Contents

1 Motivation

2 O-Auth 2.0
   • Motivation
   • Abstract Protocol Flow
   • Actors
   • Client Registration
   • Authorization Grants
     • Authorization Code
     • Implicit Grant
     • Resource Owner Password Credentials
     • Client Credentials
   • Authorization Server Response
   • Accessing Protected Resources

3 OpenID Connect 1.0

4 Single Sign On
Motivation
So you thought username-password based authentication was enough?

Which one do you think is better?

Figure: Twit Resource Access

Since weird guy has valid Twitter credentials is Twitter allowed to give away MY photos?
If yes is Twitter allowed decide which ones to give away?
Does having only valid username-password solve these problems?

Figure: Twit Resource Access 2

Weird guy has valid username-password.
Twitter does not own MY resources.
Twitter must be certain that I allowed him to get whatever he requires.
I am responsible for deciding which pictures he can read and which he cant.
Moral of the Story

We need much more than simple user-name password based authentication.

O-Auth 2.0, OpenId Connect 1.0 and Single Sign On are some web-based authentication and authorization protocols which can be issues mentioned in the previous slide.
Outline

1 Motivation

2 O-Auth 2.0
   - Motivation
   - Abstract Protocol Flow
   - Actors
   - Client Registration
   - Authorization Grants
     - Authorization Code
     - Implicit Grant
     - Resource Owner Password Credentials
     - Client Credentials
   - Authorization Server Response
   - Accessing Protected Resources

3 OpenID Connect 1.0

4 Single Sign On
Outline

1 Motivation

2 O-Auth 2.0
   - Motivation
   - Abstract Protocol Flow
   - Actors
   - Client Registration
   - Authorization Grants
     - Authorization Code
     - Implicit Grant
     - Resource Owner Password Credentials
     - Client Credentials
   - Authorization Server Response
   - Accessing Protected Resources

3 OpenID Connect 1.0

4 Single Sign On
O-Auth 2.0
Why use it?

What happens in an O-Auth 2.0?
- Random guy wants your photos on Twitter.
- He wants your permission to access 5 different photos.
- You decide to give access to read 3 photos.

What advantages does O-Auth 2.0 give?
- Users can control if third-party is allowed to access their resources.
- Users can decide which resources should be shared.
- Granted permissions can be revoked at any time.
- Users do not need to share your credentials with an untrusted third party application to give them access to their resources.
Outline

1 Motivation

2 O-Auth 2.0
   - Motivation
   - Abstract Protocol Flow
   - Actors
   - Client Registration
   - Authorization Grants
     - Authorization Code
     - Implicit Grant
     - Resource Owner Password Credentials
     - Client Credentials
   - Authorization Server Response
   - Accessing Protected Resources

3 OpenID Connect 1.0

4 Single Sign On
Figure: O-Auth 2.0 Abstract Protocol Flow
Outline

1 Motivation

2 O-Auth 2.0
   - Motivation
   - Abstract Protocol Flow
   - Actors
      - Client Registration
      - Authorization Grants
         - Authorization Code
         - Implicit Grant
         - Resource Owner Password Credentials
         - Client Credentials
      - Authorization Server Response
      - Accessing Protected Resources

3 OpenID Connect 1.0

4 Single Sign On
Typically four actors are involved in an O-Auth protocol flow.

**Resource Owner**  Owner of the resource. Entity capable of granting access to resources owned by them.

- **Example**
  - ME as a Twitter user.

**Resource Server**  Server where the resources are stored. Allows access to protected resources depending on access scope.

- **Example**
  - Twitter server at San Francisco.

**Client**  Any third party who wants to access protected resources.

- **Example**
  - Data analyst, mobile/web application etc.

**Authorization Server**  The server responsible for delegating access privileges to the “Client” based on the “Resource Owner”’s decisions.

- **Example**
  - Apache Oltu.
Outline

1 Motivation

2 O-Auth 2.0
   - Motivation
   - Abstract Protocol Flow
   - Actors
   - **Client Registration**
     - Authorization Grants
       - Authorization Code
       - Implicit Grant
       - Resource Owner Password Credentials
       - Client Credentials
     - Authorization Server Response
     - Accessing Protected Resources

3 OpenID Connect 1.0

4 Single Sign On
O-Auth 2.0
Client Registration

Figure: Client Registration Process

- **Step in protocol:** Before protocol is initiated. Typically a one-time task.
- **Medium:** Web registration form, third party validation etc.
- **Requirements:**
  1. **Client type:**
     - **Confidential** Ability to keep credentials secret.
     - **Public** Inability to keep credentials secret.
  2. **Redirection URI:** Client end-point where resource-owner should be redirected during authorization grant.
  3. **Additional Information:** Application name, website, logo, description etc.

- **Response:**
  1. **Client Identifier:** Unique string representing the registration information provided by the client. Is not a secret.
  2. **Client Secret:** A secret shared between client and authorization server.
Outline

1 Motivation

2 O-Auth 2.0
   - Motivation
   - Abstract Protocol Flow
   - Actors
   - Client Registration
   - **Authorization Grants**
     - Authorization Code
     - Implicit Grant
     - Resource Owner Password Credentials
     - Client Credentials
   - Authorization Server Response
   - Accessing Protected Resources

3 OpenID Connect 1.0

4 Single Sign On
O-Auth 2.0
Abstract Protocol Flow

Client

(A) Authorization Request
(B) Authorization Grant
(C) Authorization Grant
(D) Access Token
(E) Access Token
(F) Protected Resources

Resource Owner
Authorization Server
Resource Server

Figure: O-Auth 2.0 Abstract Protocol Flow
Covers: (A-B-C-D) in Abstract Protocol Flow.

Authorization Grant types:

1. Authorization Code
2. Implicit Grants
3. Resource Owner Password Credentials
4. Client Credentials

Abstract Flow

A:

B:
Client <-------- [Authorization Grant] <-------- Resource Owner

C:
Client ------- [Authorization Grant] ----------> Authorization Server

D:
Client <-------- [Access token] ---------- Authorization Server

Protocol Flow

Figure: O-Auth 2.0 Authorization Code Protocol Flow
(A) **Client** redirects users user-agent (web-browser) to authorization server.
   - The redirect contains the client id and redirection URI obtained during *registration.*

**Example**


(B) **Resource owner** authenticates to authorization server and grants required access.
(C) **Authorization server** redirects resource owner user-agent to client redirection URI with authorization code.

**Example**

https://dropletbook.com/callback?code=AUTHORIZATION_CODE

(D) **Client** forwards authentication code to authorization server.
   - At this point client authenticates with the authorization server.

**Example**

(E) Authorization server performs required client validation and returns access token.

Example

```
{"access_token": "ACCESS_TOKEN", "token_type": "bearer", "expires_in": 2592000, "refresh_token": "REFRESH_TOKEN", "scope": "read", "uid": 100101, ...
```
A:
Client ------- [Authorization Grant Type: Implicit Grant] -------> Resource Owner

D:
Client <-------- [Access token] ------- Resource Owner
O-Auth 2.0: Authorization Grants: Implicit Grant I

Protocol Flow

---

Figure: O-Auth 2.0 Implicit Grant Protocol Flow
O-Auth 2.0: Authorization Grants: Implicit Grant II

Protocol Flow

(A) **Client** redirects users user-agent (web-browser) to authorization server.
   - The redirect contains the client id and redirection URI obtained during *registration*.

Example


(B) **Resource owner** authenticates to authorization server and grants required access.

(C) **Authorization server** redirects resource owner user-agent to client redirection URI with access token in the URI.
   - Access token is visible to the resource owner.

Example

https://dropletbook.com/callback#token=ACCESS_TOKEN

(D) **Resource owner’s user agent** redirects to client supplied redirection URI.
   - This redirect does not include the access token.

(E) **Web Hosted Client Resource** provides a script that could extract the provided access token.

(F) **Resource owner’s user agent** runs the script.

(G) **Resource owner’s user agent** thus passes the access token to the client.
O-Auth 2.0: Authorization Grants: Resource Owner Password Credentials

Abstract Flow

B:
Client <------- [Resource Owner Password Credentials] ------ Resource Owner

C:
Client ------- [Resource Owner Password Credentials] --------> Authorization Server

D:
Client <-------- [Access token] ------ Authorization Server
O-Auth 2.0: Authorization Grants: Resource Owner Password Credentials I

Protocol Flow

Figure: O-Auth 2.0 Resource Owner Password Credentials Protocol Flow
(A) **Resource owner** initiates the flow by providing highly trusted client with his username and password.

(B) **Client** forwards credentials to authorization server.

(C) **Authorization Server** provides access token.
O-Auth 2.0: Authorization Grants: Client Credentials

Abstract Flow

C:
Client -------- [Resource Owner Credentials] --------> Authorization Server

D:
Client <-------- [Access token] ------- Authorization Server
Figure: O-Auth 2.0 Client Credentials Protocol Flow
(A) **Resource owner/Client** by authorizing to the authorization server.

**Example**

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

(B) **Authorization Server** provides access token.
Outline

1 Motivation

2 O-Auth 2.0
   - Motivation
   - Abstract Protocol Flow
   - Actors
   - Client Registration
   - Authorization Grants
     - Authorization Code
     - Implicit Grant
     - Resource Owner Password Credentials
     - Client Credentials
   - Authorization Server Response
   - Accessing Protected Resources

3 OpenID Connect 1.0

4 Single Sign On
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
{
  "access_token":"2YotnFZFEjr1zCsicMWpAA",
  "token_type":"example",
  "expires_in":3600,
  "refresh_token":"tGzv3J0kF0XG5Qx2T1KWIA",
  "example_parameter":"example_value"
}

Contains the following information:

- **access_token**: As provided by authorization server.
- **token_type**: “Bearer”, “Mac” etc.
- **expires_in** Lifetime in seconds of the access token
- **refresh_token** A refresh token which can be used to obtain a new access token.
- **scope** Space delimited string used to express access control specifications.
  - Requested initially by the client.
O-AUTH 2.0: AUTHORIZATION SERVER RESPONSE
Unsuccessful Validation

HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
{
  "error": "invalid_request"
}
Outline

1 Motivation

2 O-Auth 2.0
   - Motivation
   - Abstract Protocol Flow
   - Actors
   - Client Registration
   - Authorization Grants
     - Authorization Code
     - Implicit Grant
     - Resource Owner Password Credentials
     - Client Credentials
   - Authorization Server Response
   - Accessing Protected Resources

3 OpenID Connect 1.0

4 Single Sign On
O-Auth 2.0
Accessing Protected Resources

- **Client** sends query to resource server with access token.

**Example**
```
```

- **Resource server** interprets the authorization server’s response (access token).
- **Resource server** validates the token.
Materials referenced from:

1 Motivation

2 O-Auth 2.0
   - Motivation
   - Abstract Protocol Flow
   - Actors
   - Client Registration
   - Authorization Grants
     - Authorization Code
     - Implicit Grant
     - Resource Owner Password Credentials
     - Client Credentials
   - Authorization Server Response
   - Accessing Protected Resources

3 OpenID Connect 1.0

4 Single Sign On
0-Auth provides a mechanism of authorization.

What is the guarantee that the Client you are talking to is an authentic one?

OpenId provides that guarantee by authenticating the user.
Figure: OpenId Connect Example
Materials referenced from:

2. http://openid.net/connect/
Outline

1 Motivation

2 O-Auth 2.0
   - Motivation
   - Abstract Protocol Flow
   - Actors
   - Client Registration
   - Authorization Grants
     - Authorization Code
     - Implicit Grant
     - Resource Owner Password Credentials
     - Client Credentials
   - Authorization Server Response
   - Accessing Protected Resources

3 OpenID Connect 1.0

4 Single Sign On
Use one username password for access to multiple applications.

Each application vendor must trust primary login domain.

Eg. You login to gmail. You can access Youtube, Hangout and other Google apps using the same login session.
Figure: Single Sign On