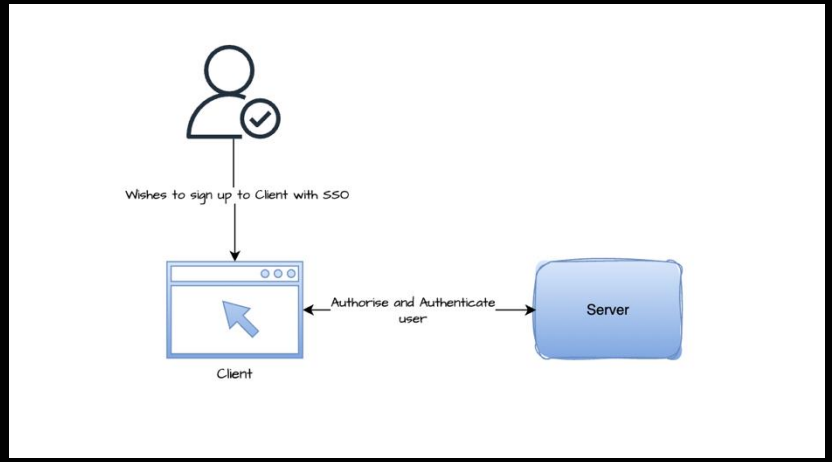
Building OAuth from First Principles

IdentityShield Summit '25

Manali Kaswa

What is OAuth?

- To allow an application to access resources hosted by other applications on behalf of a user
  - **Social Login**: "Sign in with Google"
  - **API Access/Data Sharing**
- **Solves:**
  - **Authorization: What are you allowed to do?**
  - **Authentication: Who are you? User identity information. (OpenId Connect)**
- **Est. 2012**



Live in action

Why it matters?

- **Websites: 10M implement OAuth for social login**
- **Mobile Apps: likely 1M, integrate OAuth**
- **APIs and Enterprise Systems: 100K**

Source: ChatGPT

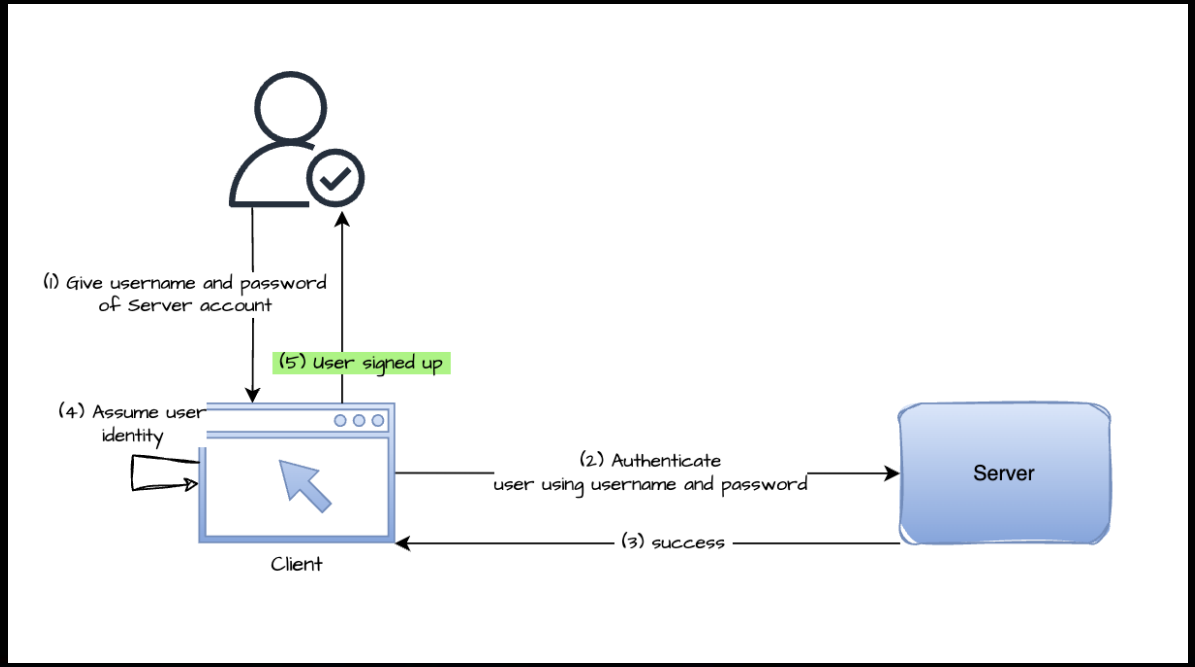
Who Manali?

- पुणेकर
- Cloud engineer at Espressif Systems
- Primarily work in AWS and DevOps
- I managed to build an entire Identity system in Golang (that too the serverless way!)

Our end goal

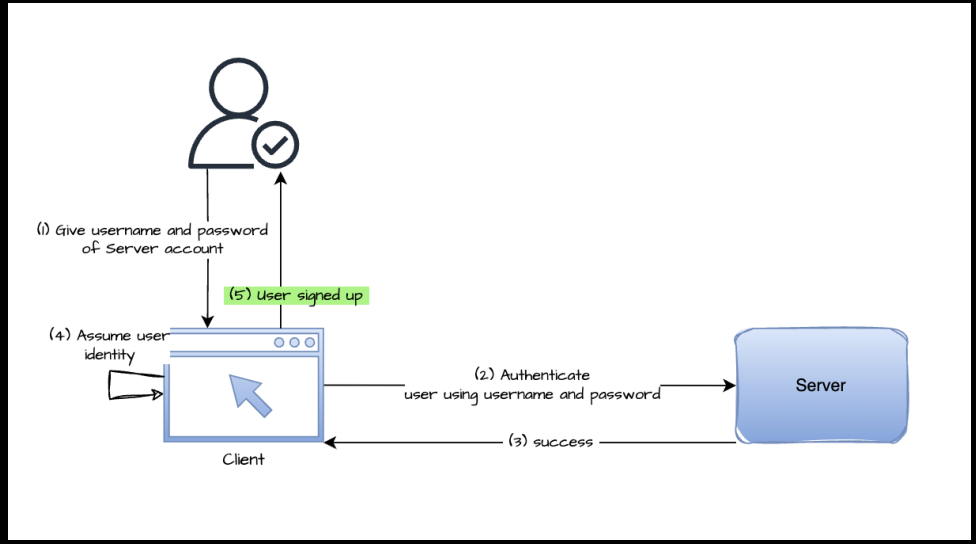
We want to build a solution to allow an application to access resources hosted by other applications on behalf of a user

Solution 1: The user shares their credentials with the client



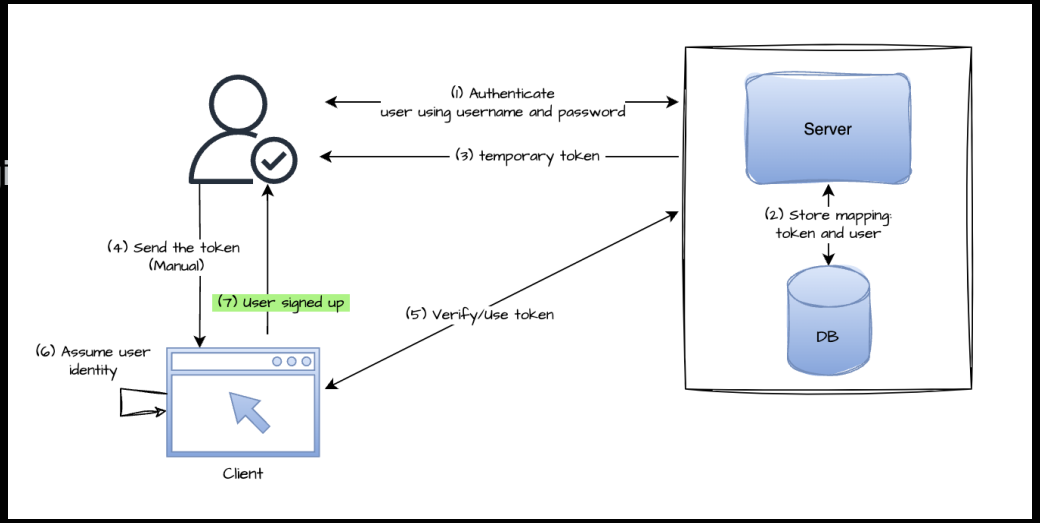
Solution 1: The user shares their credentials with the client

- **Problems:**
  - **Exposure of Credentials**
  - **Trust Issues**
  - **No Granular Access**
  - **Difficulty in Revocation**



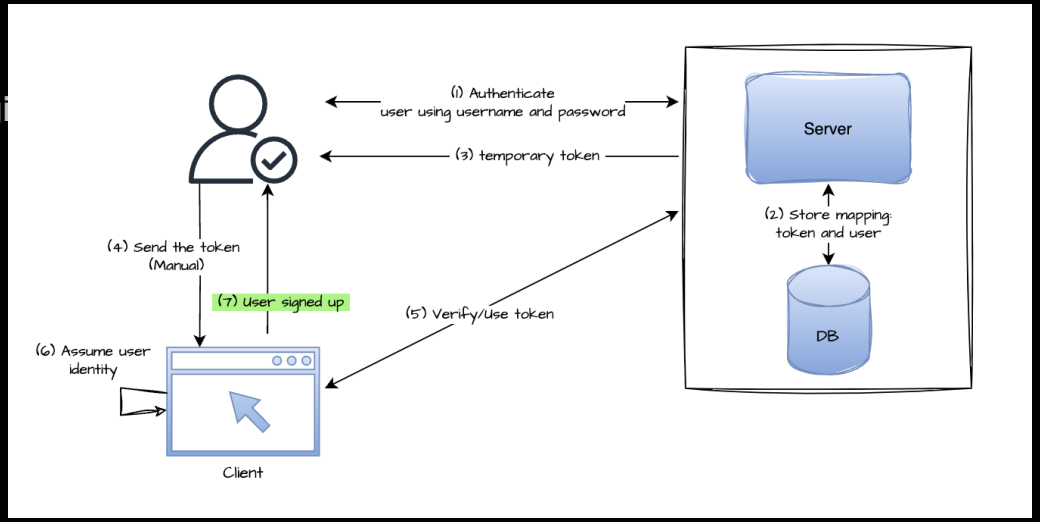
Solution 2: The user shares a temporary token with the client

- **What:**
  - **Token: unique string, has expiry, regi serves**

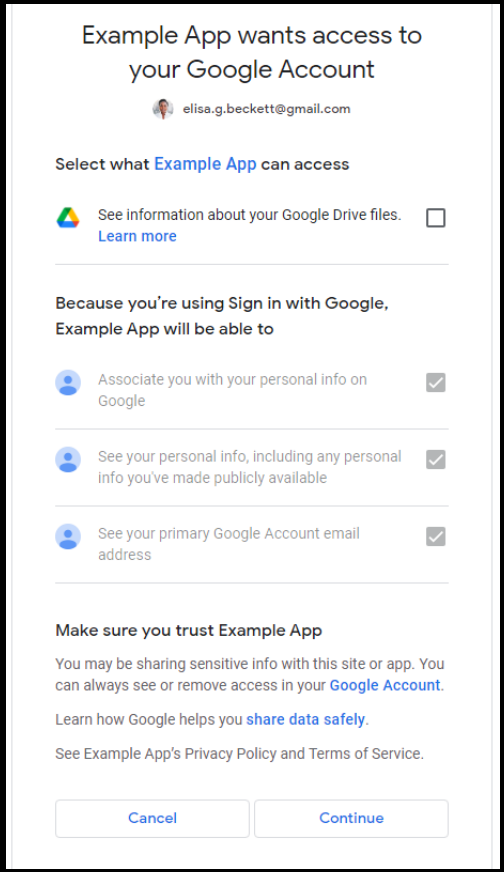


Solution 2: The user shares a temporary token with the client

- **What:**
  - **Token: unique string, has expiry, regi server**
- **Solves:**
  - **Exposure of Credentials**
  - **Trust Issues**
- **Problems:**
  - **No Granular Access**
  - **Bad user experience**
  - **Token may get misplaced. Who is responsible?**
  - **Any other client can impersonate using the token**
  - **Difficulty in revocation**

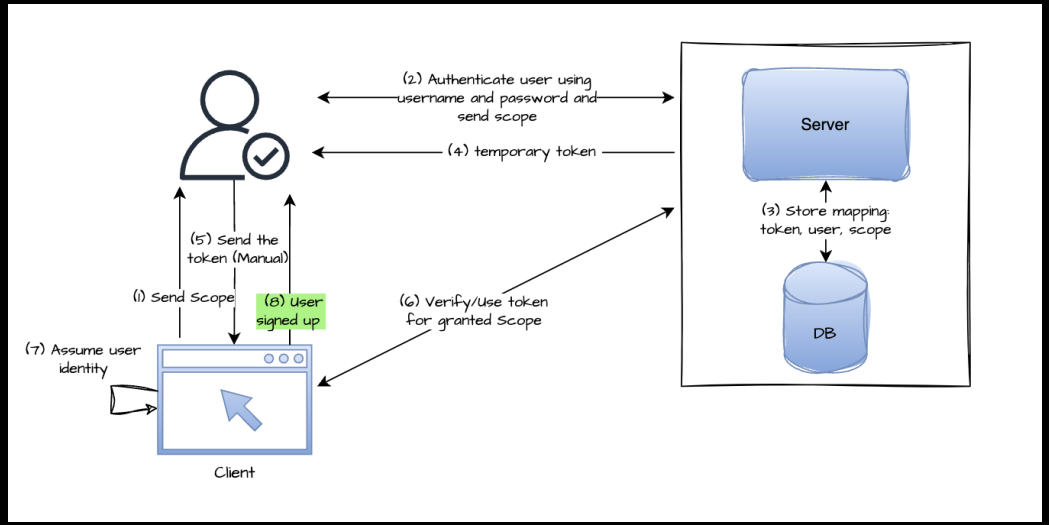


Edit: Scope


Example App wants access to your Google Account
elisa.g.beckett@gmail.com
Select what Example App can access
See information about your Google Drive files.
Learn more
Because you're using Sign in with Google, Example App will be able to
Associate you with your personal info on Google
See your personal info, including any personal info you've made publicly available
See your primary Google Account email address
Make sure you trust Example App
You may be sharing sensitive info with this site or app. You can always see or remove access in your Google Account.
Learn how Google helps you share data safely.
See Example App's Privacy Policy and Terms of Service.
Cancel
Continue

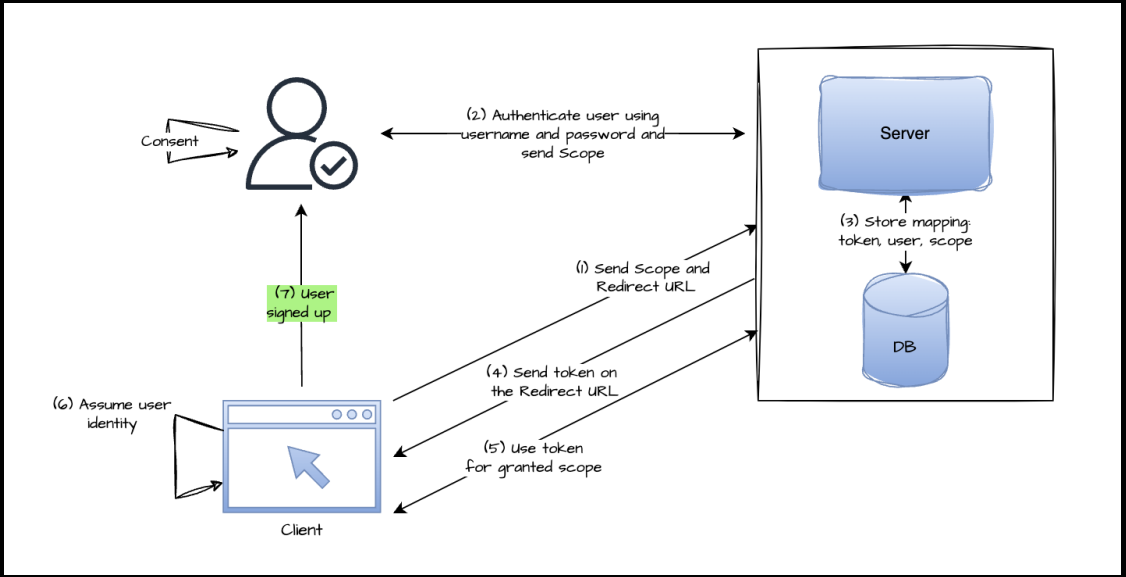
Solution 3: The user shares a temporary token with the client with scope

- **Solves:**
  - **No Granular Access**
- **Problems:**
  - **Bad user experience**
  - **Token may get misplaced. Who is responsible?**
  - **Any other client can impersonate using the token**



Solution 4: The client gets a temporary token using user's login and consent: Redirect URL

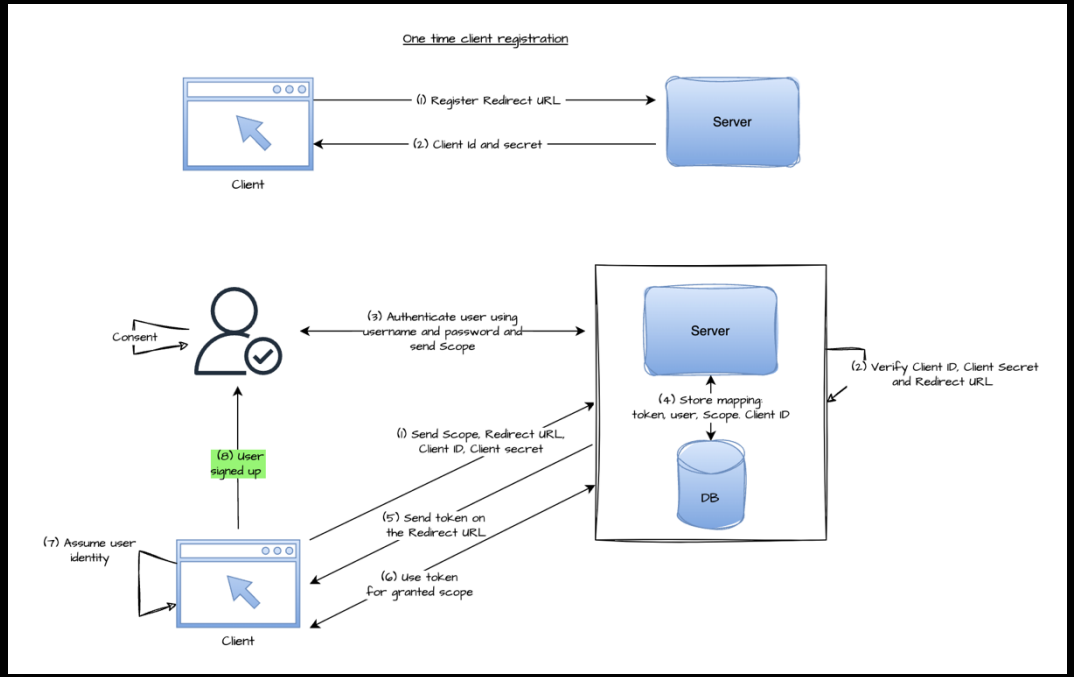
- **Solves:**
  - **Bad user experience**
  - **Token may get misplaced. Who is responsible?**
- **Problems:**
  - **Any other client can impersonate using the token**



Tip: Use HTTPs

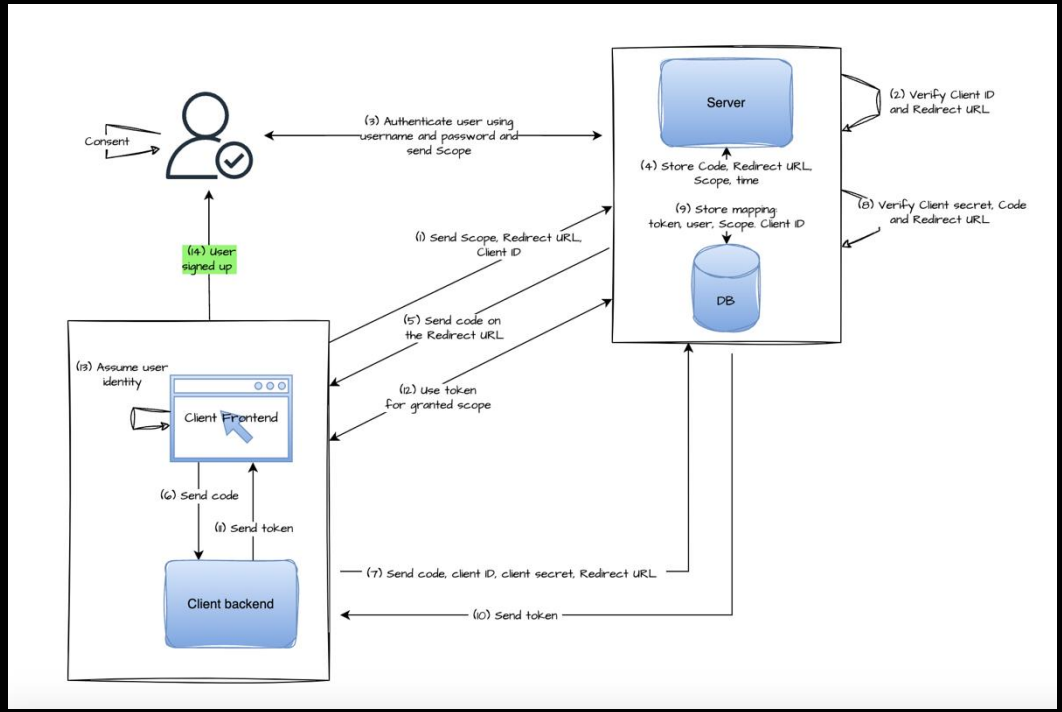
Solution 5: Client registration with Redirect URL, Client Secret

- **What:**
  - Redirect URL registration prevents attacker to redirect the token to any site
  - Client secret securely identifies the client
- **Solves:**
  - Any other client can impersonate using the token
- **Problems:**
  - Frontend cannot store client secret securely



Edit: Code

- What
  - need for code to act as an intermediary – one time use, expires
- Solves:
  - Frontend cannot store client secret securely



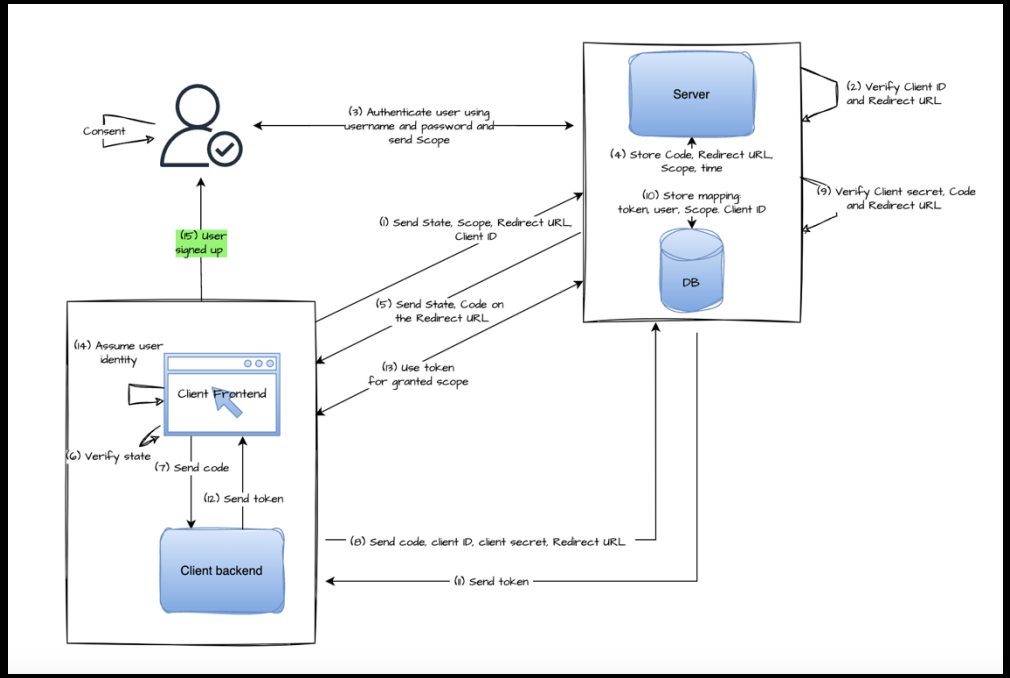
Solution 6: Client registration with Redirect URL, Client Secret and code introduced

- **Problems:**
  - **CSRF: The attacker might start an authorisation request and login themselves. The attacker will send the code to the backend themselves and get real token. This might lead to data leakage as the client believes the attacker's access token to be user's.**
  - **On expiry the user has to login again**
  - **No authentication**
  - **Server needs to manage the tokens issued for scope, expiry, client id**
  - **Secret storage not possible for SPA**
  - **XSS: Code may get intercepted**

Tip: Anything that goes into URL (fragment, query parameter) is susceptible to XSS

Edit 1: State

- What
  - A random value sent by the client during the authorization request and returned by the server. Confirms that the client initiating the request is the one receiving the response.
- Solves:
  - CSRF: The attacker might start an authorisation request and login themselves. The attacker will send the code to the backend themselves and get real token. This might lead to data leakage as the client believes the attacker's access token to be user's.



Edit 2: Refreshtoken

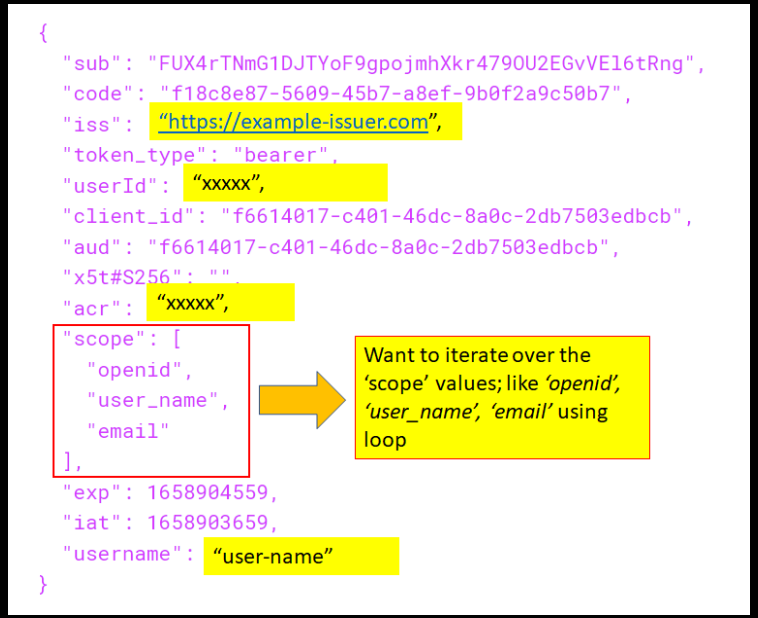
- **What**
  - **Can be used to get accesstoken from refreshtoken**
  - **Access tokens can have shorter lifespans, minimizing the risk of misuse**

- **Solves:**
  - **On expiry the user has to login again**

Tip: You can revoke refreshtokens

Edit 3: OpenId Connect

- What
  - Idtoken – JWT (Signed encoded token)
    - Must uniquely identify user
    - Can contain user specific info like email, phone number, etc.
  - To avoid tampering: Must be signed and the public key should be available publicly
- Solves:
  - No authentication



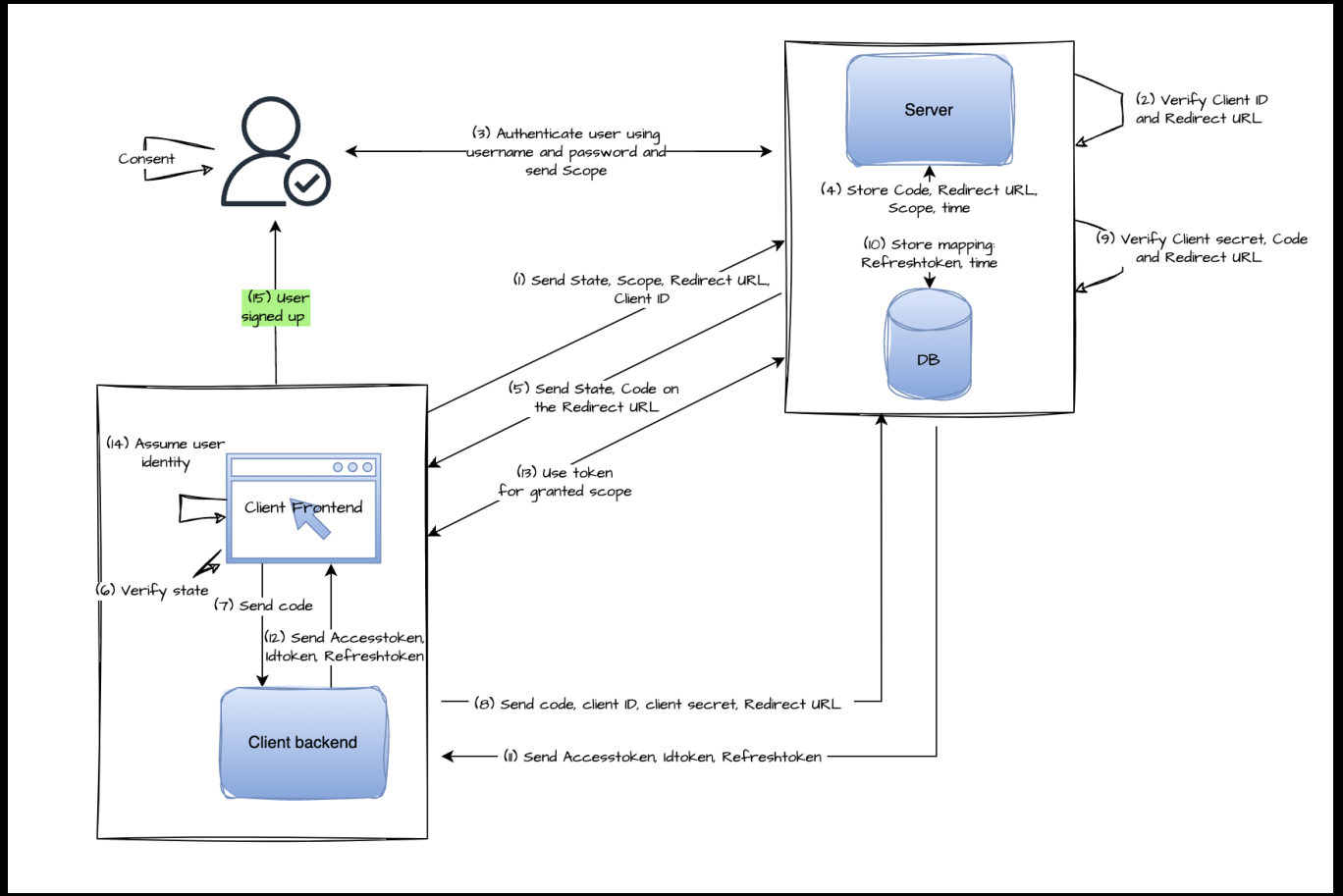
Tip: Always verify signature of tokens before use

Edit 4: JWT accesstoken (Not mandatory)

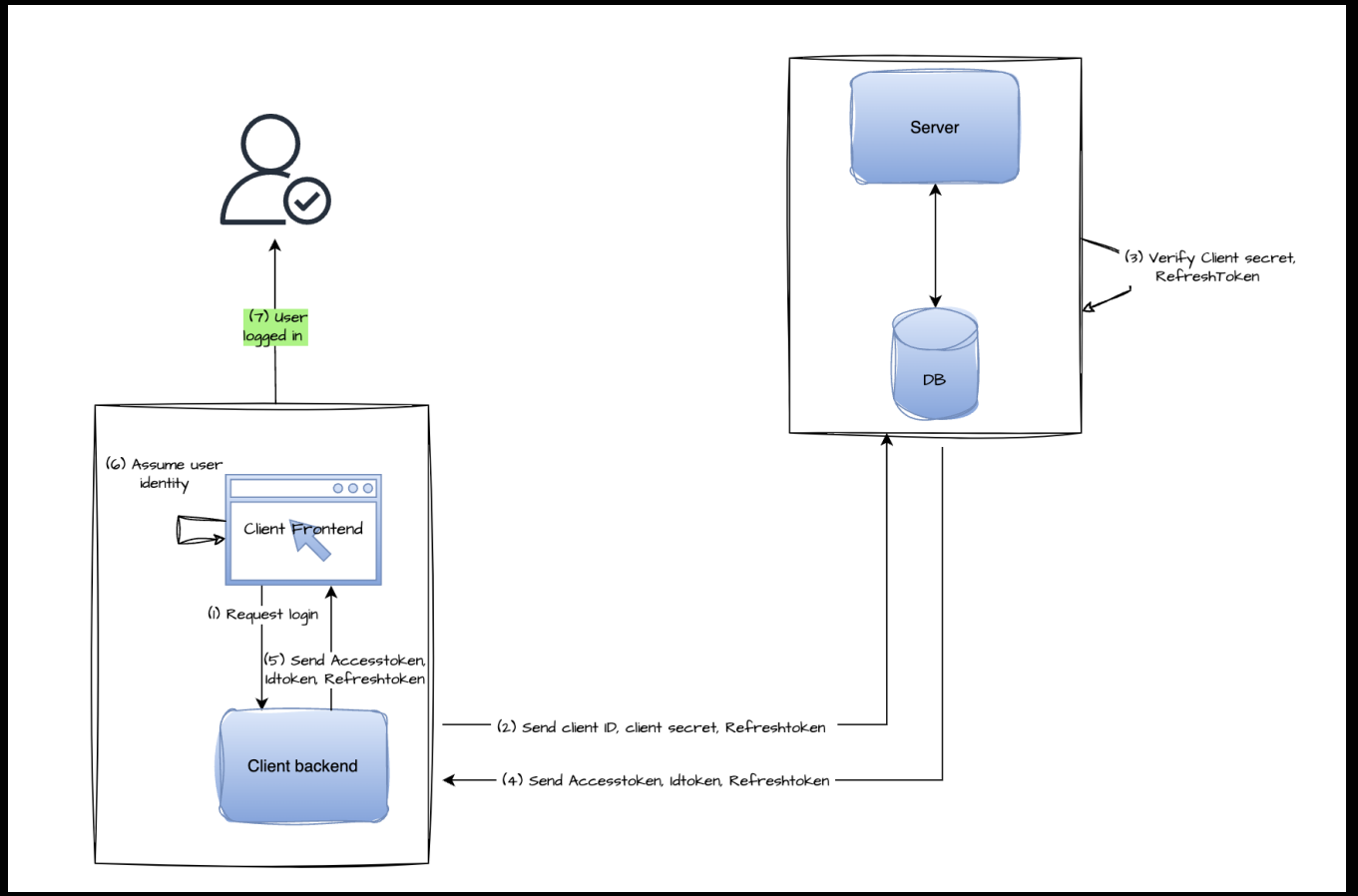
- **What**
  - **JWT accesstoken**
- **Solves:**
  - **Server needs to manage the tokens issued for scope, expiry, client id**

Tip: Use SameSite HttpOnly cookies to store

Sign-up


Server
(2) Verify Client ID and Redirect URL
Consent
(3) Authenticate user using username and password and send Scope
(4) Store Code, Redirect URL, Scope, time
(10) Store mapping Refreshtoken, time
(9) Verify Client secret, Code and Redirect URL
DB
(15) User signed up
(1) Send State, Scope, Redirect URL, Client ID
(5) Send State, Code on the Redirect URL
(13) Use token for granted scope
(14) Assume user identity
Client Frontend
(6) Verify state
(7) Send code
(12) Send Accesstoken, Idtoken, Refreshtoken
Client backend
(8) Send code, client ID, client secret, Redirect URL
(11) Send Accesstoken, Idtoken, Refreshtoken

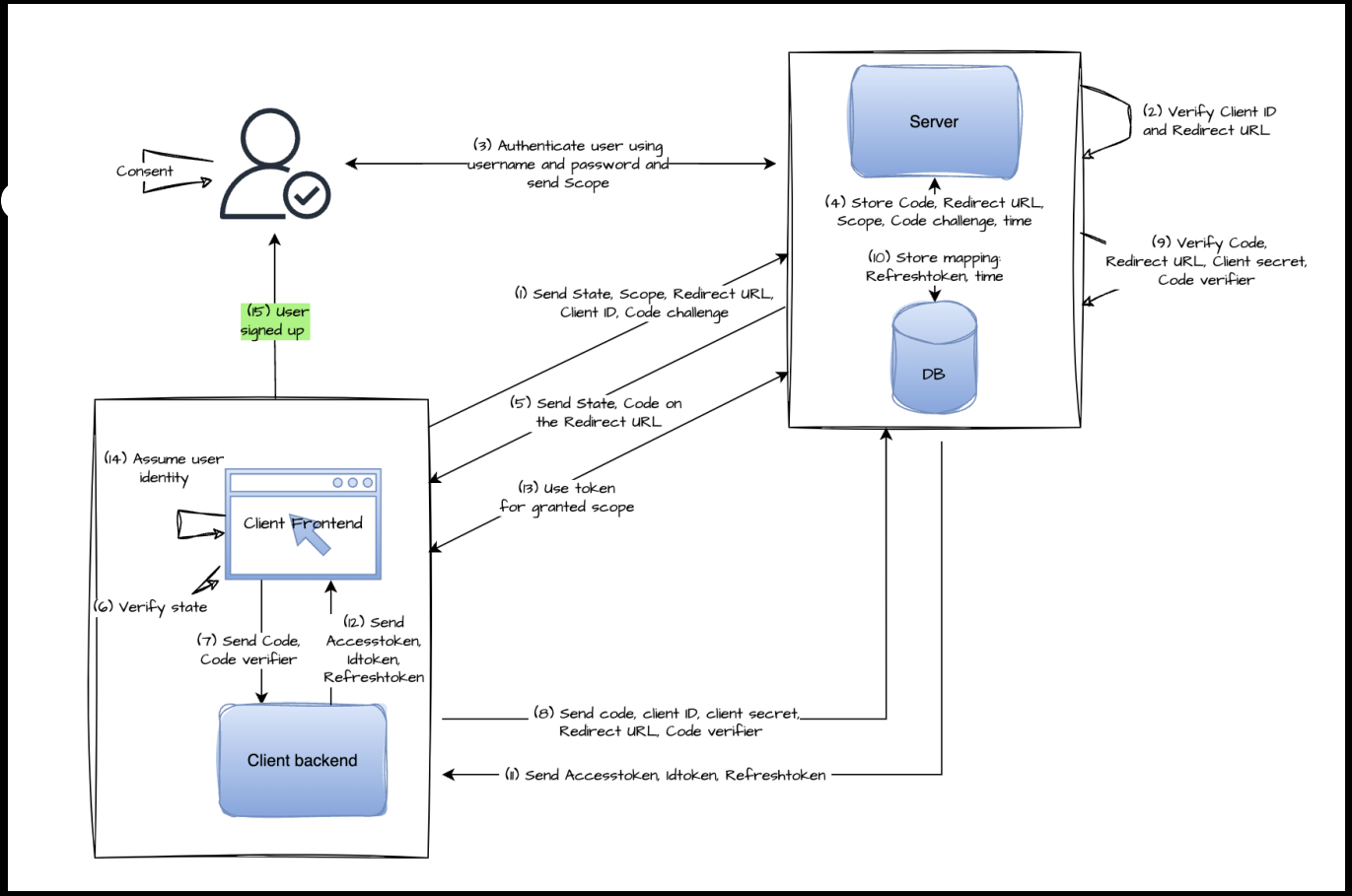
Login


Server
DB
(3) Verify Client secret, RefreshToken
(7) User logged in
(6) Assume user identity
Client Frontend
(1) Request login
(5) Send Accesstoken, Idtoken, Refreshtoken
Client backend
(2) Send client ID, client secret, Refreshtoken
(4) Send Accesstoken, Idtoken, Refreshtoken

Edit 5: PKCE (Best practice)

- What
  - Client generates code verifier and code challenge (hashed code verifier) – use once
  - code challenge -> authorise request – Tie it to the user in client (So the attacker cannot send their malicious code challenge, and thereby obtain token)
  - code verifier -> token request which the server then verifies
- Solves:
  - Secret storage not possible for SPA
  - XSS: Code may get intercepted

Solution 7 (The end go



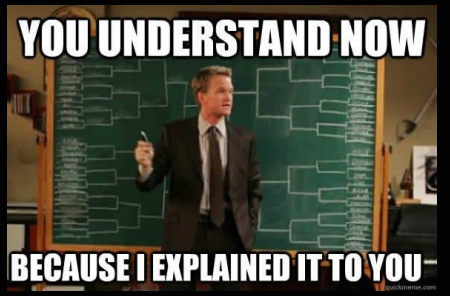
What OAuth grants we haven’t covered?

- Implict Flow (Legacy - less secure)
- Password (Legacy - less secure)
- Device code
- Client credentials

Tip: Use SameSite HttpOnly cookies to store

Live in Action again!

Do you understand OAuth and OpenID Connect?



Questions?

Helpful tools and references

- RFCs:
  - OAuth2: 6749 Best Practices
  - JWT 7519
- Specs:
  - OpenId Connect
  - Discovery
- Compiled resources: https://oauth.net/2

- Tools:
  - JWT verifier: https://jwt.davetonge.co.uk/
  - Public key PEM to JWK: https://8gwifi.org/jwkconvertfunctions.jsp

Manali Kaswa

AWS | Identity | Go | DevOps | FinOps

Thank you and have fun!