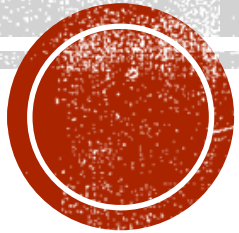
Gladwin Burboz [Speaker]

https://linkedin.com/in/gladwinb

OAUTH/OIDC

UNDER THE HOOD OF SOCIAL LOGIN

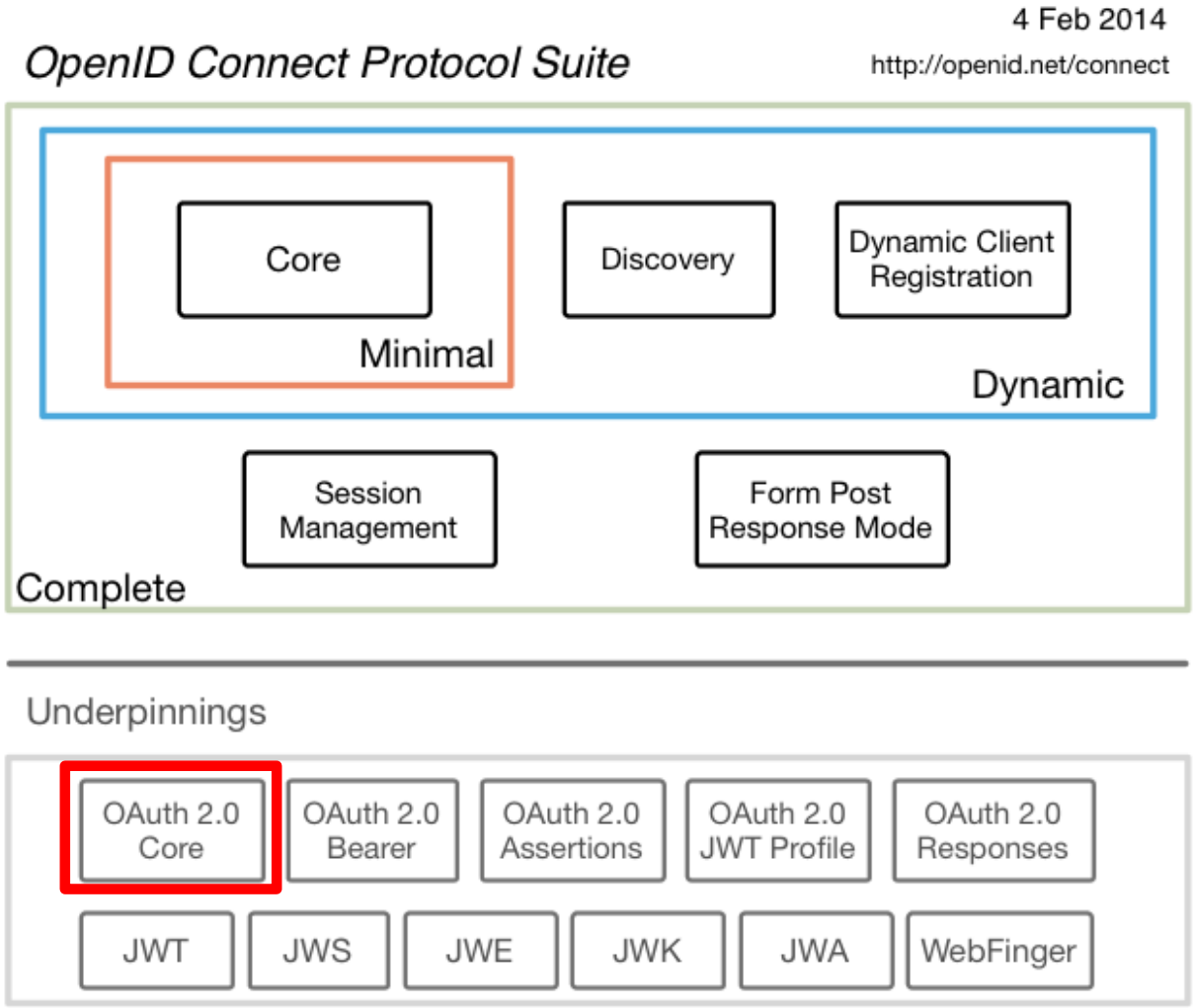


DISCLAIMER: NON AFFILIATION & LIABILITY WAIVER

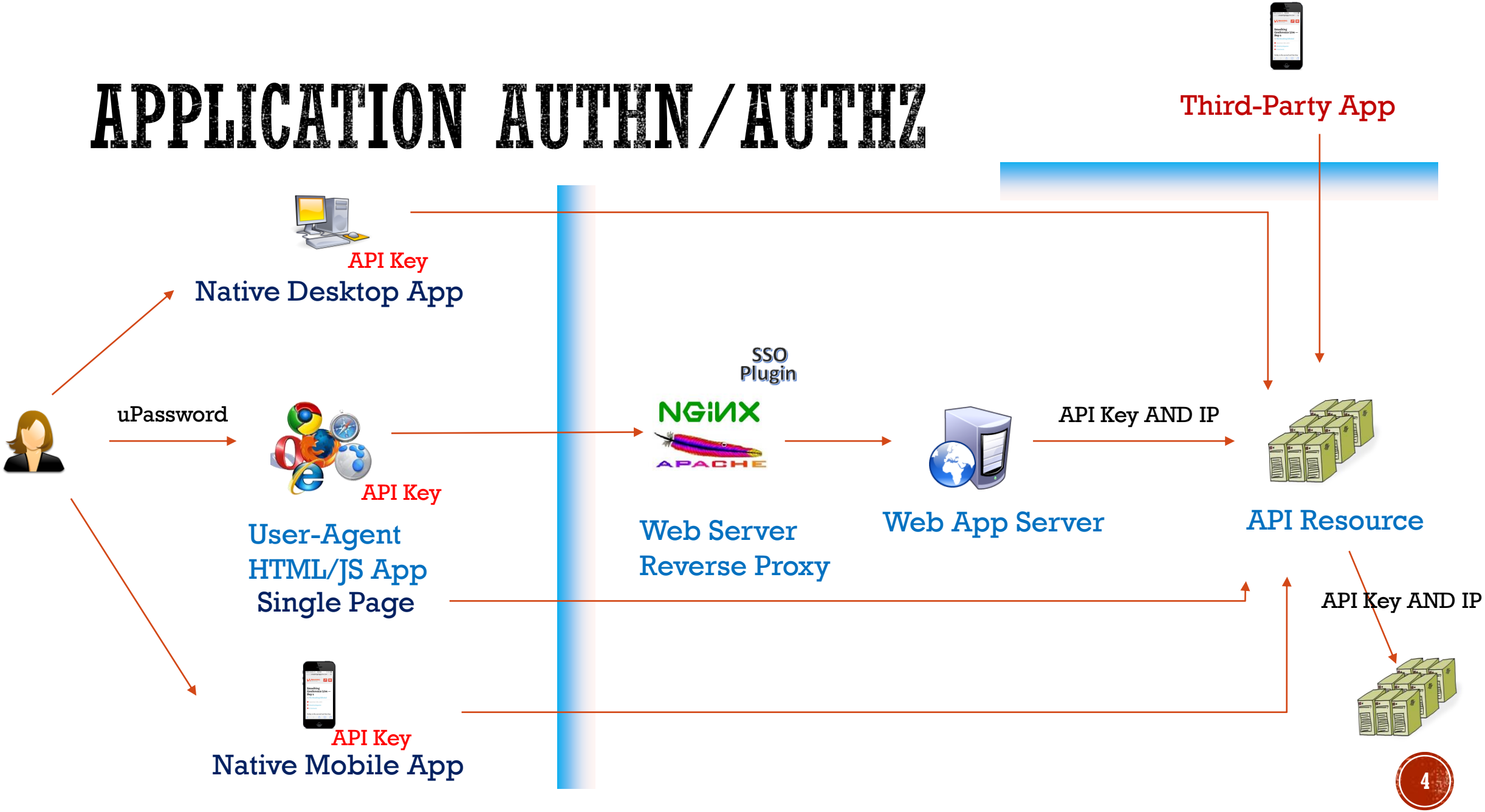
- My current or past employers has no connection or liability towards this presentation
- I am not affiliated with any of the organizations or products discussed in this presentation
- This presentation is for educational purpose only with no guarantee of accuracies with specifications
- Screenshots have been modified to present concepts

OAUTH 2.0 & OPENID CONNECT 1.0 SPEC

- The OAuth 2.0 [rfc6749]
  - https://tools.ietf.org/html/rfc6749
- Open ID Connect 1.0
  - http://openid.net/connect/



APPLICATION AUTHN/AUTHZ



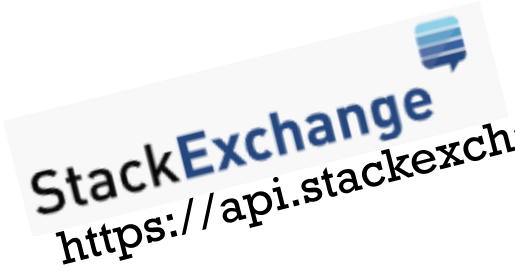
HOW DO THEY EXPOSE APIS?

https://developers.google.com/

https://developer.linkedin.com/

https://developers.facebook.com/

https://developer.github.com/



https://api.stackexchange.com/

https://developer.twitter.com/

CONSIDER A USE CASE

Emca Fraud
Protection service

Emca a popular startup company provides fraud monitoring and protection service to banks **on behalf of** bank account owner

Emca needs from Bank

- Transaction logs API
- Stop transaction API

Acme Bank

Emca requires from A/C owner

- Account to be monitored
- Bank login credentials
- Or permission for account

A/C Owner

NEED FOR ALTERNATE AUTHORIZATION

- Can Account Owner trust credentials with Emca?
- Can we limit Emca's access to read user's checking account and not his mortgage or trading accounts?
- Can we have a mobile app that can notify of events
  - What about session expiration
  - What if user restarts the phone

Emca

Acme Bank

A/C Owner

NEED FOR ALTERNATE AUTHORIZATION

- Can Account Owner trust credentials with Emca?
- Can we limit Emca's access to read user's checking account and not his mortgage or trading accounts?
- Can we have a mobile app that can notify of events
  - What about session expiration
  - What if user restarts the phone

Emca

Acme Bank

A/C Owner

Dec 2007 : OAuth 1.0
Oct 2012 : OAuth 2.0

BIRTH OF OAUTH

- OAuth is open standard authorization framework for access delegation

Authorizing third-party limited access to user resources

OAUTH 2.0 WITH SPRING SECURITY

- OAuth 2.0 Support, within the Spring projects portfolio, is spread out between Spring Security OAuth, Spring Cloud Security, Spring Boot 1.5.x

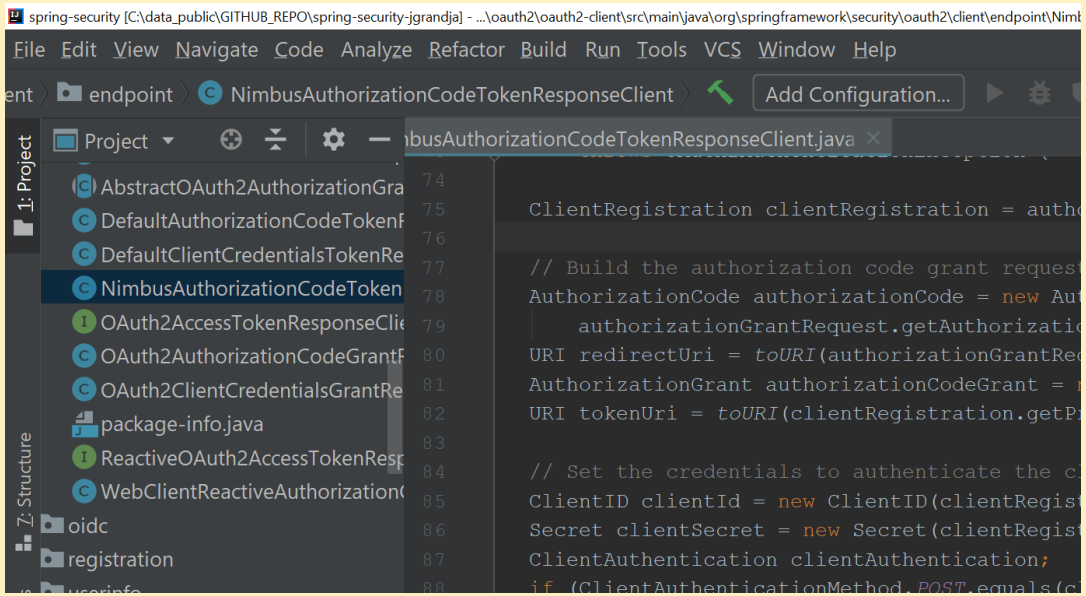
- New support introduced in **Spring Security 5**
  - More extensive support for OAuth 2.0 and OpenID Connect 1.0
  - New Client support while, Resource Server and Authorization Server coming soon

https://spring.io/blog/2018/01/30/next-generation-oauth-2-0-support-with-spring-security

SPRING BOOT DEMO

https://github.com/gburboz/gb-oauth2-springboot-talk

- Step-01-InitialSpringSecurityAppp
- Step-02-OAuthSimpleSocialLogin



Lets try some Social Login code!

- **Spring Boot 2.x**
- **Spring Security 5**

OAuth 2.0 Login Sample

HOTEL RESERVATION FLOW ANALOGY

Hotel Staff

Property

2. Req Reservation >
| Reserve. Code <

3a. Reserve. Code >
| Access Card <

4. Use w/ Access Card >
| Gets Room Access <

Superman
Property Owner

Sales
Agent

1 – Charge your Account > | Need Reserve. Code <

3. Reserve. Code > | Ok w/ Charge on Account <

Batman
Client

UNDERSTANDING OAUTH

On behalf of property owner (1)

hotel staff (2) authorizes delegated access

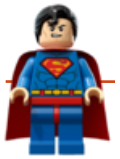
of hotel room and facilities resources (3)

to it's guest client (4)

with limited access card (5)

OAUTH ROLES

Superman
(Property Owner)



Resource owner
(the user)

Hotel Staff
(hotel front desk reception)

Authorization server
(can be the same server as the API)

Property
(rooms and facilities)

Resource server
(the protected API)

Batman Client
(hotel guest)

Client
(your app)

Access Card

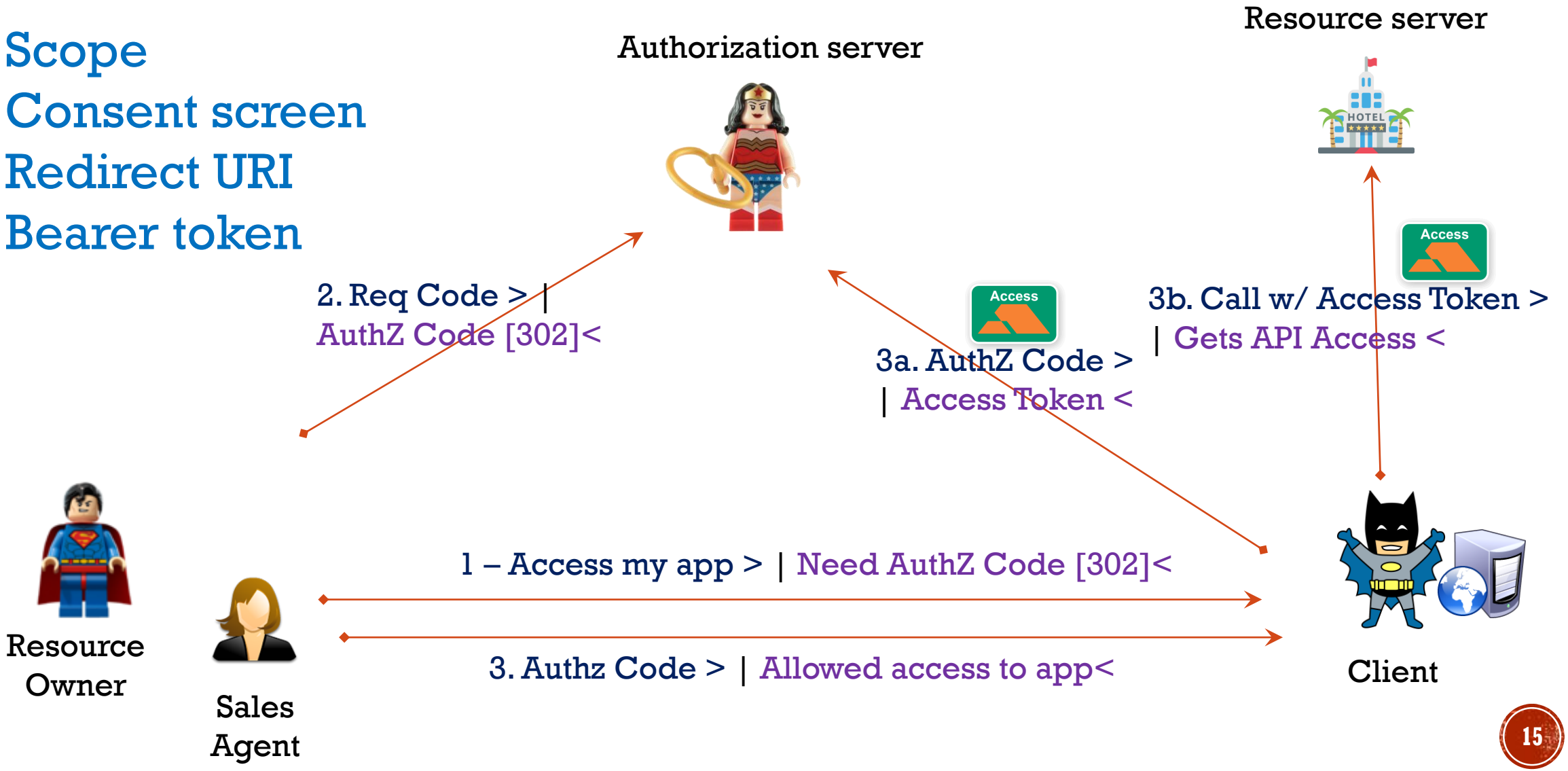
OAuth Token

Sales Agent

User Agent

OAUTH: AUTHORIZATION CODE FLOW

- Scope
- Consent screen
- Redirect URI
- Bearer token



OAUTH: AUTHORIZATION CODE FLOW

BREAK

Authorization server

Resource server

2. Req Code > |
AuthZ Code [302]<

3a. AuthZ Code >
| Access Token <

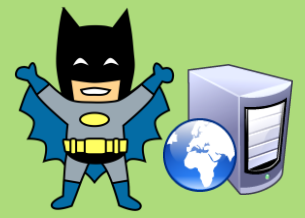
3b. Call w/ Access Token >
| Gets API Access <

Resource Owner

Sales Agent

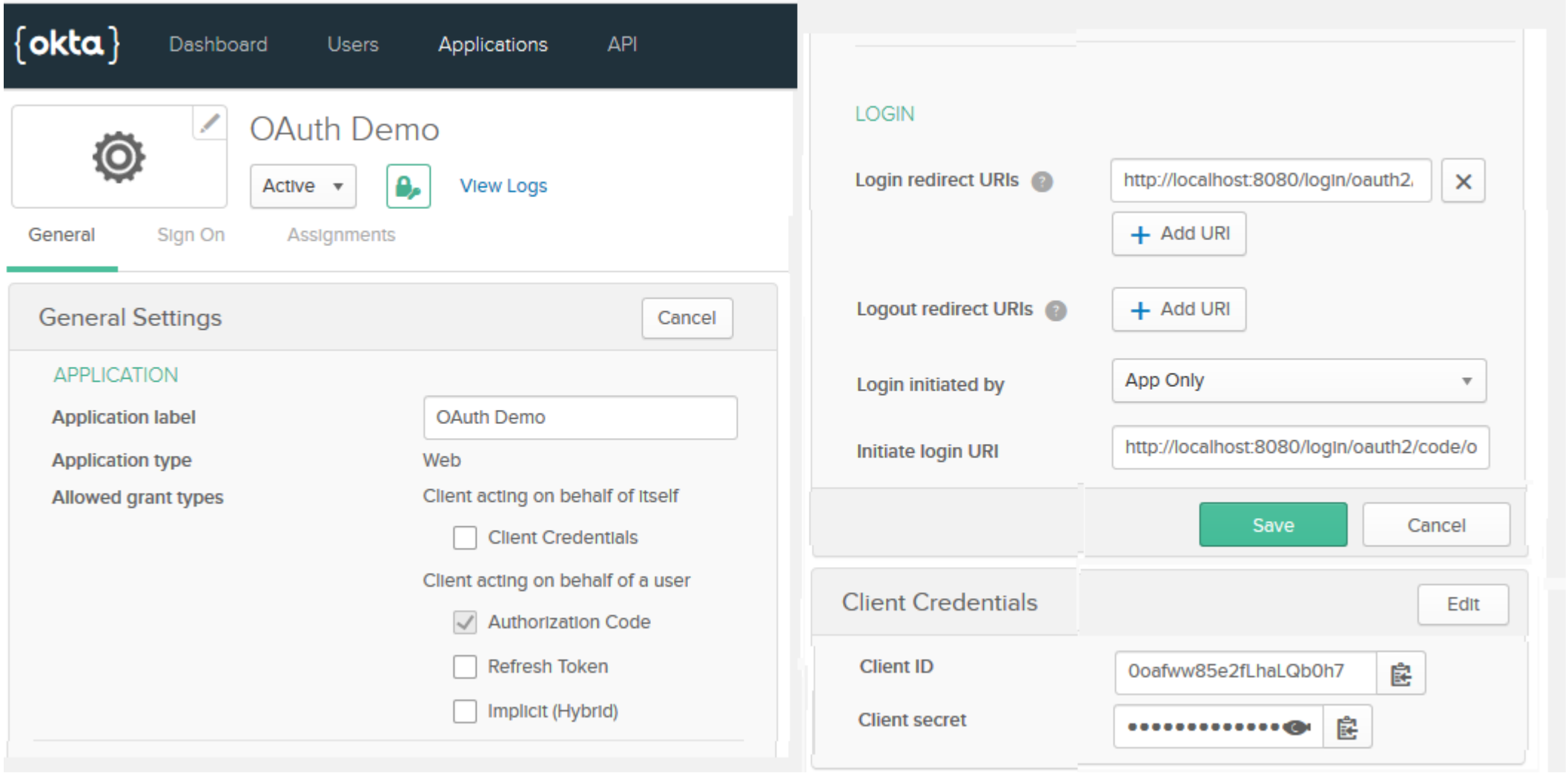
1 – Access my app > | Need AuthZ Code [302]<

3. Authz Code > | Ok w/ Charge on Account <



Client

CLIENT REGISTRATION



REDIRECT URL

- After a user authorizes an application, the authorization server will redirect the user back to this URL
- Redirect URL will be appended with sensitive information, so it is critical it is not arbitrary/open

SCOPE

- provide a way to limit the amount of access that is granted
  - read , write, admin, profile, email,

OAUTH CONSENT SCREEN

- Authorization server explicitly asks user if listed permissions can be given to the Client

- Who the Client is (from registration)
- Permission List (from scope)
- For how long will this be for
- How can it be revoked if need be

http://tools.ietf.org/html/rfc6750

BEARER TOKEN

- Bearer Tokens are the popular Access Token used with OAuth 2.0
- Any party in possession of a bearer token (a "bearer") can use it to get access
- It is opaque string with no meaning to it's client
- Some may use string of random characters like UUID
- Others may use more structured tokens like SAML or JWT

CONFUSING, TERM "CLIENT"

also called as
Relying Party

- Client is an application that
  - requests OAuth Token from provider and
  - uses it to access Resource on behalf of user.

  - Batman is client application that requests OAuth Token from Wonder Women

  - Batman uses token on behalf of Superman to access his protected resource

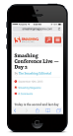
- Client is registered with OAuth Provider

CONFUSING, TERM "CLIENT"

- Client can be anywhere depending on use case

Web App Server

Browser JS App



Native Mobile App

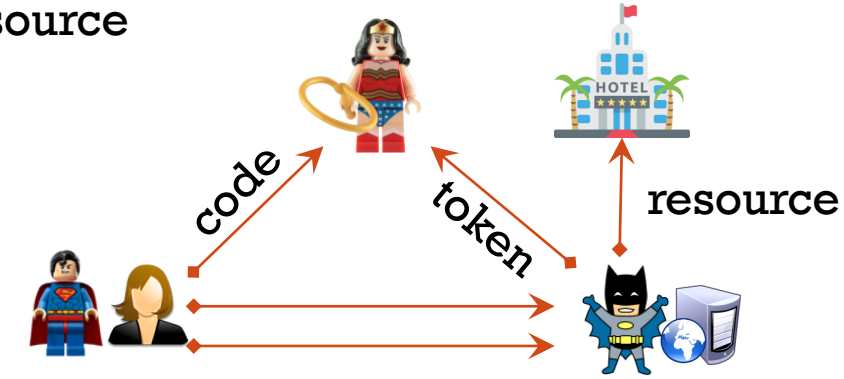
Native Desktop App

API Consumer

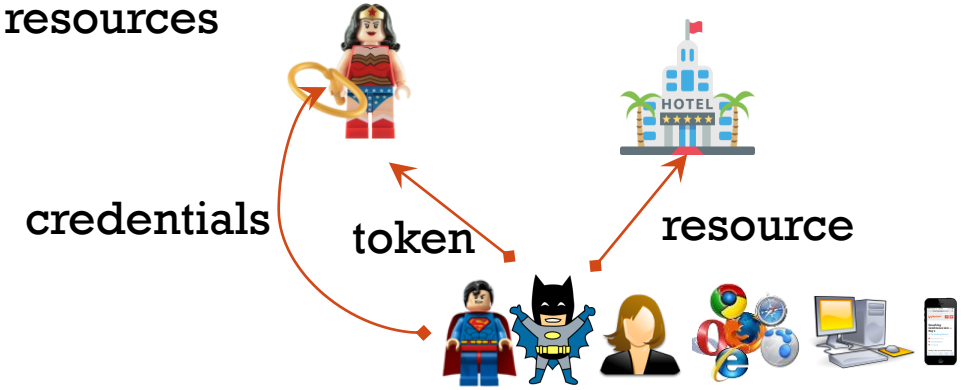
- Example
  - MeetUp.com is Client (App that uses SSO)
  - Person that logs in is a User
  - Google is Authorization Server
  - User's google profile URL is protected Resource

OAUTH FLOWS

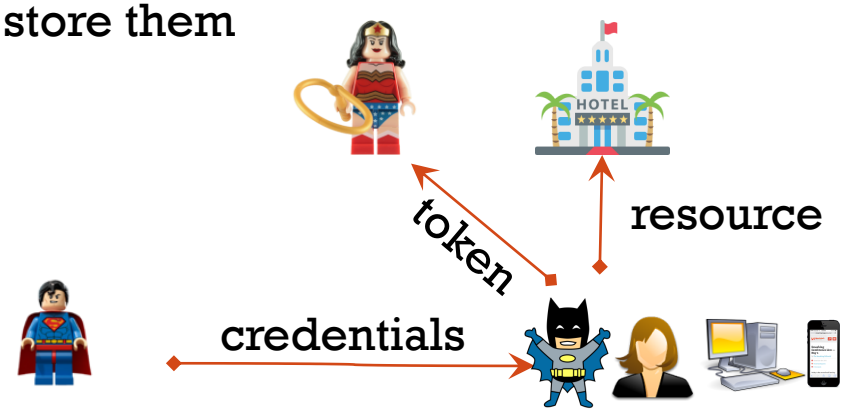
- Authorization Code Grant
  - App on Server side needs access to user's resource



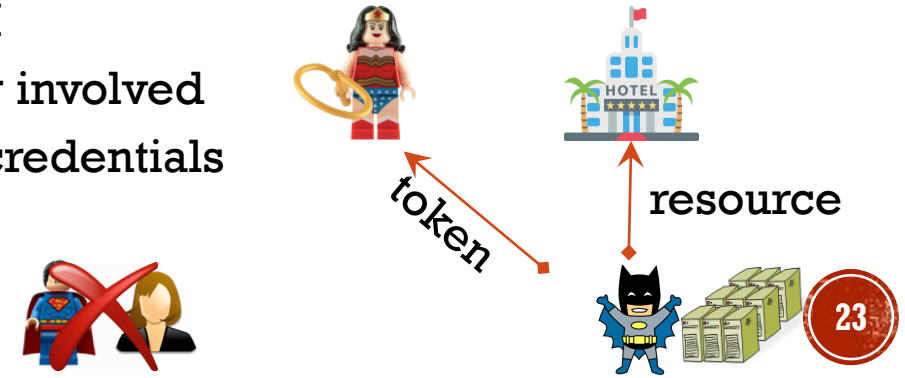
- Implicit Grant
  - App on User Agent needs access to user's resources



- Resource Owner Password Credential Grant
  - App can be trusted w/ user credentials, but does not store them



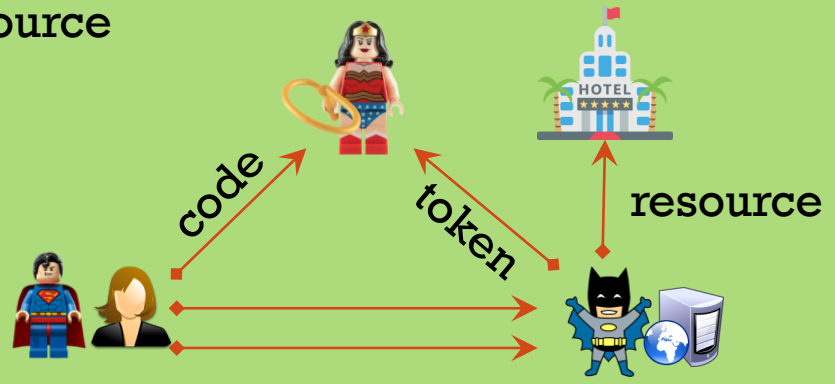
- Client Credentials Grant
  - API-API
  - No user involved
  - Client credentials



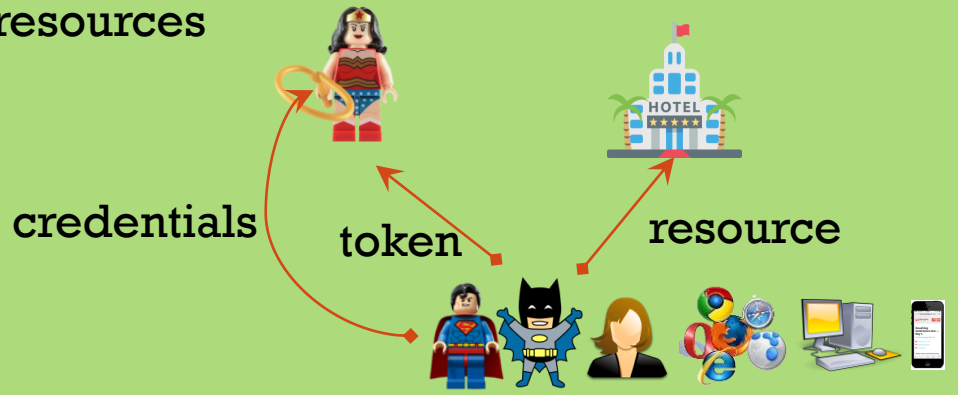
OAUTH FLOWS

BREAK

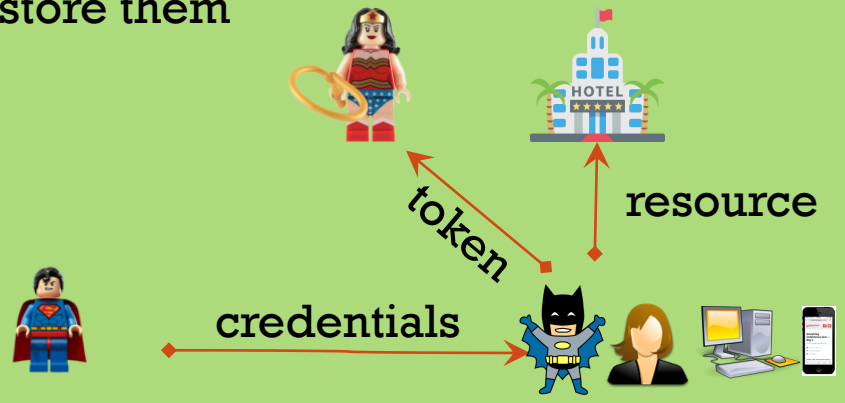
- Authorization Code
  - App on Server side needs access to user's resource



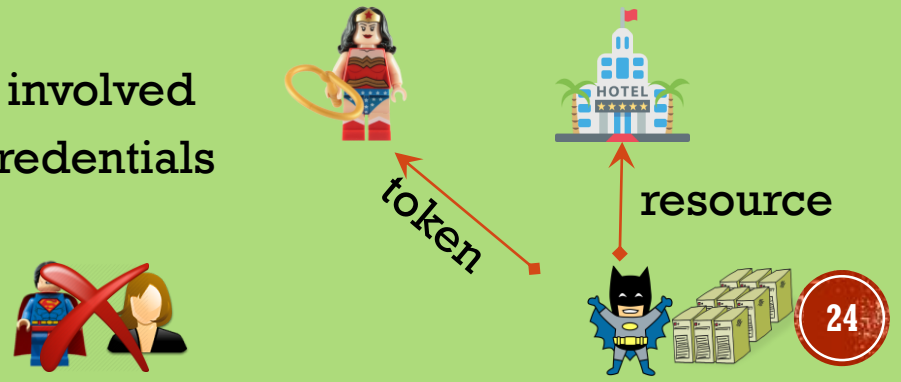
- Implicit Grant
  - App on User Agent needs access to user's resources



- Resource Owner Password Credential Grant
  - App can be trusted w/ user credentials, but does not store them



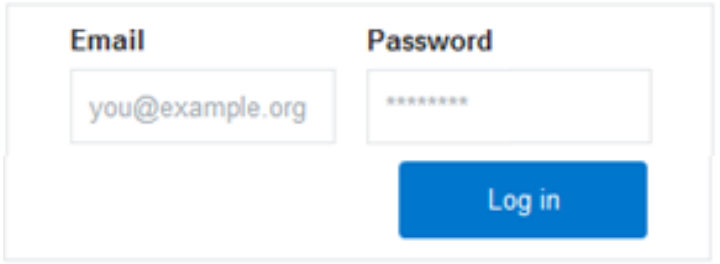
- Client Credentials
  - API-API
  - No user involved
  - Client credentials



PLEASE NO MORE NEW ACCOUNTS …

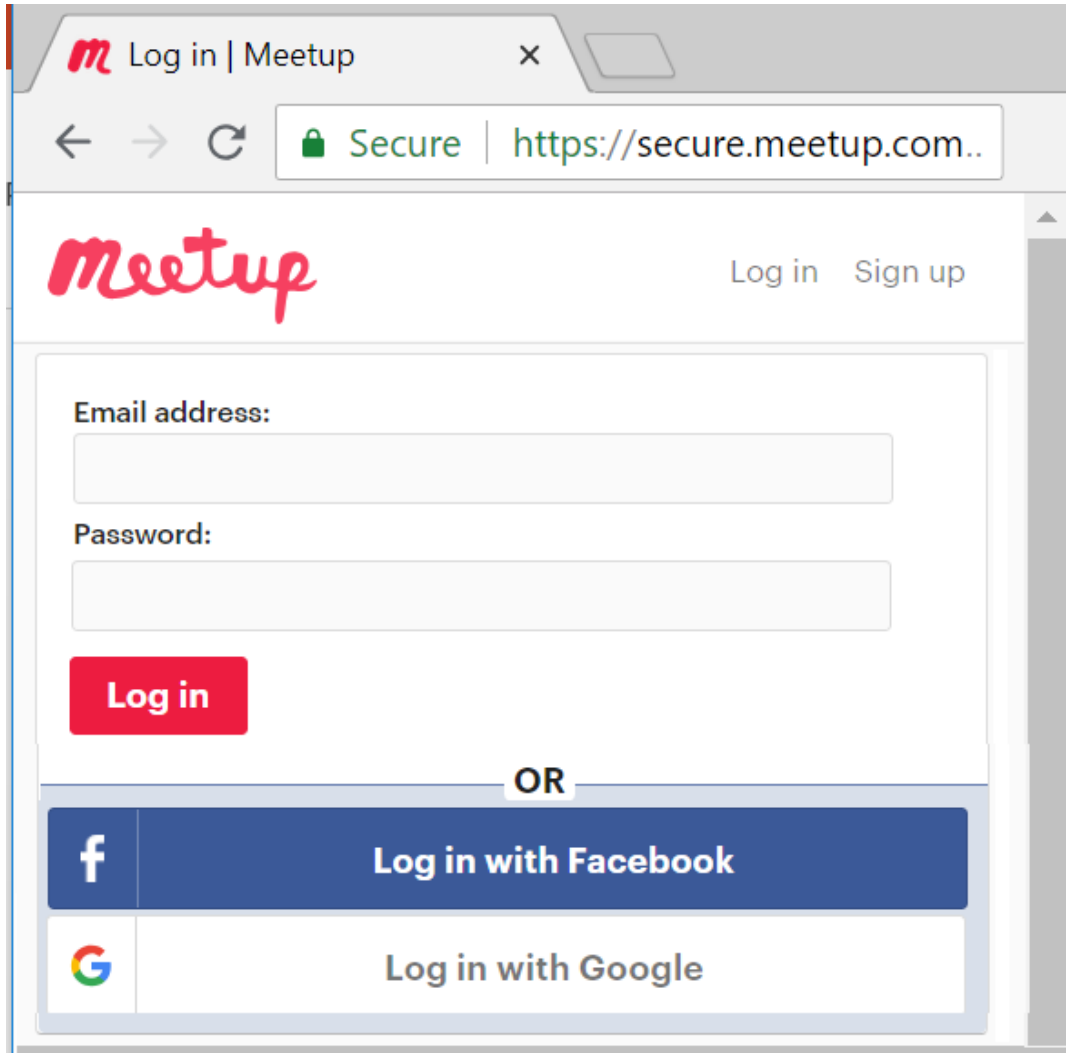
Is it secure when user has to remember so may username/password?

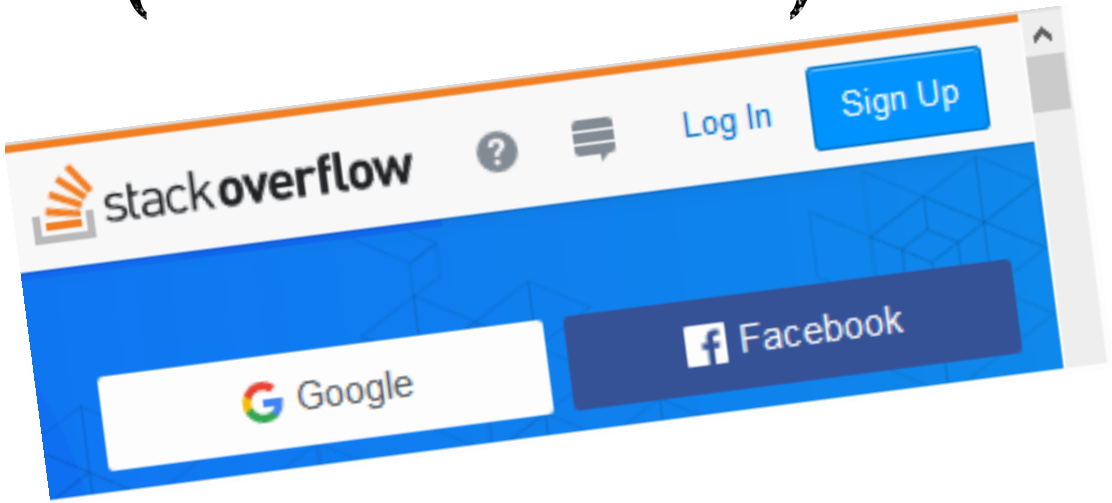
- **Example: Password Rules**
  - Minimum 8 characters long
  - At least three of following
    - upper case and lower case letter
    - Digit 0-9 and Special character @ # $ % ^
    - Change once every 30 days

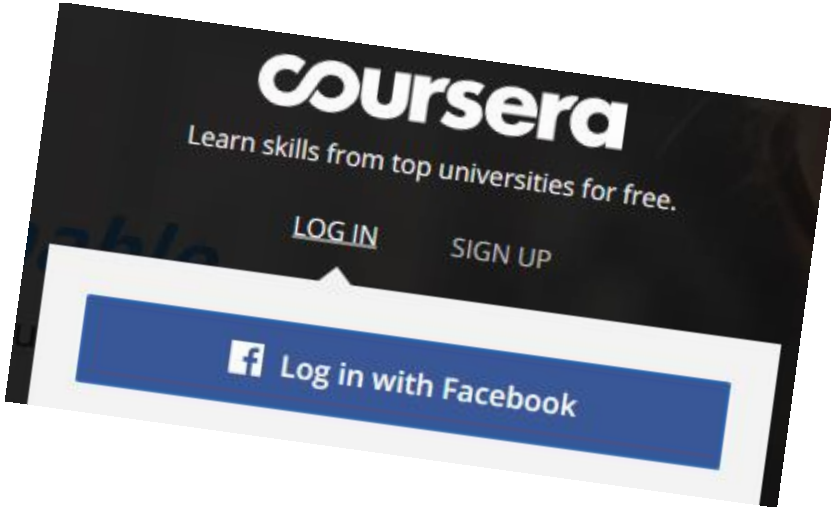


- Multiply this **pain to manage** different **passwords with all the accounts** that you have

HEARD OF SOCIAL LOGIN (THIRD-PARTY)?



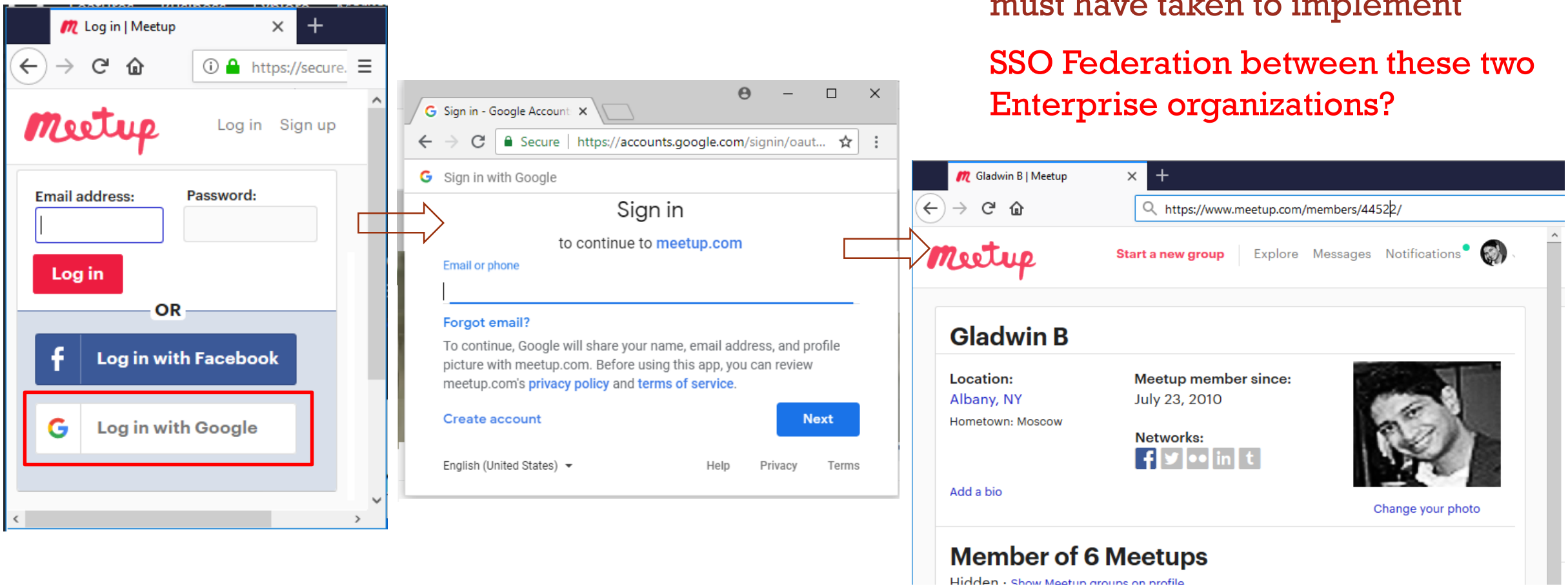




LOG IN WITH GOOGLE

How much effort do you think it must have taken to implement

SSO Federation between these two Enterprise organizations?



OAUTH FOR AUTHENTICATION

Can we somehow use this for AuthN?

Since we were got valid Access Token to access User Superman's resource

This user must be a Superman

User

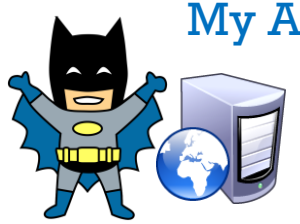
Resource Owner

User Agent Browser

1 – Access my app > | Need Authz Code/Token <

3. Authz Code/Token > | Ok w/ Charge on Account <

My App



Client

ERAN HAMMER

- Eran Hammer the main editor of OAuth 2.0 specification
  - After 3 years left the specification committee
  - RealtimeConf - OAuth 2.0 - Looking Back and Moving On
  - Blog - OAuth 2.0 and the Road to Hell

- Some points about security concerns about OAuth 2.0
  - Some have been addressed by Open ID Connect
  - Too many responsibilities on Client

✓ Specs are open (may)
✓ Bearer token
✓ Standard claims

- OpenID Connect Basic Client Implementer's Guide 1.0
  - Relying Parties using the OAuth Authorization Code Flow

CLIENT SECURITY CONSIDERATIONS

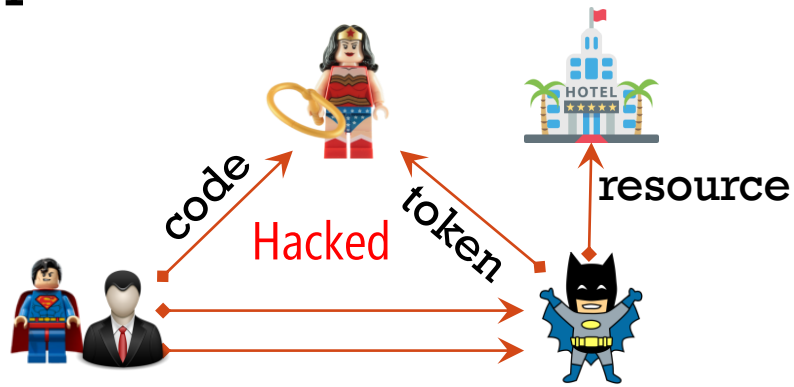
- Man in the middle attack
  - Use TLS (`https`) to secure communication channel
- Verify standard claims
  - `iss, aud, exp, iat,`
- Verify JWT signature and algorithm
- Sensitive information exposure (`query parameters`)
  - Do not pass client secret, id token or access token as query parameter

CLIENT SECURITY CONSIDERATIONS

- CSRF Attack
  - Use `state` parameter with high entropy
- Open Redirectors
  - `https://my-legit-site.com/login?targetUrl=~~/home~~http://evil-site.com`
- Injection attacks
  - Validate incoming data values and context encoding before rendering
- OAuth is only for authorization,
  - For authentication use OpenID Connect

OAUTH MISUSED – NOT DESIGNED FOR AUTHN

https://oauth.net/articles/authentication/



OAuth Misused

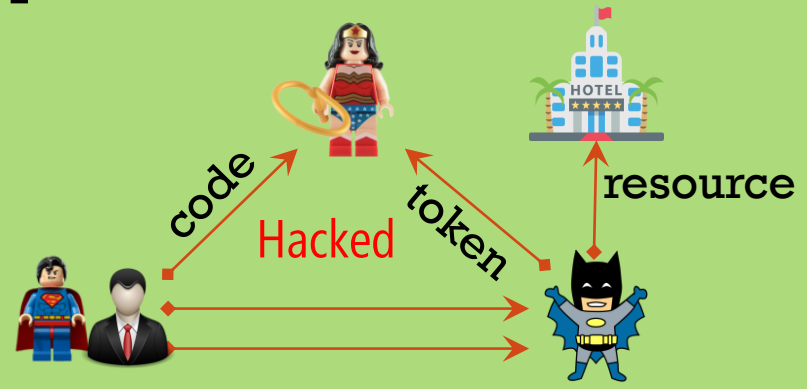
- No explicit AuthN or User ID
- Where to find User Info?
  - Not standardized
- Who is audience of token?
  - Client can impersonate as user

Open ID Connect

- New scope "openid"
- id_token
  - JSON Web Token (JWT)
  - Has special claim called "sub"
  - "aud" must match client_id
  - …
- User Info Endpoint w/ standard claims

OAUTH MISUSED – NOT DESIGNED FOR AUTHN

https://oauth.net/articles/authentication/



OAuth Misused

- No explicit AuthN or User ID
- Where to find User Info?
  - Not standardized
- Who is audience of token?
  - Client can impersonate as user

Open ID Connect

- New scope "openid"
- id_token
  - JSON Web Token (JWT)
  - Has special claim called "sub"
  - "aud" must match client_id
  - …
- User Info Endpoint w/ standard claims

BIRTH OF OPEN ID CONNECT

- OpenID Connect 1.0 (OIDC) is authentication protocol layer on top of OAuth 2.0
  - Based of previously known OpenID specs
  - Uses Authorization Code and Implicit OAuth flows but standardizes certain aspects for authentication purpose

- OAuth is open standard authorization framework for access delegation

SPRING BOOT DEMO

https://github.com/gburboz/gb-oauth2-springboot-talk

- Step-01-InitialSpringSecurityAppp
- Step-02-OAuthSimpleSocialLogin
- Step-03-OpenIDConnectLogin

https://en.wikipedia.org/wiki/JSON_Web_Token

https://jwt.io/

JSON WEB TOKEN [JWT]

- **Base 64 encoded strings:** <header>.<payload>.<signature>


```
{"alg": "HS256", "typ": "JWT"} .
{"sub": "1234567890", "iat": 1516239022,
  "name": "John Doe", "email": "john.doe@jwt.io",
  "athorities" : ["admin", "test"]} .
fmJX_Nk-gvrE-WU1Czs_Et-kVAeWATR1VorpjUCOg8E
```


```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiZW1haWwiOiJqb2huLmRvZUBqd3QuaW8iLCJpYXQiOjE1MTYyMzkwMjIsImF0aG9yaXRpZXMiOlsiYWRtaW4iXX0.fmJX_Nk-gvrE-WU1Czs_Et-kVAeWATR1VorpjUCOg8E
```

JSON WEB TOKEN [JWT]

- Open industry standard RFC 7519 for representing claims securely
- Spring Security 5 uses connect2id Nimbus JOSE + JWT library
- Access token can be JWT but are not required to be
- Open ID Connect ID token is required to be JWT

- ✓ Never let the JWT header alone drive verification
- ✓ Know the algorithms
- ✓ Use an appropriate key size

OPENID CONNECT – ENDPOINTS

ID Token is Signed JWT

Google Accounts
Identity Provider (IdP)

Google
Authorization Server
(OpenID Connect Provider)

Google Profile
Resource Server

Authentication

AuthorizationEndpoint

TokenEndpoint

UserInfoEndpoint

Input user Credentials

Authenticates and establishes User Identity

Input Client ID, Client Redirect URL, scope, and AuthN User

Provides **Authorization Code** for Client to access token/claims

Input Client ID/Secret, Authorization Code

Provides **ID Token** w/ subject info and **Access Token** that can be used to access resources

Input Access Token

Access to **Resource** / Standard Claims

Superman
User Agent

Access Client App

Sends Authorization Code

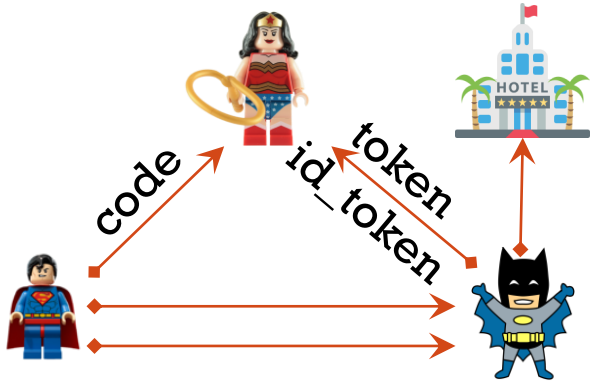
Needs to know User

Authenticated Access

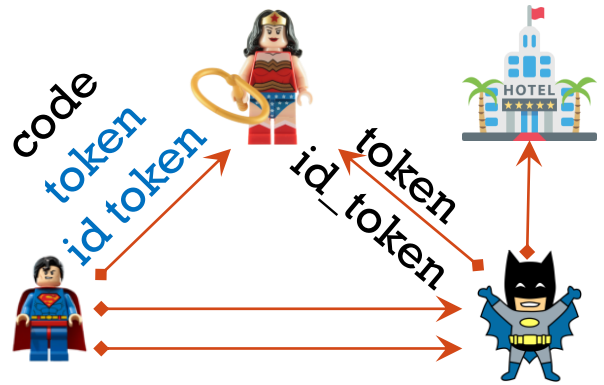
Stack overflow
Client / Relying Party
Client Secret

OPENID CONNECT FLOWS

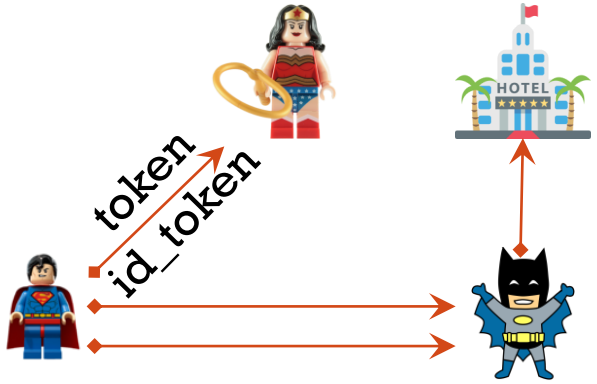
Authorization Code Flow



Hybrid Flow *(can refresh tokens)*



Implicit Flow



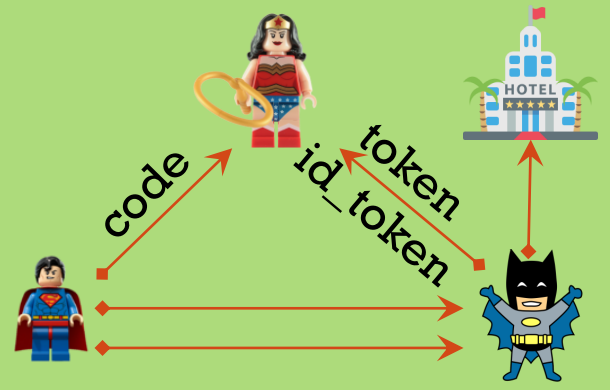
OIDC Grant

- code
  - Intermediate code returned by authz endpoint that can be used to request tokens
- id_token
  - Identity Token that contains user identity
- token
  - Token that can be used to access resources

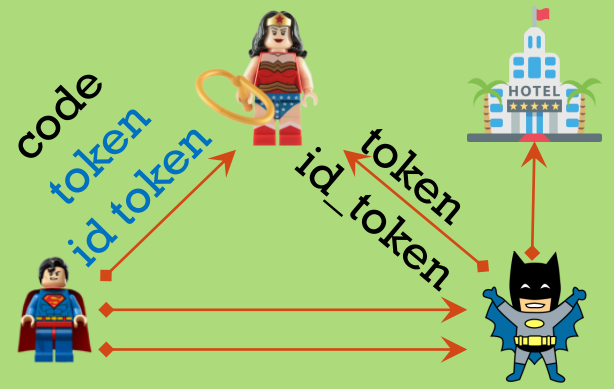
BREAK

OPENID CONNECT FLOWS

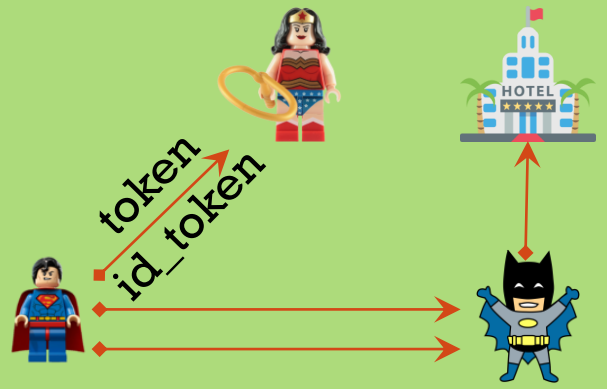
Authorization Code Flow



Hybrid Flow *(can refresh tokens)*



Implicit Flow



OIDC Grant

- code
  - Intermediate code returned by authz endpoint that can be used to request tokens
- id_token
  - Identity Token that contains user identity
- token
  - Token that can be used to access resources

https://openid.net/specs/openid-connect-discovery-1_0.html
https://accounts.google.com/.well-known/openid-configuration

OPENID CONNECT DISCOVERY

Returns provider info in JSON Format


```
{
  issuer: https://accounts.google.com,
  authorization_endpoint: https://accounts.google.com/o/oauth2/v2/auth,
  token_endpoint: https://www.googleapis.com/oauth2/v4/token,
  userinfo_endpoint: https://www.googleapis.com/oauth2/v3/userinfo,
  revocation_endpoint: https://accounts.google.com/o/oauth2/revoke,
  jwks_uri: https://www.googleapis.com/oauth2/v3/certs,
  response_types_supported: [
      "code",
      "token",
      "id_token",
      "code token",
      "code id_token",
      "token id_token",
      "code token id_token",
      "none"
  ],
  subject_types_supported: [
      "public"
  ],
  id_token_signing_alg_values_supported: [
      "RS256"
  ],
  scopes_supported: [
      "openid",
      "email",
      "profile"
  ],
  token_endpoint_auth_methods_supported: [
      "client_secret_post",
      "client_secret_basic"
  ],
  claims_supported: [
      "aud",
      "email",
      "email_verified",
      "exp",
      "family_name",
      "given_name",
      "iat",
      "iss",
      "locale",
      "name",
      "picture",
      "sub"
  ],
  code_challenge_methods_supported: [
      "plain",
      "S256"
  ]
}
```

SPRING BOOT DEMO

https://github.com/gburboz/gb-oauth2-springboot-talk

- Step-01-InitialSpringSecurityAppp
- Step-02-OAuthSimpleSocialLogin
- Step-03-OpenIDConnectLogin
- Step-04-CustomProviderLogin

THANKS EVERYONE.

OAuth/OIDC
Under the hood of social login

Gladwin Burboz