.



Grant Type: Resource Owner Password Credentials

Me llojin resource owner password credentials grant, përdoruesi i jep kredencialet shërbimit (username and password) direkt në aplikacion, i cili përdor kredencialet për të marrë një access token nga shërbimi. Ky lloj granti duhet të aktivizohet vetëm në serverin e autorizimit nëse rrjedhat e tjera nuk janë të zbatueshme. Gjithashtu, duhet të përdoret vetëm nëse aplikacioni i besohet përdoruesit (p.sh. është në pronësi të shërbimit ose OS-së së përdoruesit).

Grant Type: Client Credentials

Lloji client credentials grant i siguron një aplikacioni një mënyrë për të hyrë në llogarinë e vet të shërbimit.

Rrjedha Client Credentials

Aplikacioni kërkon një access token duke dërguar kredencialet e tij, ID-në e klientit dhe vargun sekret të karaktereve të klientit në serverin e autorizimit. Një shembull kërkese POST mund të duket si në vijim:

```
https://oauth.example.com/token?grant_type=client_credentials&client_id=CLIENT_ID
&client_secret=CLIENT_SECRET
```

Nëse kredencialet e aplikacionit kontrollohen, serveri i autorizimit kthen një access token për aplikacionin. Tani aplikacioni është i autorizuar të përdorë llogarinë e vet!