TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  
**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**



**Báo cáo bài tập lớn**

**Môn học: Nhập môn an toàn thông tin**

**Đề tài: Đăng nhập một lần với các giao thức OpenID, Oauth**

Giảng viên hướng dẫn: PGS.TS. Nguyễn Linh Giang

Nhóm sinh viên thực hiện: Nhóm 11

Thành viên trong nhóm: Ngô Việt Cường 20183698

Lê Hữu Tiến Dũng 20183719

Hoàng Đặng Tuấn Đạt 20183701

Nguyễn Minh Đức 20183713

***Hà Nội, 6/2021***

Mục lục

[CHƯƠNG 1: TÌM HIỂU VỀ ĐĂNG NHẬP MỘT LẦN 4](#_Toc73467124)

[**1. Giới thiệu về đăng nhập một lần 4**](#_Toc73467125)

[**2. Cách hoạt động 4**](#_Toc73467126)

[**3. Các giao thức trong đăng nhập một lần 6**](#_Toc73467127)

[CHƯƠNG 2: TÌM HIỂU VỀ OAUTH2 7](#_Toc73467128)

[**1. Tổng quan về Oauth2 7**](#_Toc73467129)

[**1.1. Khái niệm 7**](#_Toc73467130)

[**1.2. Các vai trò trong Oauth2 7**](#_Toc73467131)

[**1.2.1. Resource Owner (người dùng) 7**](#_Toc73467132)

[**1.2.2. Resource Server (the API) 7**](#_Toc73467133)

[**1.2.3. Client (ứng dụng bên thứ 3) 8**](#_Toc73467134)

[**1.2.4. Authorization Server 8**](#_Toc73467135)

[**1.3. Các thuật ngữ ủy quyền cơ bản 8**](#_Toc73467136)

[**1.3.1. Access Token 8**](#_Toc73467137)

[**1.3.2. Refresh Token 9**](#_Toc73467138)

[**1.3.3. Authorization code 9**](#_Toc73467139)

[**2. Nguyên lý hoạt động của Oauth2 9**](#_Toc73467140)

[**2.1. Sơ đồ luồng hoạt động của Oauth2 9**](#_Toc73467141)

[**2.2. Oauth2 3-legged và Oauth 2-legged 10**](#_Toc73467142)

[**2.2.1. Oauth 3-legged 10**](#_Toc73467143)

[**2.2.2. Oauth 2-legged 11**](#_Toc73467144)

[**3. Các loại cấp phép (Authorization Grant) 12**](#_Toc73467145)

[**3.1. Authorization Code 12**](#_Toc73467146)

[**3.2. Implicit 14**](#_Toc73467147)

[**3.3. Resource Owner Password Credentials 15**](#_Toc73467148)

[**3.4. Client Credentials 16**](#_Toc73467149)

[**4. Tính năng bảo mật của Oauth2 17**](#_Toc73467150)

[**4.1. Tokens 17**](#_Toc73467151)

[**4.2. Scope 18**](#_Toc73467152)

[**4.3. Tham số state 18**](#_Toc73467153)

[**4.4. Redirect URI 18**](#_Toc73467154)

[**4.5. Client ID 19**](#_Toc73467155)

[**5. Ưu và nhược điểm của Oauth2 19**](#_Toc73467156)

[**5.1. Ưu điểm 19**](#_Toc73467157)

[**5.2. Nhược điểm 22**](#_Toc73467158)

[CHƯƠNG 3: Tìm hiểu về OPENID 23](#_Toc73467159)

[**1. Tổng quan về OPENID 23**](#_Toc73467160)

[**1.1. Khái niệm 23**](#_Toc73467161)

[**1.2. Các thành phần của một hệ thống OpenID 24**](#_Toc73467162)

[**2. Phương thức hoạt động của OPENID 24**](#_Toc73467163)

[**2.1. Quy trình giao tiếp giữa các thành phần 24**](#_Toc73467164)

[**2.2. Chế độ Smart 25**](#_Toc73467165)

[**2.2.1. Quy trình xác định thành phần Identity Provider 25**](#_Toc73467166)

[**2.2.2. Quy trình gửi thuộc tính định danh 27**](#_Toc73467167)

[**2.2.3. Quy trình kiểm tra thuộc tính định danh 28**](#_Toc73467168)

[**2.3. Chế độ Dump 29**](#_Toc73467169)

[**2.4. Cơ chế xác thực của OpenID 29**](#_Toc73467170)

[**3. Những vấn đề phát sinh với OPENID 32**](#_Toc73467171)

[**3.1. Tính nặc danh 32**](#_Toc73467172)

[**3.2. Độ an toàn thông tin 32**](#_Toc73467173)

[CHƯƠNG 4: CHƯƠNG TRÌNH DEMO 34](#_Toc73467174)

[DANH MỤC TÀI LIỆU THAM KHẢO 36](#_Toc73467175)

# CHƯƠNG 1: TÌM HIỂU VỀ ĐĂNG NHẬP MỘT LẦN

## 1. Giới thiệu về đăng nhập một lần

Đăng nhập một lần (SSO) cho phép người dùng đăng nhập vào một ứng dụng và sau đó tự động đăng nhập vào các ứng dụng khác, bất kể nền tảng, công nghệ hoặc miền mà người dùng đang sử dụng. Người dùng chỉ đăng nhập một lần, do đó tên của tính năng (Đăng nhập một lần).

Ví dụ: nếu người dùng đăng nhập vào một dịch vụ của Google, chẳng hạn như Gmail, người dùng sẽ tự động được xác thực với YouTube, AdSense, Google Analytics và các ứng dụng khác của Google. Tương tự như vậy, nếu người dùng đăng xuất khỏi Gmail hoặc các ứng dụng khác của Google, họ sẽ tự động bị đăng xuất khỏi tất cả các ứng dụng; điều này được gọi là Đăng xuất một lần.

SSO cung cấp trải nghiệm liền mạch cho người dùng khi sử dụng các ứng dụng và dịch vụ của bạn. Thay vì phải nhớ các bộ thông tin xác thực riêng biệt cho từng ứng dụng hoặc dịch vụ, người dùng có thể chỉ cần đăng nhập một lần và truy cập bộ ứng dụng đầy đủ của bạn

Bất cứ khi nào người dùng truy cập vào miền yêu cầu xác thực, họ sẽ được chuyển hướng đến miền xác thực nơi họ có thể được yêu cầu đăng nhập. Nếu người dùng đã đăng nhập tại miền xác thực, họ có thể được chuyển hướng ngay lập tức đến miền ban đầu mà không cần đăng nhập lần nữa.

## 2. Cách hoạt động

SSO hoạt động dựa trên mối quan hệ tin cậy được thiết lập giữa một ứng dụng, được gọi là nhà cung cấp dịch vụ và nhà cung cấp danh tính, như OneLogin. Mối quan hệ tin cậy này thường dựa trên một chứng chỉ được trao đổi giữa nhà cung cấp danh tính và nhà cung cấp dịch vụ. Chứng chỉ này có thể được sử dụng để ký thông tin nhận dạng đang được gửi từ nhà cung cấp danh tính đến nhà cung cấp dịch vụ để nhà cung cấp dịch vụ biết rằng nó đến từ một nguồn đáng tin cậy. Trong SSO, dữ liệu nhận dạng này có dạng mã thông báo chứa các bit thông tin nhận dạng về người dùng như địa chỉ email của người dùng hoặc tên người dùng.

Một user khi đăng nhập vào hệ thống A thì domain của A sẽ lưu thông tin định danh vào cookie, để user này cũng là đã đăng nhập khi truy cập vào hệ thống B thì domain B sẽ phải đọc được cookie của A tạo ra, nhưng điều này là không thể. Với các trình duyệt hiện nay, domain chỉ có thể truy cập cookie do chính nó tạo ra.

Diagram

Description automatically generated

Vì vậy, single sign-on sẽ phải chia sẻ thông tin cookie giữa các domain với nhau. Mỗi giao thức single sign-on sẽ có cơ chế chia sẻ khác nhau, nhưng điểm chung đều là tạo ra một domain trung tâm (central domain). Qua domain này, thông tin về cookie sẽ được chia sẻ đến các domain con. Ví dụ, central domain có thể tạo ra một json web token (jwt) và mã hóa nó. Khi người dùng truy cập vào domain con thì sẽ được điều hướng đến domain trung tâm này, và token sẽ được trả lại và lưu ở phía trình duyệt. Sau đó, nếu người dùng tiếp tục truy cập vào domain con khác thì tương tự, cũng sẽ được điều hướng đến domain trung tâm, nhưng do lần này đã có token nên sẽ được định danh và việc đăng nhập lại là không cần thiết nữa.

Diagram

Description automatically generated

## 3. Các giao thức trong đăng nhập một lần

Các giao thức trong đăng nhập một lần:

* Federated Identity Management (FIM)
* OAuth (specifically OAuth 2.0 nowadays)
* OpenID Connect (OIDC)
* Security Access Markup Language (SAML)
* Same Sign On (SSO)

Trong bài nghiên cứu này chúng em sẽ đi sâu tìm hiểu 2 giao thức Oauth và OpenID trong đăng nhập một lần.

# CHƯƠNG 2: TÌM HIỂU VỀ OAUTH2

## 1. Tổng quan về Oauth2

### 1.1. Khái niệm

OAuth 2 là một framework cho phép các ứng dụng có quyền truy cập hạn chế vào tài khoản người dùng trên dịch vụ HTTP, chẳng hạn như Facebook, GitHub và DigitalOcean. Nó hoạt động bằng cách ủy quyền xác thực người dùng cho dịch vụ lưu trữ tài khoản người dùng và ủy quyền cho các ứng dụng của bên thứ ba truy cập vào tài khoản người dùng. OAuth2 cung cấp các luồng ủy quyền cho các ứng dụng web và máy tính để bàn cũng như thiết bị di động.

### 1.2. Các vai trò trong Oauth2

Oauth định nghĩa 4 vai trò:

* Resource Owner: là thực thể có quyền truy cập vào tài nguyên, thường là enduser
* Resource Server: Máy chủ lưu trữ các tài nguyên, đây chính là API người dùng muốn truy cập
* Client: Đây là app bên thứ 3 muốn truy cập vào tài nguyên được bảo vệ thay mặt cho Resource Owner
* Authorization Server: Máy chủ xác thực Resource Owner và phát hành access token sau khi nhận được sự ủy quyền

#### 1.2.1. Resource Owner (người dùng)

OAuth 2.0 đề cập đến người dùng là "Resource Owner" . Resource Owner là người đang cấp quyền truy cập vào một số phần trong tài khoản của họ. Tài nguyên trong trường hợp này có thể là dữ liệu (ảnh, tài liệu, danh bạ), dịch vụ (đăng bài viết trên blog, chuyển tiền) hoặc bất kỳ tài nguyên nào khác yêu cầu hạn chế truy cập. Bất kỳ hệ thống nào muốn thay mặt người dùng hoạt động trước tiên phải được họ cho phép.

#### 1.2.2. Resource Server (the API)

Oauth 2.0 đề cập đến thuật ngữ "Resource Server" chỉ "máy chủ tài nguyên". Máy chủ tài nguyên là máy chủ chứa thông tin của người dùng đang được ứng dụng của bên thứ ba truy cập. Máy chủ tài nguyên phải có thể chấp nhận và xác thực mã thông báo truy cập và cấp yêu cầu nếu người dùng đã cho phép. Máy chủ tài nguyên không nhất thiết phải biết về các ứng dụng.

#### 1.2.3. Client (ứng dụng bên thứ 3)

Client là ứng dụng đang cố gắng thay mặt người dùng hoặc truy cập tài nguyên của người dùng. Trước khi client có thể truy cập vào tài khoản của người dùng, nó cần phải xin phép. Client sẽ nhận được quyền bằng cách hướng người dùng đến máy chủ ủy quyền hoặc bằng cách xác nhận quyền trực tiếp với máy chủ ủy quyền mà không cần sự tương tác của người dùng.

#### 1.2.4. Authorization Server

Máy chủ ủy quyền là thứ mà người dùng tương tác khi một ứng dụng yêu cầu quyền truy cập vào tài khoản của họ. Đây là máy chủ hiển thị lời nhắc OAuth và là nơi người dùng chấp thuận hoặc từ chối yêu cầu truy cập. Máy chủ ủy quyền cũng chịu trách nhiệm cấp mã thông báo truy cập sau khi người dùng cho phép ứng dụng. Như vậy, máy chủ ủy quyền thường sẽ có hai URL chính, một cho yêu cầu ủy quyền và một cho các ứng dụng sử dụng để cấp mã thông báo truy cập.

### 1.3. Các thuật ngữ ủy quyền cơ bản

#### 1.3.1. Access Token

Access Token hay mã thông báo truy cập là chuỗi được sử dụng khi thực hiện các yêu cầu đã xác thực tới API. Bản thân chuỗi không có ý nghĩa gì đối với ứng dụng sử dụng nó, nhưng thể hiện rằng người dùng đã ủy quyền cho ứng dụng của bên thứ ba truy cập vào tài khoản của họ. Mã thông báo có thời hạn truy cập tương ứng, phạm vi và thông tin tiềm năng khác mà máy chủ cần.

Access Token format bởi mặc định access token được sinh ra cho việc ủy quyền ở định dạng JWT. JWTs gồm có 3 thành phần: header, playload và signature

* + **Header** thì chứa metadata về loại thông báo và thuật toán mã hóa để bảo mật nội dung của nó.
  + **Payload** chứa một tập yêu cầu, yêu cầu về quyền nên được cho phép và các thông tin khác như đối tượng dự định và thời gian hết hạn.
  + **Signture** được sử dụng để xác thực mã thông báo đáng tin cậy hay không bị làm giả.

#### 1.3.2. Refresh Token

Refresh Token hay mã thông báo làm mới là một chuỗi được sử dụng để lấy mã thông báo truy cập mới khi mã thông báo truy cập hết hạn. Không phải tất cả các API đều sử dụng mã thông báo làm mới.

#### 1.3.3. Authorization code

Authorization code hay mã ủy quyền là mã thông báo trung gian được sử dụng trong luồng ứng dụng phía máy chủ. Mã ủy quyền được trả lại cho khách hàng sau bước ủy quyền và sau đó khách hàng sẽ trao đổi mã đó để lấy mã thông báo truy cập.

## 2. Nguyên lý hoạt động của Oauth2

### 2.1. Sơ đồ luồng hoạt động của Oauth2

Bây giờ chúng ta sẽ có cái nhìn chi tiết về cách thức hoạt động của Oauth2:

Diagram

Description automatically generated

* Ứng dụng (Client) yêu cầu ủy quyền(Authorization Request) từ chủ sở hữu tài nguyên để truy cập tài nguyên
* Với điều kiện chủ tài nguyên (Resource Owner) cho phép quyền truy cập này, ứng dụng sẽ nhận được sự cho phép (Authorization Grant)
* Ứng dụng yêu cầu mã thông báo truy cập từ máy chủ ủy quyền (API) bằng cách trình bày xác thực danh tính của chính nó và cấp ủy quyền (Authorization Grant)
* Nếu danh tính ứng dụng được xác thực và việc cấp phép hợp lệ, máy chủ ủy quyền (API) sẽ cấp mã thông báo truy cập cho ứng dụng (Access Token). Việc ủy ​​quyền đã hoàn tất
* Ứng dụng yêu cầu tài nguyên từ máy chủ tài nguyên (API) và xuất trình mã thông báo truy cập(Access Token) để xác thực
* Nếu mã thông báo truy cập hợp lệ, máy chủ tài nguyên (API) sẽ cung cấp tài nguyên (Resources) cho ứng dụng

Luồng thực tế của quá trình này sẽ khác nhau tùy thuộc vào loại cấp phép đang sử dụng, nhưng trên đây là ý tưởng chung.

**2.2. Oauth2 3-legged và Oauth 2-legged**

Trước khi đi vào các loại cấp phép cụ thể thì chúng ta phải làm quen với một khái niệm gọi là flow chứng thực Client Credentials . Flow này có thể hiểu đơn giản là bước 1 và 2 ở trên, tức là: làm thế nào để người dùng nói cho facebook được là họ trao quyền cho nhà quản lý ứng dụng access vào profile của họ?

Mục đích chính của flow này là để cho bên cung cấp dịch vụ (facebook) biết được là người dùng "đã" cung cấp cho ứng dụng quyền được truy cập thông tin của họ. Thông tin này có thể được cung cấp tại lúc mà ứng dụng yêu cầu người dùng, hoặc có thể được cung cấp "trước" tại một thời điểm nào đó. Chính sự khác biệt này tạo ra 2 cách làm khác nhau là **3-legged** và **2-legged** oauth.

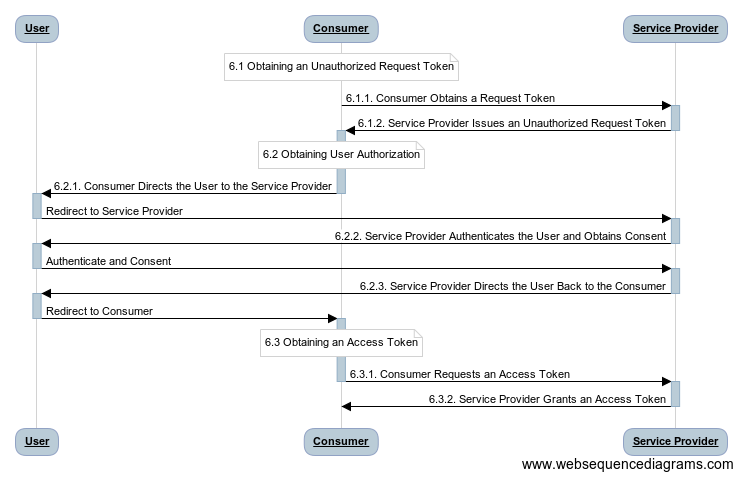
#### 2.2.1. Oauth 3-legged

Đây chính là flow phổ thông cho mô hình oauth2 hiện nay. Client Credentials Flow sẽ được thực hiện thông qua việc redirect người dùng đến website của bên cung cấp dịch vụ, người dùng sẽ lựa chọn có hoặc không đồng ý (họ có thể gợi nhớ lại khi họ đăng nhập vào facebook app lần đầu tiên, facebook sẽ cho họ một form để chọn có/không).

Sau đó dựa vào redirect\_uri mà ứng dụng cung cấp trước đó cho facebook, facebook sẽ redirect-back lại dịch vụ của nó, kèm theo một token đã được chứng thực.

Sau cùng ứng dụng sẽ sử dụng token đó như "chìa khoá" để truy cập vào profile của người dùng.

Flow chứng thực này yêu cầu cả 3 bên: người dùng, Facebook và nhà lập trình ứng dụng, do đó mà nó được gọi là 3-legged.



#### 2.2.2. Oauth 2-legged

Vậy nếu ứng dụng đó được facebook tin tưởng "hoàn toàn" thì sao. Ví dụ instagram chẳng hạn. Khi đó người dùng phải accept cho ứng dụng truy cập tài nguyên của họ trên facebook là không cần thiết. Hoặc một ví dụ khác là khi làm một widget của google app và người dùng cho phép ứng dụng quyền sử dụng gmail account của họ.

Trong những trường hợp trên thì flow redirect đi redirect lại như ở trên là không cần thiết, khi đó thì chỉ cần ứng dụng nói chuyện trực tiếp với bên cung cấp dịch vụ là họ cần thông tin của người dùng thôi là đủ. Việc cấp quyền của người dùng cho bên ứng dụng có thể được thực hiện tại một thời điểm bất kỳ như lúc bạn cài đặt app của tôi vào desktop chẳng hạn (pre-installed app)

Mô hình này không cần phía người dùng , chỉ cần phía ứng dụng và facebook, do đó mà nó được gọi là 2-legged.

Graphical user interface, diagram, text, application

Description automatically generated

## 3. Các loại cấp phép (Authorization Grant)

Trong luồng giao thức trừu tượng ở trên, bốn bước đầu tiên bao gồm việc nhận được cấp phép ủy quyền và mã thông báo truy cập. Loại cấp phép phụ thuộc vào phương thức được ứng dụng sử dụng để yêu cầu ủy quyền và các loại cấp phép được API hỗ trợ. OAuth2 xác định bốn loại cấp phép, mỗi loại đều hữu ích trong các trường hợp khác nhau:

* **Authorization Code**: Sử dụng với các ứng dụng phía máy chủ (server)
* **Implicit**: Sử dụng với ứng dụng di động hoặc ứng dụng web (ứng dụng chạy trên thiết bị của người dùng)
* **Resource Owner Password Credentials:** Được sử dụng với các ứng dụng đáng tin cậy, chẳng hạn như các ứng dụng do chính dịch vụ sở hữu
* **Client Credentials:** Được sử dụng với quyền truy cập API ứng dụng

### 3.1. Authorization Code

Đây là loại cấp mã ủy quyền được sử dụng phổ biến nhất vì nó được tối ưu hóa cho các ứng dụng phía máy chủ, nơi mã nguồn không bị lộ công khai và có thể duy trì tính bảo mật của Client Secret. Đây là luồng dựa trên chuyển hướng, có nghĩa là ứng dụng phải có khả năng tương tác với tác nhân người dùng (tức là trình duyệt web của người dùng) và nhận mã ủy quyền API được định tuyến thông qua tác nhân người dùng.

Luồng cấp mã ủy quyền:

Diagram

Description automatically generated

* **Truy cập ứng dụng**: Người dùng truy cập vào ứng dụng và kích hoạt xác thực và ủy quyền.
* **Xác thực và yêu cầu ủy quyền**: Ứng dụng chuyển hướng người dùng đến máy chủ ủy quyền nơi nó yêu cầu người dùng nhập tên người dùng và mật khẩu của họ. Lần đầu tiên người dùng xem qua quy trình này cho ứng dụng, người dùng sẽ thấy trang phê duyệt. Trên trang này, người dùng có thể chọn quyền để cho phép ứng dụng thay mặt họ truy cập tài nguyên.
* **Xác thực và cấp phép**: Máy chủ ủy quyền nhận xác thực và cấp ủy quyền.
* **Gửi mã ủy quyền:** Sau khi người dùng ủy quyền ứng dụng, máy chủ ủy quyền sẽ gửi mã ủy quyền đến ứng dụng bằng cách sử dụng chuyển hướng.
* **Yêu cầu mã trao đổi cho Token:** Ứng dụng sử dụng mã ủy quyền để yêu cầu mã thông báo truy cập từ máy chủ ủy quyền. Điều này cung cấp cho ứng dụng quyền truy cập vào các quyền đã được phê duyệt
* **Cấp Access Token:** Máy chủ ủy quyền xác thực mã ủy quyền và cấp mã thông báo truy cập (Access Token).
* **Yêu cầu tài nguyên với Access Token:** Ứng dụng cố gắng truy cập tài nguyên từ máy chủ tài nguyên bằng cách xuất trình mã thông báo truy cập.
* **Trả về tài nguyên:** Nếu mã thông báo truy cập hợp lệ, máy chủ tài nguyên sẽ trả về tài nguyên mà người dùng đã ủy quyền cho ứng dụng nhận.

### 3.2. Implicit

Được sử dụng cho các ứng dụng di động và ứng dụng web (tức là các ứng dụng chạy trong trình duyệt web), nơi mà tính bảo mật của Client không được đảm bảo. Đây cũng là một luồng dựa trên chuyển hướng nhưng mã thông báo truy cập được cấp cho tác nhân người dùng để chuyển tiếp đến ứng dụng, do đó, nó có thể được hiển thị cho người dùng và các ứng dụng khác trên thiết bị của người dùng. Ngoài ra, luồng này không xác thực danh tính của ứng dụng và dựa vào URI chuyển hướng (đã được đăng ký với dịch vụ) để phục vụ mục đích này. Loại grant này không hỗ trợ refresh token

Sơ đồ yêu cầu ủy quyền của ứng dụng:

Diagram

Description automatically generated

* **Truy cập ứng dụng**: Người dùng truy cập vào ứng dụng và kích hoạt xác thực và ủy quyền.
* **Xác thực và yêu cầu ủy quyền**: Ứng dụng yêu cầu người dùng nhập tên người dùng và mật khẩu của họ. Lần đầu tiên người dùng xem qua quy trình này cho ứng dụng, người dùng sẽ thấy trang phê duyệt. Trên trang này, người dùng có thể chọn quyền để cho phép ứng dụng thay mặt họ truy cập tài nguyên.
* **Xác thực và cấp phép**: Máy chủ ủy quyền nhận xác thực và cấp ủy quyền.
* **Cấp Access Token**: Máy chủ ủy quyền xác thực mã ủy quyền và trả về mã thông báo truy cập có URL chuyển hướng.
* **Yêu cầu tài nguyên với Access Token trong URL:** Ứng dụng cố gắng truy cập tài nguyên từ máy chủ tài nguyên bằng cách hiển thị mã thông báo truy cập trong URL.
* **Trả về tài nguyên:** Nếu mã thông báo truy cập hợp lệ, máy chủ tài nguyên sẽ trả về tài nguyên mà người dùng đã ủy quyền cho ứng dụng nhận.

### ****3.3. Resource Owner Password Credentials****

Với mô hình ủy quyền **Owner Password Credentials, người dùng cung cấp thông tin đăng nhập dịch vụ của họ (tên người dùng và mật khẩu) trực tiếp cho ứng dụng, ứng dụng này sử dụng thông tin đăng nhập để lấy mã thông báo truy cập từ dịch vụ. Loại tài trợ này chỉ nên được bật trên máy chủ ủy quyền nếu các luồng khác không khả thi. Ngoài ra, nó chỉ nên được sử dụng nếu ứng dụng được người dùng tin cậy.**

Diagram

Description automatically generated

* **Xác thực với tên người dùng và mật khẩu**: Người dùng xác thực với ứng dụng bằng tên người dùng và mật khẩu của họ.
* **Gửi tên người dùng/ mật khẩu:** Ứng dụng gửi tên người dùng và mật khẩu đến máy chủ ủy quyền để xác thực.
* **Cấp Access Token:** Máy chủ ủy quyền xác thực tên người dùng và mật khẩu và cấp mã thông báo truy cập.
* **Yêu cầu tài nguyên với Access Token:** Ứng dụng cố gắng truy cập tài nguyên từ máy chủ tài nguyên bằng cách xuất trình mã thông báo truy cập.
* **Trả về tài nguyên:** Nếu mã thông báo truy cập hợp lệ, máy chủ tài nguyên sẽ trả về tài nguyên mà người dùng đã ủy quyền cho ứng dụng nhận.

### ****3.4. Client Credentials****

**Loại cấp thông tin xác thực khách hàng cung cấp cho ứng dụng một cách để truy cập vào tài khoản dịch vụ của chính nó. Ví dụ về thời điểm điều này có thể hữu ích bao gồm nếu ứng dụng muốn cập nhật mô tả đã đăng ký hoặc chuyển hướng URI hoặc truy cập dữ liệu khác được lưu trữ trong tài khoản dịch vụ của ứng dụng thông qua API.**

**Diagram

Description automatically generated**

* **Xác thực với Client ID và Client Secret:** Ứng dụng xác thực với máy chủ ủy quyền bằng Client ID và Client Secret.
* **Cấp Access Token**: Máy chủ ủy quyền xác thực Client ID, Secret và cấp Access Token.
* **Yêu cầu tài nguyên với Access Token**: Ứng dụng cố gắng truy cập tài nguyên từ máy chủ tài nguyên bằng cách xuất trình mã thông báo truy cập.
* **Trả về tài nguyên**: Nếu mã thông báo truy cập hợp lệ, máy chủ tài nguyên sẽ trả lại tài nguyên cho ứng dụng.

## 4. Tính năng bảo mật của Oauth2

### 4.1. Tokens

OAuth sử dụng rộng rãi nhiều loại token (access token, refresh token, "mã" ủy quyền- authorization "codes"). Nội dung thông tin của một token có thể được trình bày theo hai cách như sau:

Xử lý (Handle) : Một handle là một tham chiếu đến một số cấu trúc dữ liệu nội bộ trong máy chủ ủy quyền (authorization server); cấu trúc dữ liệu nội bộ chứa các thuộc tính của token, chẳng hạn như id người dùng (UID), phạm vi, v.v. Các handle cho phép thu hồi đơn giản và không yêu cầu cơ chế mã hóa để bảo vệ nội dung token khỏi bị sửa đổi. Mặt khác, các handle yêu cầu sự giao tiếp giữa thực thể phát hành và tiêu thụ (ví dụ: máy chủ ủy quyền và máy chủ tài nguyên) để xác thực token và lấy dữ liệu token-bound. Giao tiếp này có thể có tác động tiêu cực đến hiệu suất và khả năng mở rộng nếu cả hai thực thể ở trên các hệ thống khác nhau. Do đó, handle thường được sử dụng nếu thực thể phát hành và tiêu thụ là như nhau. Một 'handle' token thường được gọi là một token mã hóa mờ đục vì máy chủ tài nguyên không cần phải giải nghĩa token một cách trực tiếp; nó chỉ đơn giản là sử dụng token.

Xác nhận (Assertion - còn gọi là mã thông báo độc lập - self-contained token): Một assertion là một token có thể phân tích cú pháp. Một assertion thường có thời lượng, có đối tượng và được chữ ký số để đảm bảo tính toàn vẹn dữ liệu và xác thực nguồn gốc. Nó chứa thông tin về người dùng và khách hàng. Ví dụ về các định dạng assertion là các Security Assertion Markup Language (SAML) [OASIS.saml-core-2.0-os] và thẻ Kerberos [RFC4120]. Các assertion thường có thể được xác nhận trực tiếp và được sử dụng bởi một máy chủ tài nguyên mà không cần tương tác với máy chủ ủy quyền. Điều này dẫn đến hiệu suất và khả năng mở rộng tốt hơn trong các triển khai nơi các thực thể phát hành và tiêu thụ ở trên các hệ thống khác nhau. Việc assertion thực hiện thu hồi token khó khăn hơn so với handle.

Các token có thể được sử dụng theo hai cách để gọi yêu cầu trên các máy chủ tài nguyên, đó là:

* **bearer token**: là token có thể được sử dụng bởi bất kỳ client nào đã nhận được token (ví dụ: [RFC6750]). Vì chỉ sở hữu là đủ để sử dụng token, điều quan trọng là giao tiếp giữa các điểm cuối phải được bảo mật để đảm bảo rằng chỉ các điểm cuối được ủy quyền mới có thể nắm bắt token. bearer token thuận tiện cho các ứng dụng khách, vì nó không yêu cầu họ làm bất cứ điều gì để sử dụng chúng (chẳng hạn như bằng chứng nhận dạng). bearer token có đặc điểm tương tự như cookie đăng nhập một lần (SSO) được sử dụng trong trình duyệt.
* **proof token**: là token mà chỉ một khách hàng cụ thể có thể sử dụng. Mỗi lần sử dụng token yêu cầu khách hàng thực hiện một số hành động chứng minh rằng đó là người dùng được ủy quyền của token. Ví dụ về điều này là các MAC-type access tokens, (ví dụ: [OAuth-HTTP-MAC]).

### 4.2. Scope

Scope thể hiện ủy quyền truy cập được liên kết với một token cụ thể đối với các máy chủ tài nguyên, tài nguyên và phương thức trên các tài nguyên đó. Scope là cách để OAuth quản lý rõ ràng sức mạnh liên quan đến một access token. Một scope có thể được kiểm soát bởi máy chủ ủy quyền và / hoặc người dùng cuối nhằm hạn chế quyền truy cập vào tài nguyên cho các client OAuth mà các bên này cho rằng kém an toàn hoặc đáng tin cậy. Theo tùy chọn, client có thể yêu cầu phạm vi áp dụng cho token nhưng chỉ với scope nhỏ hơn mức được cấp, ví dụ, để giảm tác động nguy hại nếu token này được gửi qua các kênh không an toàn. Một scope thường được bổ sung bởi một hạn chế về thời gian tồn tại của token.

### 4.3. Tham số state

Tham số "state" được sử dụng để liên kết các yêu cầu và lời gọi lại để ngăn chặn các cuộc tấn công giả mạo yêu cầu giữa các trang web trong đó kẻ tấn công cho phép truy cập vào tài nguyên của chính mình và sau đó lừa người dùng tiếp tục một chuyển hướng với token của kẻ tấn công. Tham số này phải liên kết với trạng thái được xác thực trong tác nhân người dùng và, theo thông số OAuth cốt lõi, tác nhân người dùng phải có khả năng giữ nó ở vị trí chỉ có thể truy cập bởi client và tác nhân người dùng, nghĩa là được bảo vệ bởi chính sách cùng nguồn gốc.

### 4.4. Redirect URI

URI chuyển hướng giúp phát hiện các client độc hại và ngăn chặn các cuộc tấn công lừa đảo từ các client đang cố lừa người dùng tin rằng kẻ lừa đảo là client. Giá trị của URI chuyển hướng thực tế được sử dụng trong yêu cầu ủy quyền phải được trình bày và được xác minh khi "mã" ủy quyền được trao đổi với các token. Điều này giúp ngăn chặn các cuộc tấn công trong đó "mã" ủy quyền được tiết lộ thông qua các bộ chuyển hướng và client giả mạo ứng dụng web. Máy chủ ủy quyền phải yêu cầu các public client và confidential client sử dụng loại cấp quyền ngầm để đăng ký trước URI chuyển hướng của họ và xác thực đối với URI chuyển hướng đã đăng ký trong yêu cầu ủy quyền.

### 4.5. Client ID

Các giao thức xác thực thường không tính đến danh tính của thành phần phần mềm thay mặt cho người dùng cuối. OAuth thực hiện điều này để tăng mức độ bảo mật trong các tình huống ủy quyền và vì client sẽ có thể hành động mà không cần sự có mặt của người dùng.

OAuth sử dụng định danh client để đối chiếu các yêu cầu liên quan đến cùng một người khởi tạo, chẳng hạn như:

* một quy trình ủy quyền của người dùng cuối và yêu cầu tương ứng trên điểm cuối của token để trao đổi "mã" ủy quyền cho các token
* ủy quyền ban đầu và phát hành một token của người dùng cuối cho một client cụ thể và các yêu cầu tiếp theo của client này để nhận token mà không cần sự đồng ý của người dùng (tự động xử lý ủy quyền lặp lại)

Mã định danh này cũng có thể được sử dụng bởi máy chủ ủy quyền để hiển thị thông tin đăng ký có liên quan cho người dùng khi yêu cầu đồng ý cho một phạm vi được yêu cầu bởi một client cụ thể. Mã định danh client có thể được sử dụng để giới hạn số lượng yêu cầu cho một client cụ thể hoặc tính phí cho client theo yêu cầu. Ngoài ra, có thể hữu ích để phân biệt quyền truy cập của các client khác nhau.

## 5. Ưu và nhược điểm của Oauth2

### 5.1. Ưu điểm

**Bảo vệ API**

Giả sử, một nhà phát triển cung cấp API cho một số application (có thể là RESTful). Để truy cập vào nó một cách an toàn, có một cách là sử dụng xác thực cơ bản, trong đó username và password được gửi bằng Base64- nhưng không được mã hóa hoặc băm nhỏ và SSL được sử dụng để bảo mật việc truyền dữ liệu. Hạn chế ở đây là, trong loại xác thực này, người dùng phải nhập cả username và password của mình.

Vậy điều này có thể cải thiện bằng cách nào? Để bảo mật với ủy quyền truy cập (Oauth2), mỗi khi yêu cầu được gửi tới API, thay vì username và password, một access token được gửi. Application (client) nhận được token này trước khi thực hiện các yêu cầu, thay mặt người dùng sử dụng API.

Chẳng hạn, một application đang sử dụng API, bắt đầu hoạt động sai và sử dụng nó không đúng cách. Thì tất cả những gì phải làm là thu hồi access token, làm cho tất cả các yêu cầu trong tương lai không hợp lệ (nếu chúng được thực hiện với cùng một token).

Một điều quan trọng khác cần lưu ý là access token có thể đặt theo phạm vi truy cập, được gọi là scope. Vì vậy, một application có thể không có quyền truy cập API đầy đủ khi sử dụng access token. Ngược lại, nếu application đang sử dụng username và password thay vì access token, và các thông tin đăng nhập này đã bị đánh cắp, toàn bộ API có thể bị sử dụng sai cho đến khi ngăn chặn được việc tấn công tài khoản.

**Các ứng dụng doanh nghiệp nội bộ (Internal enterprise applications)**

Một ví dụ khác, một công ty sử dụng nội bộ một bộ ứng dụng doanh nghiệp. Các nhân viên trong công ty phải nhập username và password theo cách thủ công trong mỗi ứng dụng riêng biệt. Điều này đặt ra rủi ro an ninh trong một số lĩnh vực; cần một cơ sở dữ liệu để lưu trữ các password, các password được gửi nhiều lần...

Với ủy quyền truy cập (OAuth 2.0), những gì xảy ra là có một application mà người dùng đã đăng nhập bằng username và password của mình (nhà cung cấp dịch vụ dựa trên OAuth 2.0) và trong tất cả các application khác, người dùng chỉ cần chuyển hướng đến nhà cung cấp- nơi người dùng đã đăng nhập và xác nhận rằng anh ta muốn được ủy quyền. Bằng cách này, thay vì lưu trữ mật khẩu, các application này đang lưu trữ token cho người dùng. Lợi ích của điều này là nếu như mật khẩu bị đánh cắp, người dùng phải đặt lại mật khẩu của mình thì khi token bị đặt cắp, token bị thu hồi (không hợp lệ).

Một ví dụ liên quan khác là bảo mật trong các công ty BYOD(bring your own device). Nếu người dùng mang điện thoại thông minh của mình và sử dụng ứng dụng nội bộ của công ty trên đó. Khi điện thoại thông minh này bị đánh cắp hoặc bị xâm phạm, password của người dùng sẽ không bị lộ và chỉ các token đang sử dụng sẽ bị vô hiệu hóa. Người dùng có thể nhận được một thiết bị mới và khi anh ta ủy quyền cho nó, anh ta sẽ nhận được một token mới.

***Tích hợp dịch vụ và ủy quyền***

Trong trường hợp người dùng có tài khoản trên dịch vụ chia sẻ ảnh, và cả dịch vụ đặt hàng in.

Nếu hai dịch vụ này có tích hợp OAuth2.0 giữa chúng, người dùng có thể ủy quyền cho dịch vụ in để truy cập tài nguyên từ dịch vụ chia sẻ ảnh. Ví dụ: người dùng sẽ đăng nhập vào dịch vụ in và yêu cầu tích hợp, sau đó người dùng được chuyển hướng trở lại dịch vụ chia sẻ ảnh, nơi người dùng chấp thuận yêu cầu ủy quyền.

Từ thời điểm này, dịch vụ in có một access token thay mặt cho người dùng và có thể nhận biết ảnh nào mà người dùng đã tải lên. Và người dùng có thể truy cập ảnh trên dịch vụ in mà không cần tải lại chúng ở đó, nhưng được truy xuất thông qua dịch vụ chia sẻ ảnh.

Ưu điểm ở đây là người dùng không cẩn cung cấp username và password cho dịch vụ in, nhưng thay vào đó, dịch vụ in đã nhận được một access token khi được người dùng ủy quyền.

Điều này cho thấy một trong những thế mạnh chính của việc sử dụng ủy quyền truy cập. Với ủy quyền truy cập (OAuth 2.0), bạn có thể cấp quyền truy cập vào dữ liệu / tài nguyên của mình trên một số dịch vụ cho một dịch vụ khác và dễ dàng thu hồi quyền truy cập này khi bạn đổi ý. Điều này áp dụng ở mọi nơi, từ các ứng dụng doanh nghiệp nội bộ đến các dịch vụ khác nhau như ứng dụng truyền thông xã hội.

***Nhận dạng liên kết***

Một ưu điểm khác của ủy quyền là nhận dạng liên kết. Với nhận dạng liên kết, nhận dạng số và thông tin của một người (như là email, tên, họ và giới tính) có thể được liên kết giữa một số dịch vụ riêng biệt.

Hai ví dụ trước đã cho thấy điều này, nhưng hãy xem một ví dụ khác; Người dùng có thể đăng nhập vào một nhà cung cấp OAuth 2.0 (ví dụ: Facebook hoặc LinkedIn) và sau đó đăng nhập vào các dịch vụ web khác thông qua nhà cung cấp này mà không cần nhập username và password mới. Nếu người dùng muốn để lại nhận xét trên một số bài đăng trên blog và cần được xác định để nhận xét được chấp nhận.

Trường hợp này còn được gọi là sử dụng OAuth 2.0 để xác thực. Application (ví dụ: nền tảng blog) chỉ yêu cầu thông tin của người dùng sau khi được ủy quyền mà không đòi hỏi gì hơn.

***Giám sát dịch vụ dễ dàng hơn***

Cuối cùng nhưng không kém phần quan trọng, doanh nghiệp có thể theo dõi và giám sát dễ dàng hơn access token nào đang thực hiện yêu cầu gì; dựa trên điều này, họ có thể thực hiện các tính toán và hiểu rõ hơn về dịch vụ nào được khách hàng sử dụng thường xuyên hơn và tối ưu hóa.

### 5.2. Nhược điểm

* Trong quá trình xác minh người dùng, đôi khi bạn phải thực hiện các yêu cầu bổ sung để có được thông tin người dùng tối thiểu. Nó có thể được giải quyết với sự trợ giúp của mã thông báo jwt, nhưng không phải tất cả các dịch vụ đều hỗ trợ nó.
* Khi mã thông báo bị đánh cắp, kẻ tấn công sẽ có quyền truy cập vào dữ liệu an toàn trong một thời gian.Ví dụ như tài khoản facebook hay google, nếu mất quyền truy cập vào các tài khoản đó, người dùng sẽ mất quyền truy cập vào tất cả các trang web sử dụng tài khoản đó để đăng nhập. Để giảm thiểu rủi ro này, có thể sử dụng mã thông báo có chữ ký.
* Mặc dù vẫn còn tồn tại lỗ hổng bảo mật khi duyệt web ở chrome,  nhưng người dùng có thể hạn chế bằng cách sử dụng HTTPS thay thì HTTP thông thường.

# CHƯƠNG 3: Tìm hiểu về OPENID

## 1. Tổng quan về OPENID

### 1.1. Khái niệm

OpenID là một tiêu chuẩn mở và là một một giao thức Authen được phân cấp. OpenID là một hệ thống đăng nhập một lần không có tính tập trung. Đối với những trang Website có sử dụng OpenID giúp người dùng giảm việc tái thiết lập các thông tin về tài khoản và mật khẩu cho các Website, họ chỉ cần đăng ký trước 1 tài khoản OpenID tại một trong những nhà cung cấp OpenID. Do OpenID không mang tính tập trung nên bất kỳ trang web nào cũng có thể sử dụng được OpenID như là một cách đăng nhập cho người dùng. OpenID là dạng liên kết URL, URL này có thể là tên miền của Website hoặc URL của nhà cung cấp định danh OpenID.

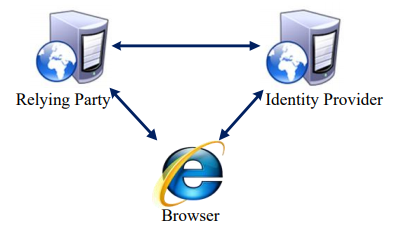
Trong hình ảnh ví dụ bên dưới, người dùng gửi yêu cầu đăng nhập tới ứng dụng, thay vì xử lý thông tin đăng nhập của người dùng, ứng dụng sẽ chuyển hướng chúng đến nhà cung cấp danh tính và xác thực danh tính. Sau khi xác thực, ứng dụng sẽ nhận được bằng chứng nhận dạng này và sẽ cho phép người dùng truy cập.



Với OpenID, chỉ duy nhất nhà cung cấp dịch vụ định danh quản lý password, và nhà cung cấp này này sẽ xác nhận định danh của bạn tới các website mà bạn đến thăm, ko có một website nào có thể biết được password của bạn - đây là một yếu tố bảo mật rất cao. Một vài tổ chức lớn cho phép hoặc phát hành OpenIDs or tái sử dụng lại OpenID ví dụ như Google, Facebook, Yahoo!, Microsoft, AOL, MySpace, Sears, Universal Music Group, France Telecom, Novell, Sun, Telecom Italia v.v....

### 1.2. Các thành phần của một hệ thống OpenID

c



Identity Provider (IdP): là thành phần có nhiệm vụ quản lý các thuộc tính định danh của người dùng hệ thống. IdP có chức năng truyền những thông tin cần thiết để thực hiện chứng thực đến Relying Party sau khi xác định đúng là người dùng đang sử dụng dịch vụ. Identity Provider – nhà cung cấp danh tính trong thông số kỹ thuật OpenID còn được gọi là nhà cung cấp OpenID (OP).

Relying Party: là dịch vụ sử dụng cơ chế định danh để chứng thực. Hiện nay có rất nhiều thành phần Relying Party trên mạng. Phần lớn đã hỗ trợ định danh bằng tài khoản của Facebook, Email, …

Identity Selector (Browser): là thành phần trung gian của hệ thống, là thành phần tương tác chính với người dùng.

## 2. Phương thức hoạt động của OPENID

### 2.1. Quy trình giao tiếp giữa các thành phần

OpenID có hai cơ chế giao tiếp: Smart mode và Dumb mode. Hai cơ chế này thực hiện trên khả năng của Relying Party. Ở chế độ Dumb người dùng không theo dõi trạng thái kết nối, vì vậy đối với mỗi người dùng cuối đăng nhập nên thực hiện theo toàn bộ bước xác thực. Mặt khác, ở chế độ Smart, người dùng duy trì dữ liệu trạng thái và bộ nhớ cache sau khi đăng nhập.

Quy trình định danh của hệ thống OpenID ở chế độ Smart có thể chia làm ba quy trình như sau:

* Xác định thành phần Identity Provider
* Quy trình gửi thuộc tính định danh
* Quy trình kiểm tra thuộc tính định danh

### 2.2. Chế độ Smart

#### 2.2.1. Quy trình xác định thành phần Identity Provider

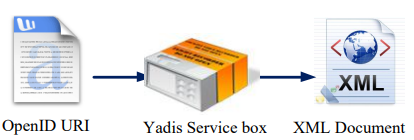


Quy trình xác định thành phần Identity Provider gồm các bước theo kịch bản sau:

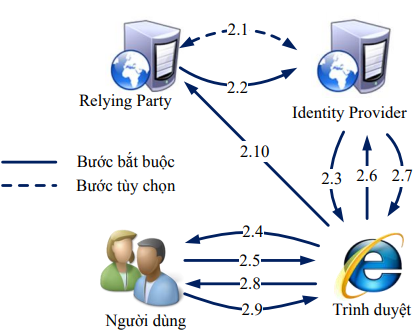
* Bước 1.1: Người dùng truy cập vào trang Web muốn đăng nhập. Trang Web yêu cầu các danh sách định danh để lựa chọn cho người dùng.
* Bước 1.2: Dựa vào dịch vụ định danh người dùng chọn, Browser sẽ giao tiếp với thành phần Relying Party.
* Bước 1.3: Relying Party trả về Browser trang đăng nhập có hỗ trợ OpenID, trong trang đăng nhập đó có yêu cầu người dùng nhập vào URI của Identity Provider.
* Bước 1.4: Browser trả về trang đăng nhập cho người dùng theo khuôn mẫu của Relying Party yêu cầu.
* Bước 1.5: Người dùng điền URI của Identity Provider vào trang đăng nhập Browser mới trả về. Và thực hiện “Đăng nhập”.
* Bước 1.6: Browser chuyển thông tin URI người dùng nhập vào đến Relying Party. Sau khi đã nhận được thông tin, Relying Party sẽ tiến hành dùng URI của người dùng nhập vào để xác định thành phần Identity Provider.

URI sẽ có hai loại:

* Loại 1: URI người dùng nhập chính là địa chỉ của Identity Provider. Khi đó, Relying Party đã có được địa chỉ của Identity Provider chính là URI người dùng nhập vào cung cấp.
* Loại 2: URI này không phải là địa chỉ của Identity Provider. Trong trường hợp này, Relying Party sẽ dùng Yadis để lấy địa chỉ của Identity Provider. Dịch vụ Yadis có vai trò nhận vào một URI và sẽ trả về địa chỉ, thông tin Identity Provider tương ứng. Dưới đây là hình ảnh minh họa về quá trình xác định địa chỉ Identity Provider với Yadis.



#### 2.2.2. Quy trình gửi thuộc tính định danh



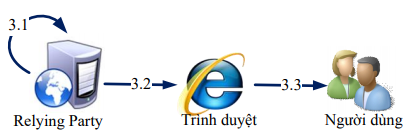
Quy trình gửi thuộc tính định danh lên Relying Party gồm các bước như sau:

* Bước 2.1: Relying Party sau khi xác định được thành phần Identity Provider ở quy trình xác định thành phần Identity Provider, bước 2.1 là bước tùy chọn xảy ra hai trường hợp sau:
* Trường hợp 1: Relying Party và Identity Provider chưa có khóa chia sẻ bí mật ở những lần định danh trước, hoặc khóa đã hết thời gian sử dụng. Khi này, Relying Party sẽ kết nối bằng một kênh truyền an toàn với Identity Provider để chia sẻ khóa mật. Khóa mật sẽ được sử dụng để kiểm tra các thuộc tính định danh ở quy trình kiểm tra thuộc tính định danh sau này.
* Trường hợp 2: Nếu Relying Party đã có khóa mật ( chưa hết thời gian sử dụng), thì không cần phải thực hiện bước 2.1.

*(Vì vậy, bước 2.1 chỉ xảy ra khi khóa mật của Relying Party chưa có hoặc khóa hết thời gian sử dụng).*

* Bước 2.2: Relying Party gửi các thuộc tính yêu cầu Identity Provider cung cấp chức thực.
* Bước 2.3: Identity Provider sẽ yêu cầu người dùng đăng nhập với biểu mẫu trả về Browser.
* Bước 2.4: Browser hiển thị trang đăng nhập tương tác với người dùng.
* Bước 2.5: Người dùng đăng nhập với các thông tin yêu cầu của Identity Provider.
* Bước 2.6: Browser chuyển thông tin đăng nhập đến Identity Provider để kiểm tra.
* Bước 2.7: Identity Browser sẽ kiểm tra thông tin đăng nhập. Dựa trên danh sách các tên thuộc tính yêu cầu từ Relying Party, Identity Provider sẽ tạo một thông điệp chứa các thuộc tính tương ứng và ký tên danh sách các thuộc tính định danh. Cuối cùng trả về Browser.
* Bước 2.8: Browser hiện lên tất cả thuộc tính định danh nhận được từ Identity Provider cho người dùng.
* Bước 2.9: Người dùng kiểm tra các thuộc tính định danh có hợp lệ hay không. Sau đó, người dùng xác nhận gửi các thuộc tính định danh.
* Bước 2.10: Browser truyền các thông tin định danh của người dùng đến Relying Party.

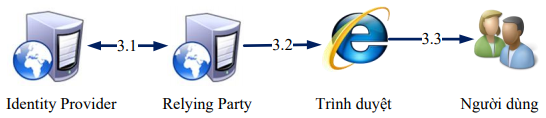
#### 2.2.3. Quy trình kiểm tra thuộc tính định danh



Quy trình kiểm tra thuộc tính định danh gồm các bước sau:

* Bước 3.1: Relying Party dựa trên thuộc tính định danh nhận được ở quy trình gửi thuộc tính định danh. Relying Party sẽ kiểm tra xem thuộc tính định danh này có hợp lệ hay không.
* Bước 3.2: Relying Party trả kết quả định danh về Browser.
* Bước 3.3: Browser hiển thị kết quả định danh đến người dùng.

### 2.3. Chế độ Dump



Ở chê độ Dump tương tự như chế độ Smart. Chế độ Dump, Relying Party không có khả năng lưu trữ các thông tin trước đó. Vì vậy, thành phần Identity Provider và Relying Party sẽ chưa tồn tại khóa chia sẻ để kiểm tra thuộc tính định danh.

Do đó, ở quy trình kiểm tra thuộc tính định danh: Bước 3.1 sẽ là bước Relying Party kết nối an toàn với Identity Provider để có thể kiểm tra thuộc tính định danh. Các bước tại 3.2 và 3.3 giống với chế độ Smart.

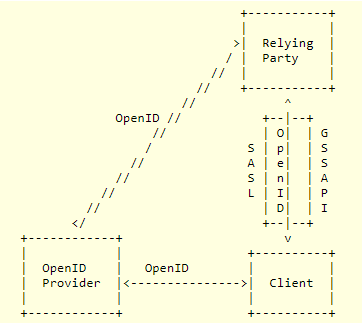
### 2.4. Cơ chế xác thực của OpenID

OpenID là giải pháp cho Web đăng nhập một lần. Lớp bảo mật và xác thực đơn giản ( Simple Authentication and Security Layer – SASL) và Giao diện chương trình ứng dụng bảo mật chung (Generic Security Service Application Program Interface – GSS-API) là các cơ chế khái quát hóa xác thực.

SASL được sử dụng bởi các giao thức ứng dụng như IMAP, POP, XMPP với mục tiêu mô-đun hóa các lớp xác thực và bảo mật.

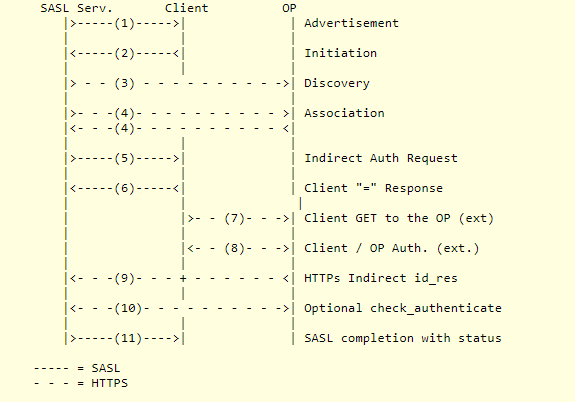
GSS-API cung cấp một khuôn khổ cho các ứng dụng để hỗ trợ nhiều cơ chế xác thực thông qua một giao diện thống nhất.

Hình dưới mô tả sự tương tác của OpenID và SASL, để cải tiến giữa Relying Party và Client, thông điệp yêu cầu của OpenID được truyền qua đường hầm với cơ chế SASL / GSS-API.



*Mô tả sự giao tiếp giữa OpenID và SASL*

* **Xác thực SASL**



1. Relying Party hoặc máy chủ SASL thông tin tới Khách hàng rằng có hỗ trợ cơ chế SASL OpenID
2. Máy khách bắt đầu xác thực SASL và truyền mã định danh do người dùng cung cấp làm phản hồi đầu tiên.
3. Sau khi chuẩn hóa mã định danh do người dùng cung cấp, bên Relying Party thiết lập URL điểm cuối OP mà người dùng cuối sử dụng để xác thực.
4. Bên Relying Party và OP thiết lập một liên kết bí mật chung bằng cách sử dụng trao đổi khóa Diffie-Hellman. OP sử dụng một liên kết để ký các thông điệp tiếp theo và Relying Party xác minh các thông điệp đó, điều này loại bỏ được sự cần thiết của xác minh chữ ký sau mỗi yêu cầu/ phản hồi yêu cầu trong các lần yêu cầu trực tiếp tiếp theo.
5. Relying Party yêu cầu xác thực OP để có được một khẳng định dưới dạng một yêu cầu gián tiếp. Các thông điệp này truyền qua máy khách thay vì truyền trực tiếp giữa Relying Party và OP. OpenID định nghĩa hai phương thức để giao tiếp gián tiếp là chuyển hướng HTTP và gửi biểu mẫu HTML. Cả hai cơ chế này không được áp dụng trực tiếp để sử dụng SASL. Vì vậy, yêu cầu nội dung thông điệp được mã hóa bằng URI.
6. Máy khách SASL bây giờ gửi phản hồi chỉ ra rằng xác thực tiếp tục thông qua dòng OpenID bình thường.
7. Tại thời điểm này, ứng dụng khách phải xây dựng một URL chứa nội dung nhận được trong thông điệp trước đó từ Relying Party. URL này được truyền đến OP bằng ứng dụng máy khách SASL hoặc trình xử lý thích hợp, chẳng hạn như trình duyệt.
8. Tiếp đến, khách hàng tùy chọn xác thực OP và sau đó phê duyệt hoặc từ chối xác thực cho bên Relying Party.
9. OP sẽ truyền tải thông tin về sự thành công hoặc thất bại của giai đoạn xác thực trên trở về RP, một lần nữa sử dụng phản hồi gián tiếp thông qua trình duyệt. Máy khách truyền qua HTTP/TLS chuyển hướng kết quả OP sang RP.
10. RP có thể gửi yêu cầu OpenID check\_authentication trực tiếp đến OP.
11. Máy chủ SASL gửi phản hồi SASL thích hợp cho máy khách.

## 3. Những vấn đề phát sinh với OPENID

Hiện nay, hệ thống OpenID đang được sử dụng rất rộng rãi. Nhưng OpenID vẫn còn tồn đọng một số vấn đề cần xem xét.

**3.1. Tính nặc danh**

Nếu sử dụng Identity Provider là tổ chức không đáng tin cậy, Identity Provider có thể có được rất nhiều thông tin người dùng về các website đã truy cập. Ví dụ, Identity Provider có thể lưu vết những hoạt động của người dùng bằng cách lưu những website người dùng sử dụng, thời gian sử dụng… Đây là những thông tin nhạy cảm có thể ảnh hưởng đến người dùng nếu thuộc tính này bị lộ ra bên ngoài. Một số đề xuất:

* Người dùng lựa chọn thành phần Identity Provider đáng tin cậy.
* Người dùng phải đọc kỹ các chính sách của Identity Provider để cung cấp các thuộc tính định danh của mình.
* Nên xem xét tự xây dựng Identity Provider riêng cho các hệ thống lớn.

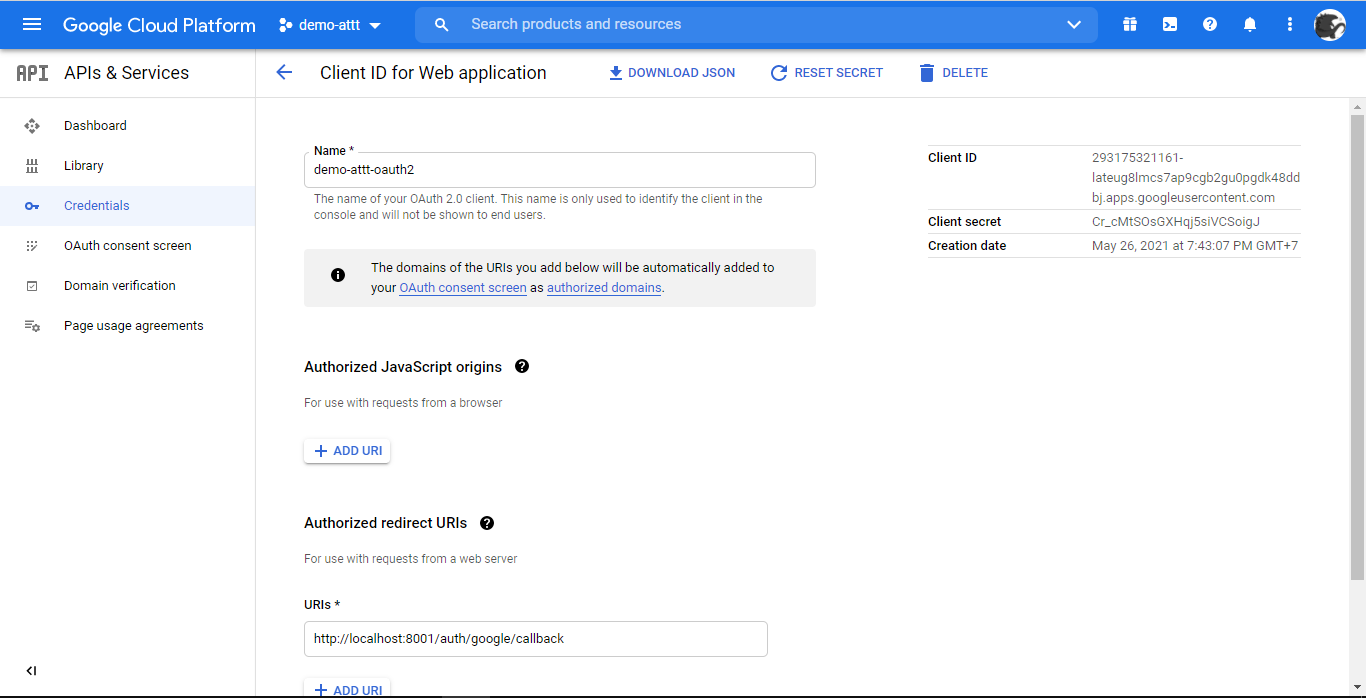
**3.2. Độ an toàn thông tin**

Độ an toàn thông tin thể hiện qua khả năng của hệ thống có thể chống lại các phương án tấn công để lấy thuộc tính định danh của người dùng. Dưới đây là một số phương pháp tấn công ảnh hưởng đến hệ thống OpenID:

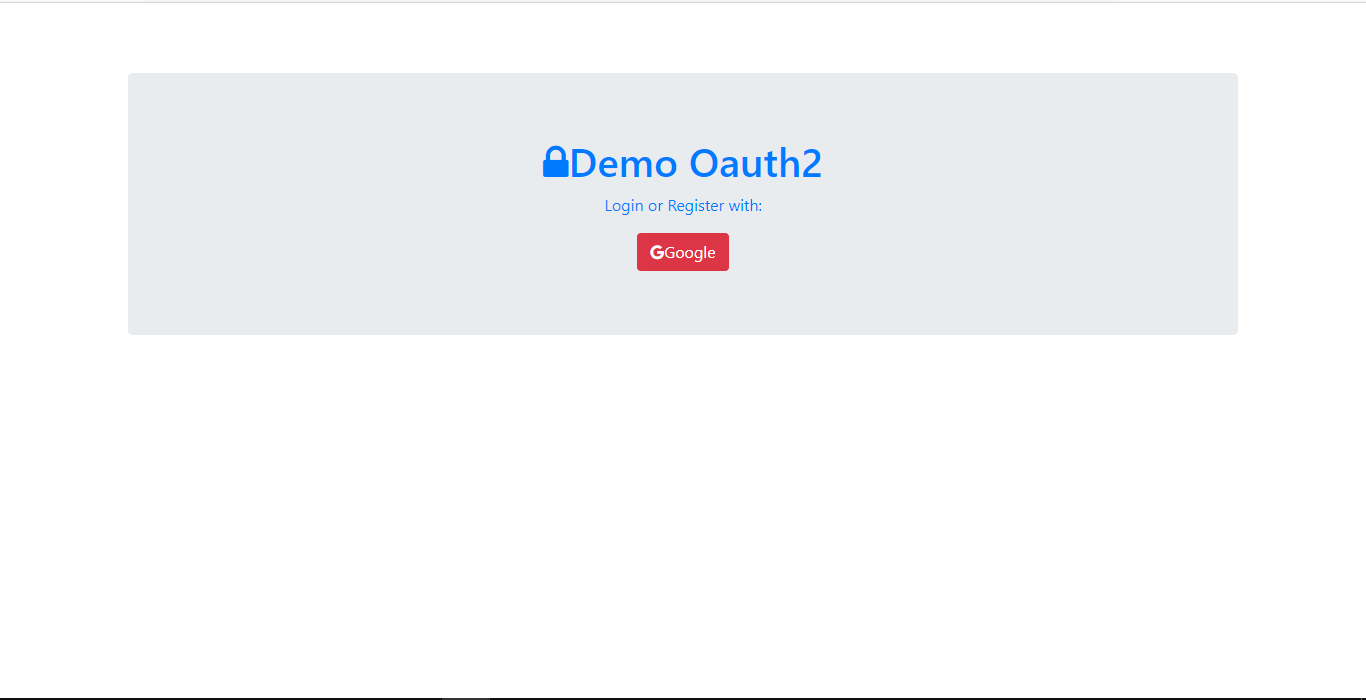
* Tấn công Replay: Trong một số trường hợp, OpenID dễ dàng bị tấn công phát lại. Xác suất bị tấn công được hạn chế bởi sự có mặt của giá trị ngẫu nhiên phát sinh (nonce) kèm một nhãn thời gian (timestamp) trong các thông điệp của OpenID. Tuy nhiên, nếu Relying Party không lưu trữ các giá trị ngẫu nhiên phát sinh hoặc chứa một nhãn thời gian quá dài thì xác suất bị tấn công replay rất cao. Để tránh bị tấn công Replay, các giải pháp khắc phục là:
* Relying Party và Identity Provider nên dùng NTP ( Network Time Protocol) để giữ cho đồng hồ luôn được đồng bộ theo một đồng hồ chuẩn.
* Relying Party nên từ chối những thông điệp với timestamp quá lớn so với thời gian hiện hành tại Relying Party.
* Tấn công phishing: Khả năng OpenID bị tấn công phishing là rất cáo.
* Vấn đề bảo mật đường truyền: Sử dụng SSL rất quan trọng đối với hệ thống OpenID, SSL mã hóa tất cả dữ liệu tại tầng vận chuyển và được sử dụng phổ biển như một phương thức bảo mật khi truyền dữ liệu trên mạng. SSL được đề nghị cho mọi giao tiếp giữa Relying Party, Identity Provider và Browser.
* Vấn đề về lịch sử trình duyệt: Do một số thông điệp OpenID được chuyển bằng HTTP GET, vì vậy có khả năng những thông điệp này được lưu trữ tại lịch sử trình duyệt ( Browser History). Nếu một người tấn công truy cập được vào máy, lịch sử trình duyệt có thể chứa rất nhiều thông tin quan trọng liên quan đến người dùng. Vấn đề này thường xảy ra khi người dùng sử dụng máy tính công cộng. Giải pháp cho vấn đề này là nhân cao nhận thức và hiểu biểu của người dùng. Người dùng nên xóa tất cả toàn bộ lịch sử trình duyệt sau khi sử dụng xong.

# CHƯƠNG 4: CHƯƠNG TRÌNH DEMO

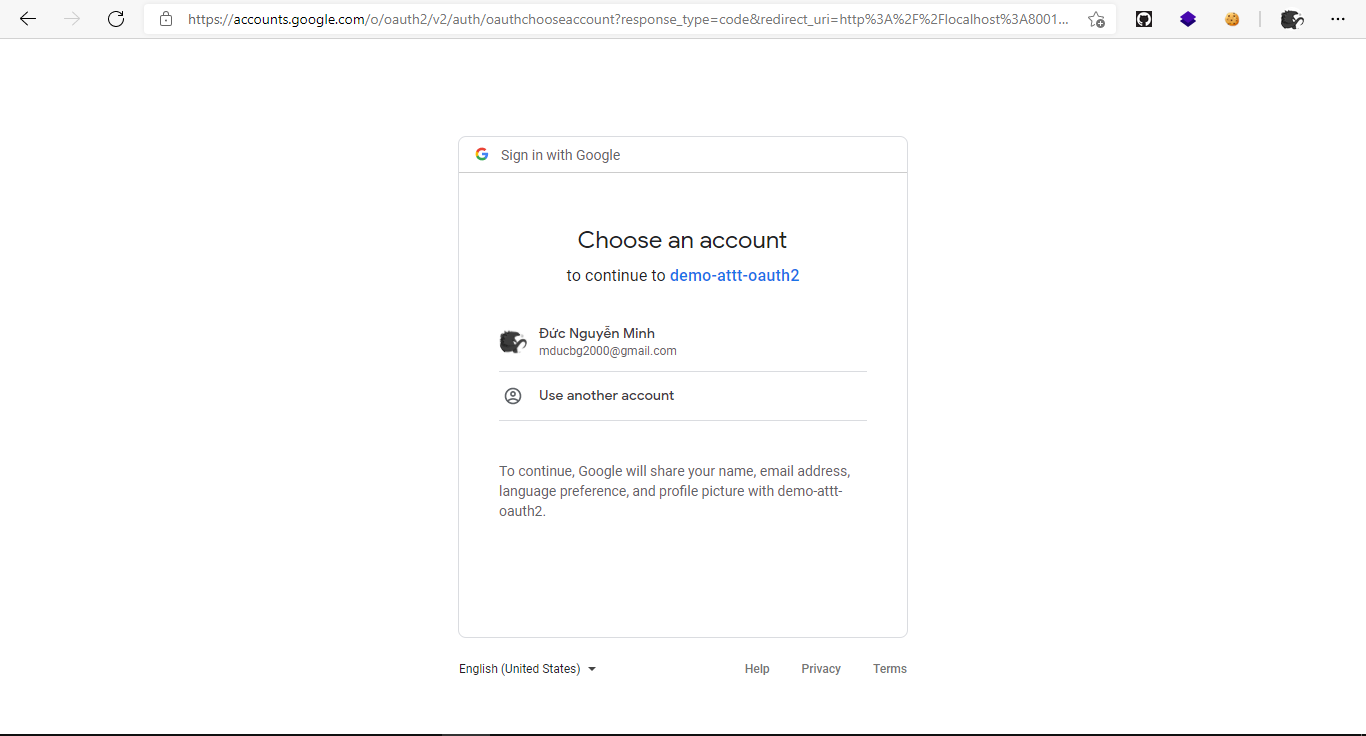
* Đăng ký project trên google console



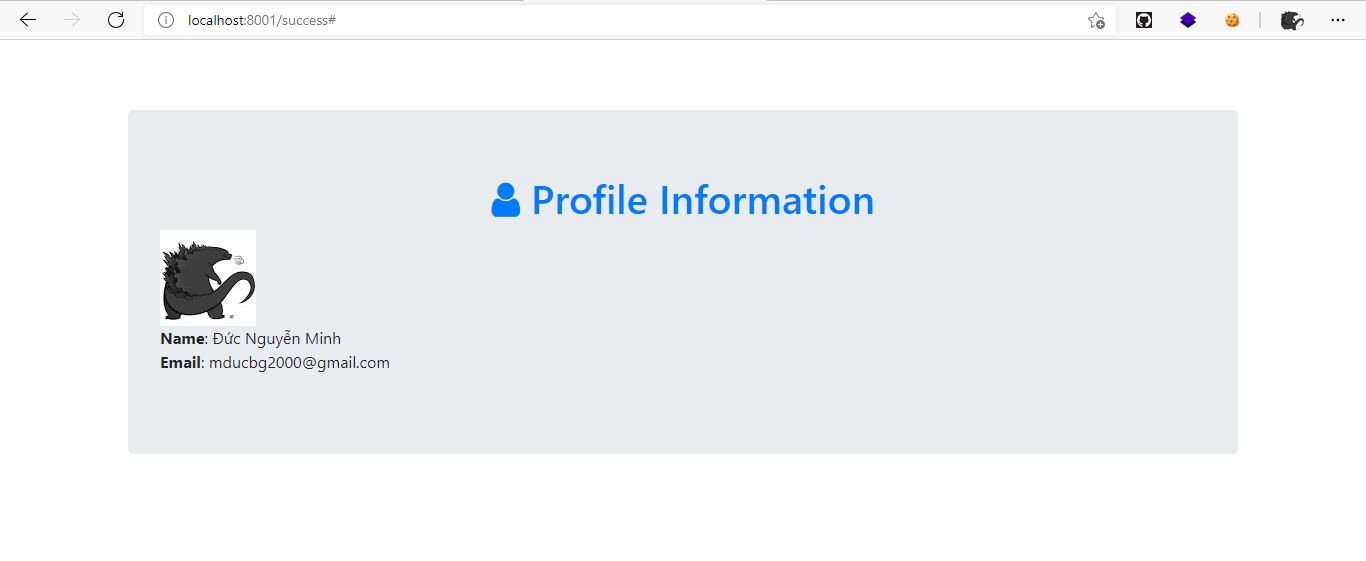
* Màn hình đăng nhập



* Sau khi click button, hiển thị màn hình consent cho người dùng chọn tài khoản google



* Profile trả về của tài khoản gồm có avatar, tên và email



# DANH MỤC TÀI LIỆU THAM KHẢO

[1] A SASL & GSS-API Mechanism for OpenID ( <https://tools.ietf.org/id/draft-ietf-kitten-sasl-openid-07.html> )

[2] <https://www.computerweekly.com/news/2240033602/What-is-OpenID-How-to-use-OpenID-SSO-in-your-organisation>

[3] https://viblo.asia/p/single-sign-on-sso-la-gi-hoat-dong-ra-sao-bWrZn4oQ5xw

[4] <https://viblo.asia/p/tim-hieu-oauth-voi-openid-connect-1VgZv82R5Aw>

[5] <https://blog.runscope.com/posts/understanding-oauth-2-and-openid-connect#:~:text=The%20four%20grant%20types%20%2D%20Authorization,used%20in%20different%20use%20cases>.

[6] <https://stfalcon.com/en/blog/post/oauth-2.0>

[7] https://www.oauth.com/