Informaciona bezbednost

OAuth 2.0

dr Milan Stojkov

Katedra za informatiku

2022.

Fakultet tehničkih nauka
Univerzitet u Novom Sadu

OAuth 2.0

- RFC6249
- *The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf.*

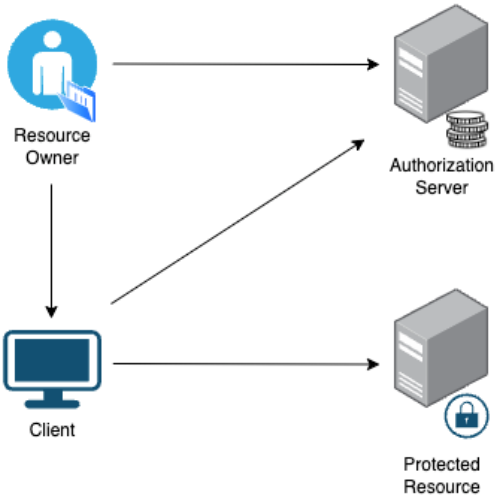
OAuth 2.0

- RFC6249
- *The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf.*

U prevodu

- OAuth 2.0 je delegacioni protokol koji dopušta ljudima da dozvole aplikacijama da pristupe resursima u njihovo ime

Ko je uključen u komunikaciju?



Vlasnik resursa/resource owner

- Ima pristup nekom resursu ili API-u
- Može da delegira pristup tom resursu ili API-u
- Obično ima pristup web pretraživaču
- Obično je osoba

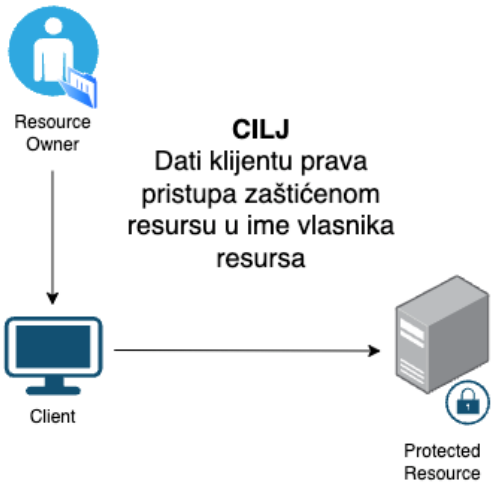
Zaštićeni resurs/protected resource

- Web servis (API) sa implementiranim bezbednosnim kontrolama
- Štiti resurse za vlasnika
- Deli pristup resursu na zahtev vlasnika

Klijentska aplikacija

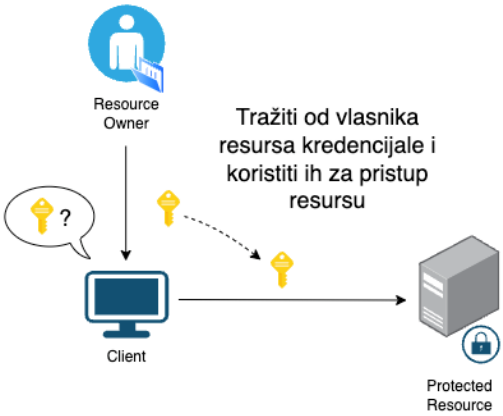
- Želi da pristupi zaštićenom resursu
- Želi da uradi nešto sa tim resursima u ime vlasnika
- Može da bude web server
  - Ali je i dalje klijent u OAuth terminologiji
  - Može da bude i nativna aplikacija

Šta OAuth 2.0 treba da reši?

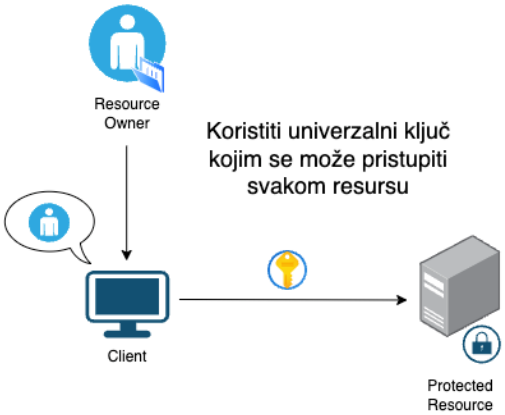


Opcija 1: Ukrasti ključeve

Opcija 2: Tražiti kredencijale



Opcija 3: Koristiti uniuverzalni ključ

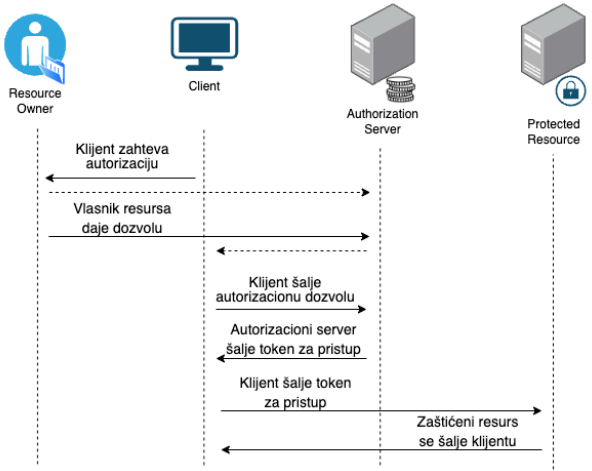


Opcija 4: Koristiti token specifičan za servis kojem se pristupa

Autorizacioni server (AS)

- Generiše tokene za klijenta
- Autentifikuje vlasnike resursa (korisnike)
- Autentifikuje klijente
- Upravlja autorizacijom

Pojednostavljeni tok komunikacije



OAuth 2.0 sekvenca

1. Vlasnik resursa ukazuje klijentskoj aplikaciji da želi da aplikacija uradi operaciju za njega
   - Npr. da učita slike sa servisa kako bi se moglo izvršiti štampanje
2. Klijentska aplikacija zahteva autorizaciju od vlasnika resursa na autorizacionom serveru
3. Vlasnik resursa daje prava klijentskoj aplikaciji
4. Klijentska aplikacija dobija token od autorizacionog servera
5. Klijentska aplikacija predstavlja token zaštićenom resursu

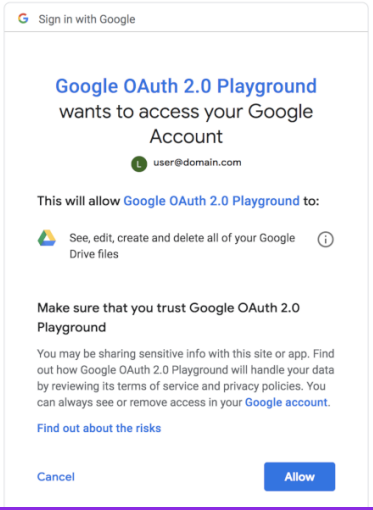
OAuth tokeni

- Predstavljaju dozvoljena delegirana prava
  - Data od strane vlasnika resursa za klijenta za zaštićeni resurs
- Izdaje ih autorizacioni server
- Koristi ih klijent
  - Format treba da je neprepoznatljiv klijentu
- Koriste ih zaštićeni resursi

OAuth tokeni - primeri

- `92d42038006dba95d0c501951ac5b5eb`
- `2df029c6-b38d-4083-b8d9-db67c774d13f`
- `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.`
  `eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IlBlcmEgUG`
  `VyaWMiLCJhZG1pbiI6dHJ1ZX0.`
  `B1nQ1ly6X8RQ28SnJVSSNxOHNhln901wcYFz75iXN2c`

Primer poznate situacije



Šta je zapravo OAuth 2.0?

- Umesto jednog protokola definiše zajedničke koncepte i komponente i načine na koji oni međusobno treba da interaguju
- Nije jedan standard već skup standarda za različite slučajeve korišćenja
  - Npr. način na koji se dobijaju tokeni i kako se koriste tokeni

Šta OAuth 2.0 nije?

- OAuth 2.0 nije definisan van HTTP protokola
- Oslanja se na TLS za obezbeđivanje poruka i tokena
- Postoje pokušaji da se OAuth koristi i preko drugih protokola (Constrained Application Protocol – CoAP, RFC 7252)

Šta OAuth 2.0 nije?

- OAuth 2.0 nije autentifikacioni protokol
- Oslanja se na autentifikaciju
  - Npr. vlasnik resursa se autentifikuje autorizacionom serveru
- Ne komunicira ništa o korisniku
- Autentifikacioni protokoli se mogu napraviti pomoću OAuth 2.0
  - Npr. OpenID Connect

Šta OAuth 2.0 nije?

- OAuth 2.0 nije namenjen za delegiranje prava drugim korisnicima
- Delegira prava softveru a ne drugom korisniku
- Može se koristiti da se kreira sistem sposoban za delegiranje prava drugim korisnicima
  - Npr. User-Managed Access (UMA)

Šta OAuth 2.0 nije?

- OAuth 2.0 ne definiše mehanizme za procesiranje informacija o autorizaciji
- Definiše da to treba da se desi ali konkretna implementacija se ostavlja sistemu
  - Npr. treba se koristiti token ali šta sve sa njim može da se uradi je na konkretnoj implementaciji

Šta OAuth 2.0 nije?

- OAuth 2.0 ne definiše format tokena
- Tokeni treba da budu neprepoznatljivi klijentu
- Tokeni treba da budu izdati od strane autorizacionog servera i da ih koristi server resursa ali mogu biti kreirani u bilo kom formatu
  - Npr. JSON Web Token (JWT) je jedna standardizovana reprezentacija
  - RFC 7519

Šta OAuth 2.0 nije?

- OAuth 2.0 ne definiše kriptografske metode
- Oslanja se na TLS za zaštitu informacija u tranzitu
- Ostavljen je prostor da se razviju specifikacije koje će se baviti tim pitanjima
  - Npr. JSON Object Signing and Encryption (JOSE)

OAuth 2.0 sekvenca - Korak 1

- Vlasnik resursa ukazuje klijentskoj aplikaciji da želi da aplikacija uradi operaciju za njega
  - Npr. da učita slike sa servisa kako bi se moglo izvršiti štampanje
- Klijentska aplikacija shvata da mora da dobije OAuth token i šalje vlasnika resursa na autorizacioni server sa zahtevom koji ukazuje na to da treba da se desi delegacija prava sa vlasnika resursa na klijentsku aplikaciju

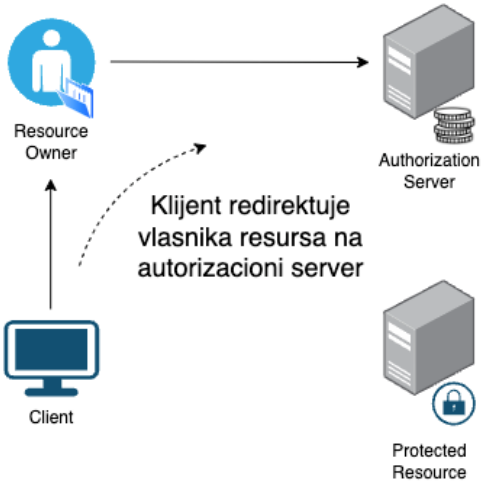
```
HTTP/1.1 302 Moved Temporarily
x-powered-by: Express
Location: http://myauthserver.com/authorize?response_type=code&scope=foo&client
_id=oauth-client-1&redirect_uri=http%3A%2F%2Fmyapp.com%2Fcallback&
state=Lwt50DDQKUB8U7jtfLQCVGDL9cnmwHH1
Vary: Accept
Content-Type: text/html; charset=utf-8
Content-Length: 456
Date: Tue, 20 Dec 2022 21:55:14 GMT
Connection: keep-alive
```

OAuth 2.0 sekvenca - Korak 1

- Web pretraživač šalje HTTP GET zahtev autorizacionom serveru

```
GET /authorize?response_type=code&scope=foo&client_id=oauth-client
-1&redirect_uri=http%3A%2F%2Flmyapp.com%
2Fcallback&state=Lwt50DDQKUB8U7jtfLQCVGDL9cnmwHH1 HTTP/1.1
Host: myauthserver.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:39.0)
Gecko/20100101 Firefox/39.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: http://myapp.com/
Connection: keep-alive
```

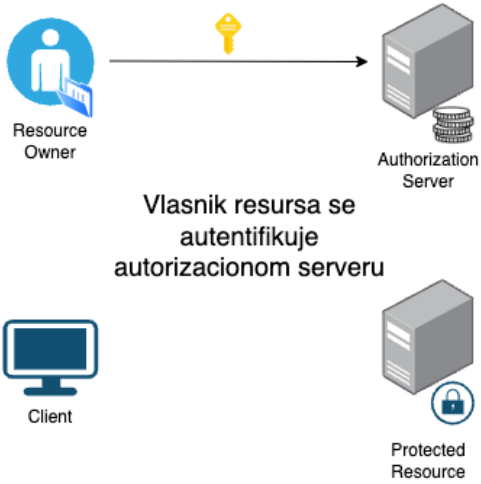
Korak 1


Resource Owner
Authorization Server
Klijent redirektuje vlasnika resursa na autorizacioni server
Client
Protected Resource

OAuth 2.0 sekvenca - Korak 2

- Vlasnik resursa se autentifikuje
- Autentifikacija se obavlja direktno sa autorizacionim serverom u web pretraživaču (nikad kroz klijentsku aplikaciju)

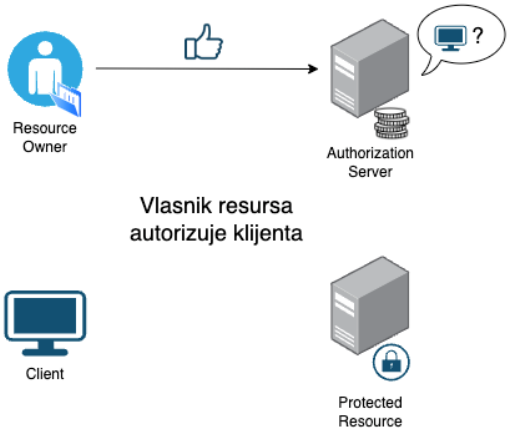
Korak 2


Resource Owner
Authorization Server
Vlasnik resursa se autentifikuje autorizacionom serveru
Client
Protected Resource

OAuth 2.0 sekvenca - Korak 3

- Vlasnik resursa autorizuje klijentsku aplikaciju
- Vlasnik resursa bira koji deo odgovornosti će da da klijentskoj aplikaciji

Korak 3


Resource Owner
Authorization Server
?
Vlasnik resursa autorizuje klijenta
Client
Protected Resource

OAuth 2.0 sekvenca - Korak 4

- Autorizacioni server redirektuje korisnika nazad na klijentsku aplikaciju
- `code` parametar sadrži vrednost koja je jednokratna a koja predstavlja rezultat autorizacije korisnika
- Klijentska aplikacija proverava da li se `state` parametar podudara sa parametrom poslatim u prethodnom koraku

```
HTTP 302 Found
Location:
http://myapp.com/oauth_callback?code=8V1pr0rJ&state=Lwt50DDQKUB8U7jtfLQCVGDL9cnmwHH1
```

- Web pretraživač pravi GET zahtev nazad ka klijentskoj aplikaciji

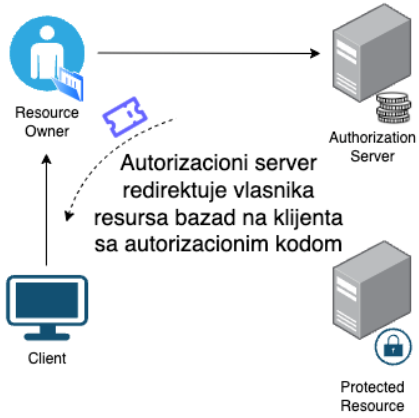
```
GET /callback?code=8V1pr0rJ&state=Lwt50DDQKUB8U7jtfLQCVGDL9cnmwHH1 HTTP/1.1
Host: myapp.com
```

- HTTP zahtev je za klijentsku aplikaciju a ne na autorizacionom serveru

```
User-Agent: Mozilla/5.0 ...
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: http://myauthserver.com/authorize?response_type=code&scope=foo&
client_id=oauth-client-1&redirect_uri=http%3A%2F%2Fmyapp.com%2Fcallback&
state=Lwt50DDQKUB8U7jtfLQCVGDL9cnmwHH1

Connection: keep-alive
```

Korak 4

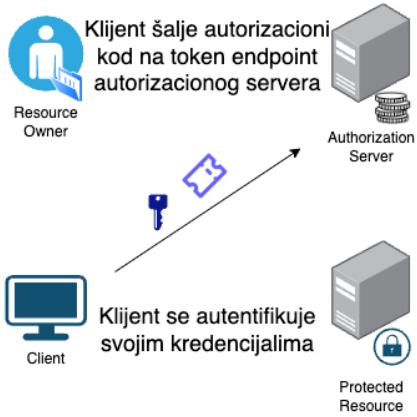

Resource Owner
Authorization Server
Autorizacioni server redirektuje vlasnika resursa bazad na klijenta sa autorizacionim kodom
Client
Protected Resource

OAuth 2.0 sekvenca - Korak 5

- Klijentska aplikacija pravi HTTP POST zahtev na autorizacioni server
- Šalje `client_id` i `client_secret` kao HTTP Basic authorization zaglavlje
- Šalje `code`
- Ovaj zahtev se obavlja direktno između klijentske aplikacije i autorizacionog servera bez mešanja web pretraživača i vlasnika resursa

```
POST /token
Host: myauthserver.com
Accept: application/json
Content-type: application/x-www-form-encoded
Authorization: Basic b2F1dGgtY2xpZW50LTE6b2F1dGgtY2xpZW50LXNlY3JldC0x
grant_type=authorization_code&
redirect_uri=http%3A%2F%2Fmyapp.com%2Fcallback&code=8V1pr0rJ
```

Korak 5

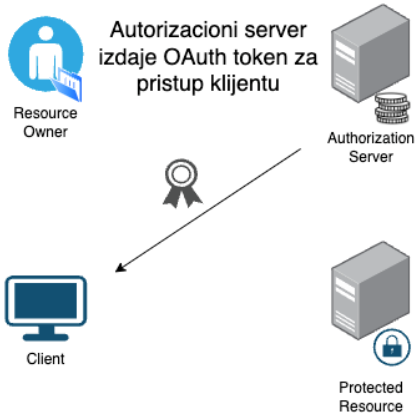

Klijent šalje autorizacioni kod na token endpoint autorizacionog servera
Resource Owner
Authorization Server
Client
Klijent se autentifikuje svojim kredencijalima
Protected Resource

OAuth 2.0 sekvenca - Korak 6

- Autorizacioni server proverava klijentske kredencijale iz `Authorization` zaglavlja
- Proverava vrednost `code` parametra
- Ako je zahtev validan, autorizacioni server izdaje token

```
HTTP 200 OK
Date: Tue, 20 Dec 2022 21:56:01 GMT
Content-type: application/json
{
"access_token": "987tghjkiu6trfghjuytrghj",
"token_type": "Bearer"
}
```

Korak 6

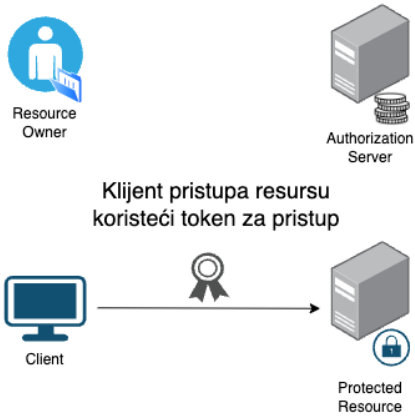

Autorizacioni server izdaje OAuth token za pristup klijentu
Resource Owner
Authorization Server
Client
Protected Resource

OAuth 2.0 sekvenca - Korak 7

- Klijentska aplikacija može dalje da koristi token za pristup zaštićenom resursu

```
GET /resource HTTP/1.1
Host: mycoolresourceserver.com
Accept: application/json
Connection: keep-alive
Authorization: Bearer 987tghjkiu6trfghjuytrghj
```

Korak 7


Resource Owner
Authorization Server
Klijent pristupa resursu koristeći token za pristup
Client
Protected Resource

OAuth 2.0 scopes

- Stringovi koji reprezentuju šta token može da uradi
- Klijentska aplikacija može da traži *scope*
- Vlasnik resursa odobrava *scope*
- Token za pristup je vezan za *scope*

OAuth 2.0 refresh token

- Takođe ga izdaje autorizacioni server
- Refresh token se ne šalje zaštićenom resursu
- Klijentska aplikacija koristi refresh token da zahteva novi token za pristup bez potrebe da uključuje vlasnika resursa u proces