日不落开发工作室
sunneversets.studio

Introduction to OAuth 2.0

Sunneversets Studio Dev Share

YU Pengfei

Why

- Alice likes to Tweet with videos
- She want to use **the Studio** to edit videos
- The Studio can send videos on Alice behalf
- How should Alice and Twitter allow the Studio

Requirement

- Alice should not give password to the Studio
- The Studio can access and only access Alice’s Twitter
- Twitter can know that Alice authorizes the Studio

Solution

- Alice should not give password to the Studio
    - Alice logs in to Twitter
- The Studio can access and only access Alice’s Twitter
    - Twitter issues a specific token to the Studio
- Twitter can know that Alice authorizes the Studio
    - How ?

Roles

- Resource Owner: End user (human)
- User-agent: Browser

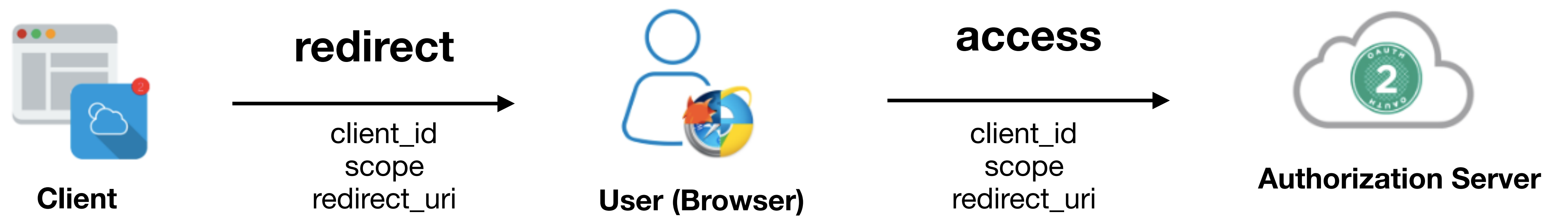
- Client

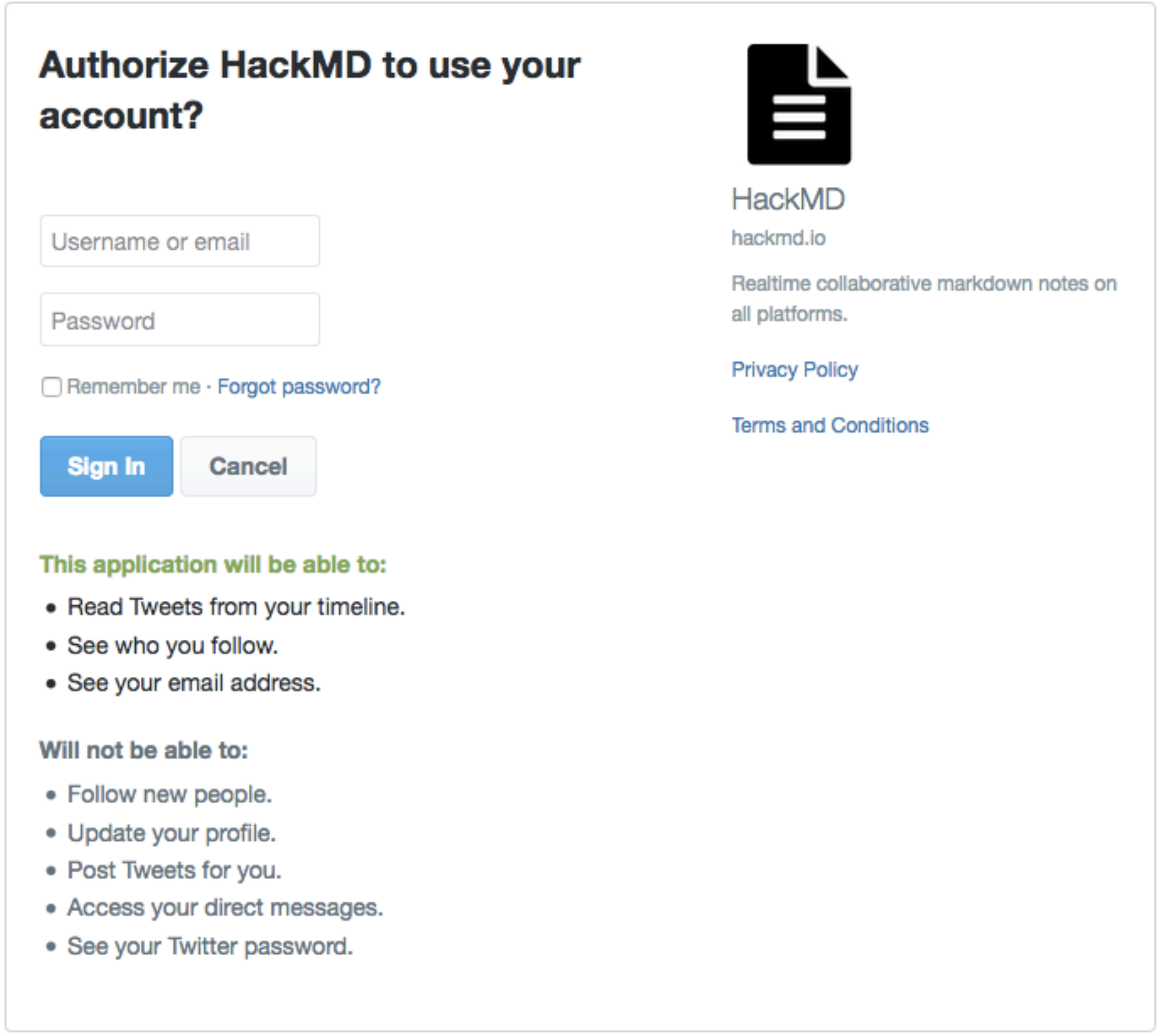
- Authorization Server
- Resource Server

Grant Flows

- Authorization Code
    - The most complete and complex grant flow
- Implicit
- Resource Owner Password Credentials
- Client Credentials

1. Client redirects User to Authorization Server





Authorize HackMD to use your account?

Username or email

Password

Remember me · Forgot password?

Sign In Cancel

This application will be able to:

- Read Tweets from your timeline.
- See who you follow.
- See your email address.

Will not be able to:

- Follow new people.
- Update your profile.
- Post Tweets for you.
- Access your direct messages.
- See your Twitter password.

HackMD
hackmd.io
Realtime collaborative markdown notes on all platforms.
Privacy Policy
Terms and Conditions

Client: I am xxx. I want an access token for yyy. If the user successfully logs in, call me at zzz.

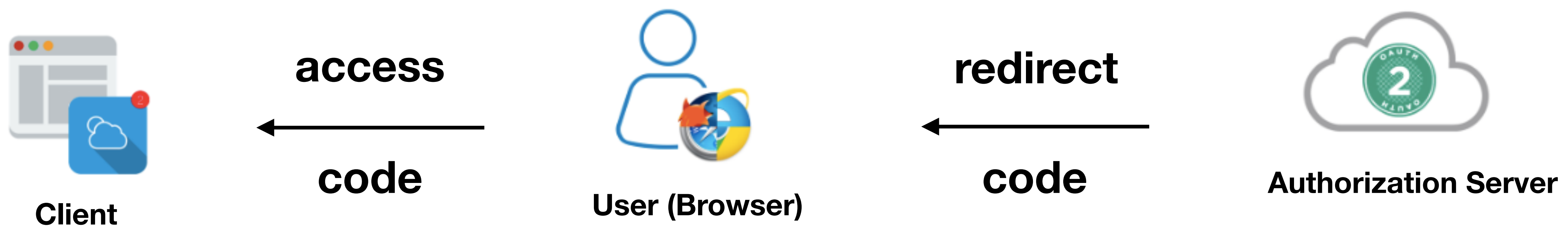
Client ID: xxx

Scope: yyy

Redirect URI: zzz

State: Random string (nonce)

2. Authorize User
3. Redirect back to Client

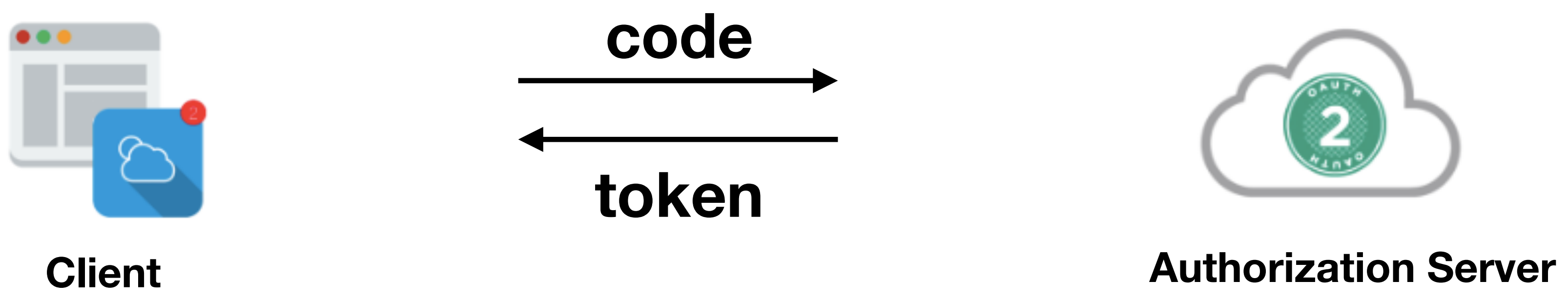


Server: The user is authorized, please use the code to exchange token.

Code: “Temporary token”, only used to request real token

State: The same as in step 1

4. Client requests Token



Client:
Code
Redirect URI
Client ID

Server:
Access Token
Refresh Token
Expiry
Scope

Refresh Token

- It never expires
- Use refresh token to get a new access token when previous one expires

Security Issues

- Attacker can be:
  - Client
  - User
  - Third Party
- Threat Model
  - User’s computer/browser can be compromised by passive attacker
  - Client’s server (if it has) and Authorization server is safe
  - Transportation can be eavesdropped, but we just use TLS

Credential Guessing

- Attacker can guess:
  - Code
  - Access Token
  - Refresh Token
- Solution
  - Use random string with enough length (>128 bit)
  - Code has timeout
  - Code can only be used once

Client Impersonation

- Malicious client pretends to be another client
- In Step 1 (initiate), 4 (code -> token)
- Solution:
  - Check redirect URI
  - Check client credential in step 4 (code -> token)

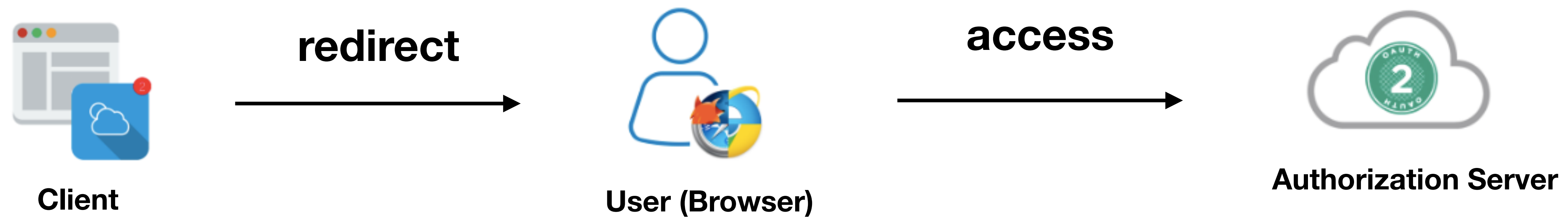
CSRF
(cross-site request forgery)

- Let User use Attacker’s access token
    - User may store sensitive information (e.g. credit card) to Attacker’s account
- Attacker tricks User to click redirect link that leads to Attacker’s code (in step 3 (redirect back))
- Solution
    - Use state in step 1 (initiate)
    - Check state is the same in step 3 (redirect back)

Implicit

- Similar to Authorization Code Grant Flow
- Server responses with token instead of code

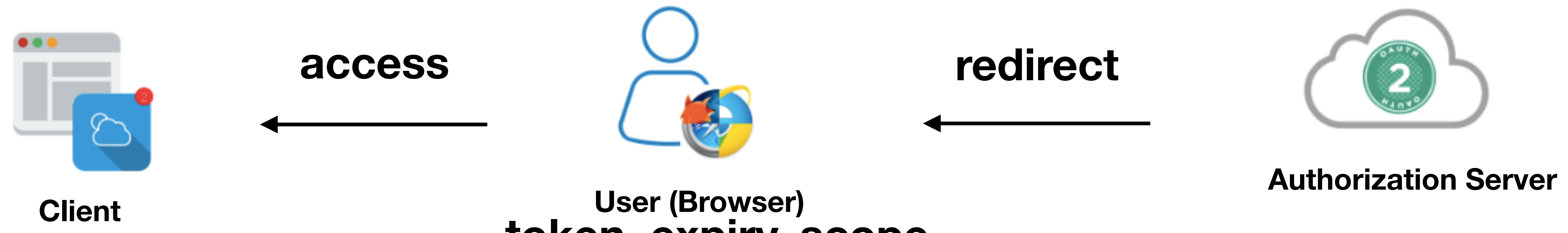
1.



client_id, scope, redirect_uri

2. User enters username and password

3.



token, expiry, scope

Client Type

- Confidential: Can protect its secret
    - Website
- Public: Cannot protect its secret
    - Desktop/Mobile App
    - SPA (Single Page App) without backend

Security Issue for Public Client

- Public clients cannot protect its secret, so
  - client credentials can be stolen
  - code/token can be stolen

PKCE Extension

- Proof Key for Code Exchange
- Used in Authorization Code Grant Flow for Public Client
- Steps
  - 0. Generates v=rand_str(), h=sha_256(v)
  - 1. (Step 1 (initiate)) send h to authorization server
  - 2. (Step 4 (code->token)) send v to authorization server

Resource Owner Password Credentials

- Client will get User’s password
- User needs to trust Client
- Spec says Client should not store password, but…
- Steps
    1. User gives username & password to Client
    2. Client sends them to Authorization Server
    3. Server responses with token

Client Credentials

- Authorization Server trust Client
- Nothing to do with User

Memes

- Alice: a character commonly used in cryptographic story (https://en.wikipedia.org/wiki/Alice_and_Bob)
- Twitter: Twitter OpenID (kind of) inspires the development of OAuth; Twitter is also a pioneer to use OAuth

Further Reading & Reference

- OAuth 2.0 筆記 (https://blog.yorkxin.org/2013/09/30/oauth2-1-introduction.html)
- OAuth - Wikipedia (https://en.wikipedia.org/wiki/OAuth)
- https://www.oauth.com/
- PKCE (https://www.oauth.com/oauth2-servers/pkce/)

Image Reference

- https://developer.okta.com/assets/blog/oauth/oauth-actors-cd8b4861e839037400d8521e97c5d8cf0cb029add65d1036488991c7e85dcb72.png

- https://upload.wikimedia.org/wikipedia/zh/thumb/9/9f/Twitter_bird_logo_2012.svg/1200px-Twitter_bird_logo_2012.svg.png

- https://img.informer.com/icons/png/128/5314/5314620.png