OAuth 2.0

Grundlagen des Software Engineering

Julian Stadler

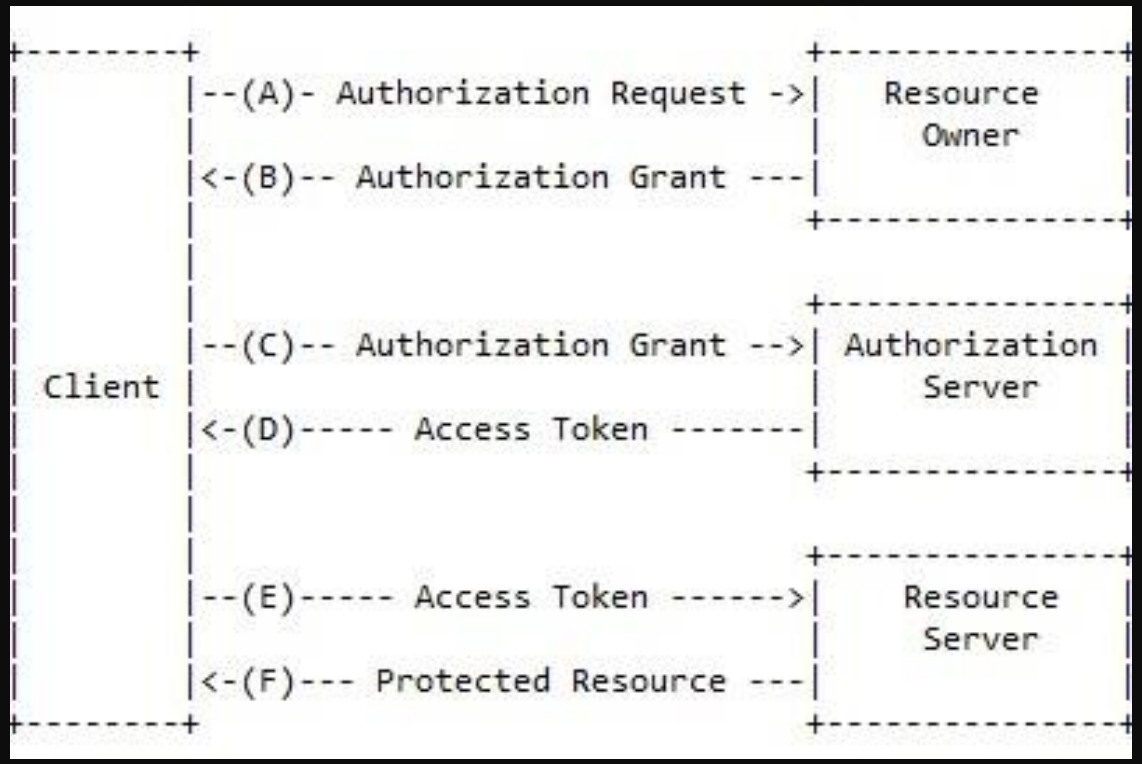
Gliederung

- Das OAuth 2.0 Framework
- Protocol Flow
- Nutzerrollen innerhalb des Frameworks
- Der Autorisierungsprozess
  - Authorization Grant
  - Client Types
  - Client registration
- Praxisbeispiel

OAuth 2.0 Framework

- Autorisierungsstandart für API Endpunkte
- Bietet Autorisierung für Desktop-, Web- und Mobileanwendungen
- Fokus auf einfache Kliententwicklung
  - Jedoch auch spezifizierte Autorisierungsflows für spezielle Anwendungen

Protocol Flow



Nutzerrollen

Resource Owner

- Hauptsächlich Eigentümer
- Vergibt Berechtigungen

Resource Server

- Server, auf welchem die geschützten Daten liegen
- Kann nur durch ein access token benutzt werden

Client

- Anwendung, welche auf die geschützten Daten zugreifen möchte

Authorization Server

- Erteilt nach erfolgreicher Autorisierung das Access Token

Authorization Grant

= Eine Referenz, welche die Autorisierung des Besitzers wiederspielgelt, um auf seine Ressourcen zuzugreifen.

- Meist benutzten Arten sind – Authorization Code, Client Credentials, und Refresh Token

Authorization Grant – Authorization Code

- Autorisierungsserver als Vermittler zwischen Client und Resource Owner
  - Client verbindet den Resource Owner mit dem Autorisierungsserver, welcher wiederum den Resource Owner mit dem Autorisierungscode zurück an den Client leitet

-> dadurch werden die Autorisierungsdaten des Resource Owners niemals direkt mit dem Client geteilt

Authorization Grant – Client Credentials

```
POST /token HTTP/1.1
Host: authorization-server.com

grant_type=client_credentials
&client_id=xxxxxxxxxx
&client_secret=xxxxxxxxxx
```

- Wird genutzt, wenn auf eigene Ressourcen zugegriffen werden will und nicht im Namen eines Benutzers
- Normalerweise werden zusätzliche Parameter wie client_id und client_secret mitgegeben
  - Werden dann zur Autorisierung genutzt

Authorization Grant – Refresh Token

- Wird genutzt, um nach Ablauf des eigentlichen Access Tokens ein Neues anzufragen
  - Normalerweise nur bei vertraulichen Clients genutzt
- Dadurch können weitere Zugangstoken zur Verfügung gestellt werden, ohne eine erneute Interaktion mit dem Nutzer

Client Types

Confidential clients

- In der Lage, sich sicher am Autorisierungsserver zu autorisieren
- Bewahren das Token sicher vor Dritten auf

Public clients

- Können Anmeldedaten nicht sicher halten
- Zum Beispiel Webanwendungen oder Mobileanwendungen

Client Registration

- Confidential clients autorisieren sich am Autorisierungsserver bei der Datenanfrage
- Normalerweise autorisiert sich der Client über das client_secret
- Andere Autorisierungsmöglichkeiten sind aber über Extensions verfügbar
  - Mutual TLS: Zugrifftoken wird an ein Zertifikat des Clients gebunden
  - Private Key JWT: Der Client erstellt und signiert einen JWT (JSONWeb Token) mit seinem eigenen privaten Schlüssel

Praxisbeispiel