OAuth Concepts

What is OAuth for?

An **authorization** protocol to allow an application to access a user's resources on a different system.

Analogy:

- *ATS authorization at bank to allow AIS or True to debit your mobile phone bill.*
- *หนังสือมอบอำนาจ (power of attorney) to authorize someone else to view your tax records.*

Counter-example

Email client apps require you to input your username & password for the mail server.

- The client app can do <u>anything</u> you can do.
- Your password is stored in the app.
- Difficult to revoke.

OAuth Roles (the Players)

Resource Owner	User who owns the resource
Client	Application that wants to access the resource
Authorization Server	Server that authenticates Resource Owner and grants access to Client. Does this using an Authorization Code and Access Token.
Resource Server	Hosts the resource(s) and manages access to it.
User Agent	software used by the user to interact with client, such as a web browser.

Example

Resource Owner	
Client	
Authorization Server	
Resource Server	

The Steps in Using OAuth

Prerequisite: Client **registers** the application on the Authorization Server.

Example: https://github.com/settings/applications/new

Client provides: *KU Polls Example*

App Name	`KU Polls`
Homepage URL	`http://localhost:8000/polls`
App Description	`Polls for KU`
Scopes	`what you want to access`

Callback URL

`http://127.0.0.1:8000/accounts/github/login/callback/`

Client Registration

Client application receives *KU Polls Example*

client_id `Ov23liFqvTsWsOScBptr`

client_secret

`bab2ca077c6daa534bc437b66b52767e7eaea951`

key `(not used on Github)`

authorization Url (where to send User Agent)

`https://github.com/login/oauth/authorize`

Using OAuth

When user of Client wants to authenticate or access his resources...

1. Client requests authorization by redirecting the User Agent (browser) to Auth Server with an "Authorization Request" for resource it wants to access.

2. User (Resource Owner) authenticates himself and agrees to grant access to Client app

3. Auth Server gives User Agent a temporary authorization code & redirects him back to Client.

Using OAuth (cont.)

4. User Agent gives Auth Server the authorization code along with Client's credentials

5. Auth Server gives Client an Access Token. This grants access to specified resources.

6. Client includes the Access Token in each request it sends to Resource Server.
The Resource Server checks validity before granting access (in case the token has been revoked or Client's credential were revoked)

OAuth Use Cases

Server-side web app: The server-side can securely store secrets.

Single Page Web App: Javascript code running in web browser. Cannot keep a secret.

Mobile App: storing a client secret is difficult or impossible.

Server-to-server apps with no user interaction

OAuth Use Cases & Grant Flows

Grant Flow	Use Case	Security
Authorization Code	Web app where back-end securely stores secrets.	High
Authorization Code + PKCE	SPA web apps & mobile apps	Medium
Client Credentials	Server-to-server apps	High
Resource Owner Password	First-party & trusted apps	Moderate (not recommended)

PKCE = Proof Key for Code Exchange

"Implicit Grant" flow is deprecated. Use Auth Code + PKCE

Resources

My Intro to OAuth has links to resources
To avoid duplication, I don't repeat them here.

https://cpske.github.io/ISP/authentication/oauth

Exercise 1

What sites use Google to authenticate you?

1. Go to https://accounts.google.com
2. Choose **Security**
3. Look under "**Your connections to third-party apps & services**"

How many are there?

Any that you do not use (or want)?

Exercise 2

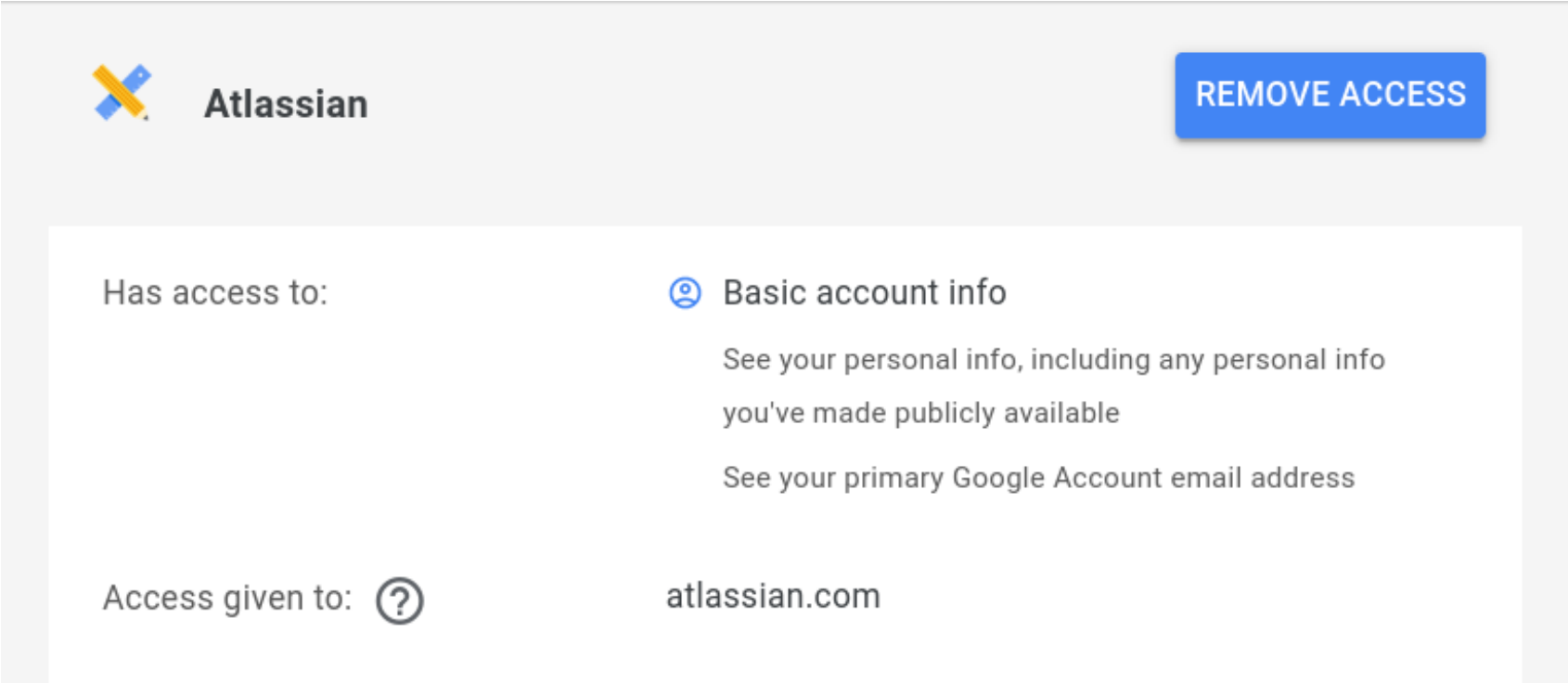
What data do you share with other apps?

This may also be in same place.

What Privileges (access) do sites have?

When a site requests OAuth access to your account, it specifies the privileges (**scope**) it wants.

Click on a site name to view details:



Hands on: OAuth Playground

`https://www.oauth.com/playground/`

Choose "Authorization Code" Flow
and work through the exercise