# Chapter 1: The Hydra of Modern Identity

## Identity Challenges

identity management là một khái niệm đơn giản trong lý thuyết, nhiều yếu tố cần làm việc cùng nhau để làm cồn việc tốt hơn trong thực tế. Nó yêu cầu việc lên kế hoạch, thiết kế và phát triển một cách cẩn thận để thực thi identity management cho một application trong khi đó cân bằng vô số kì vọng xuất phát từ yêu cầu về business và bảo mật. Thật không may, identity management không phải là one-size-fits-all. Không có một master solution nào chúng ta có thể cung cấp, cái mà có thể phù hợp với mọi case

Ví dụ, ở đây có một vài thách thức khác nhau mà ta có lẽ nghĩ về nó. Một application có lẽ có nhiều users người mà có thể login nhanh chóng bằng việc sử dụng Facebook. Nhiều người thậm chí muốn có thể sử dụng multiple social để login tới application và vẫn ghi nhận là cùng một person. Bạn sẽ cần xử lý vấn đề này nếu không muốn users có thể từ bỏ ứng dụng của bạn bởi vì nó bất tiện để sign up và login. Mặt khác, các users là nhân viên cần truy cập tới các ứng dụng thông qua tài khoản làm việc của họ, và họ muốn sự tiện lợi của một tài khoản duy nhất. Hơn nữa, các công ty tổ chức có yêu cầu phức tạp về việc cấp quyền, thường là dựa trên roles, để quản lý quyền cho các nhân viên trong tổ chức đó có thể làm.

Một application với nội dung nhạy cảm có lẽ yêu cầu mạnh mẽ hơn về forms của authentication chứ không chỉ là một password đơn thuần, yêu cầu đầy đủ về bảo mật nhưng vẫn thân thiện với users. Có nhiều options mạnh cho authentication. Ta cần các solutions dễ dàng cho users chấp nhận và sử dụng bởi các solution cồng kền có lẽ dẫn tới users phá vỡ solution hoặc đơn giản là từ bỏ application.

Nếu bạn phục vụ nhiều applications, hoặc thậm chí một cổng hỗ trợ bên ngoài tới application của bạn, các users của bạn sẽ muốn đăng kí một lần và sau đó có thể access trên nhiều applications. Điều này chung cấp tính tiện lợi tới users cũng như là một chỗ để control chính sách authentication. Tuy nhiên, authentication là cổng tới application của bạn. Nó phải có tính sẵn sàng cao và có khả năng mở rộng cao hơn application của bạn; vì vậy, nó sẽ nó đột nhiên trở thành chướng ngại vật chứ không phải là một cổng. Single sign-on không hiệu quả nếu nó làm cản trở khả năng truy cập tới application của bạn.

Application của bạn thiết kế có lẽ cần để chứa đựng nhiều constraints từ platform. Trên Web, một user có lẽ muốn browser chuyển hướng tới một sign-in page cho việc authentication. Trái ngược lại, các users của native desktop applications có lẽ thích login đã được nhúng bên trong application hoặc tận dụng một session đã được cung cấp bên dưới hệ thống. Khác nữa là các applications mobile có lẽ sữ dụng các cách tiếp cận khác. Có lẽ chuyển hướng bạn tới identity provider để log in, nhưng nhiều cái khác có lẽ vẫn sử dụng trực tiếp bên trong application. Bạn sẽ cần cân đo các cách tiếp cận và thiết kế cho ux thích hợp cho ứng dụng của bạn.

Việc design identity management cho application là cần thiết để trả lời cho tất cả các câu hỏi và nắm giữ tài khoản phù hợp cho tất cả yêu cầu business liên quan.

## Objective

Mục tiêu trong việc viết cuốn sách này là cung cấp cho bạn topic về identity management, dựa trên kinh nghiệm của chúng tôi về việc xây dựng và deploying applications. Trọng tâm là các khía cạnh của identity management cho software applications, như là việc tạo accounts, authentication, API authorization, single sign-on, account management, and logging users out. Để thiết lập các mong đợi thực tế, identity management là một topic lớn. Nên cover tất cả trong một cuốn sách là khó có khả năng.

Chúng ta sẽ cover 3 giao thức identity pohoor biến là OAuth 2.0, OIDC, và SAML 2.0, vấn đề của mỗi cái đã thiết kế để giải quyết, cách mà chúng làm việc và cách để thực thi authentication và authorization cho các trường hợp đơn lẻ, và khắc phục sự cố. CHúng ta không thể cover mọi biến hoặc mọi trường hợp, nhưng bạn có lẽ hiểu vơ bản về mỗi protocol và cách sử dụng nó.

Chúng tôi đã tổ chức nội dung xoay quanh các sự kiện trong cuộc sống về identity. Chúng tôi bắt đầu với việc thảo luận account và một vài options. Sau đó chúng tôi chia vào trong API authorization và authentication và cung cấp cái nhìn tổng quát về 3 giao thực sử dụng ngày nay là OAuth 2.0, OpenID Connect và Security Asertion Markup Language (SAML 2.0). Các chapters này cover việc xác thực users và xử lý cấp quyền cho applications và APIs. Sau đó covering các cơ chế cơ bản của các giao thức, chúng ta có một chapter giải thích một sample cái mà đồng hành với cuốn sách này và cách nó sử dụng các protocols.

Các chapter sau cover vấn đề gặp phải sau khi user logins in lần đầu thiên, với giới thiệu thông tin về sesions, single sign-on, stronger forms of authentication, account management, logout and deprovisioning. Trong trường hợp application của bạn hkoogn hoạt động hoàn hỏa trong lần đầu, chúng tô đã thêm một chapter với guidance on troubleshooting. Chúng tôi chúng chia sẽ thoong tin trên các kịch bản vấn đề có lẽ nảy sinh, và hơn nữa là các trường hợp bất thường mà tôi từng gặp. Chúng ta kết thúc nhanh chóng tổng quan về sự tuân thủ cũng như sai lầm dẫn đến một số điều rất đáng tiếc. Cũng có thể học hỏi từ quá khứ

Chúng tôi đề xuất đọc các chapters theo thứ tự, ít nhất là qua chapter 15, cũng như nhiều chapters build phía trước. Chúng tôi yêu cầu đặc biệt đọc chapter 4 đến 9 vì chúng liên kết với nhau chặt chẽ theo thứ tự từng chường. Các chapters sau 15 có thể đọc theo bất cứ thứ tự nào. Chapter 16 cần khi debug. Chapter 18 về các yêu cầu ít phổ biến hơn có lẽ có giá trị để đọc sơm trước trong một dự án vì nó giúp bạn identity items để đưa vào kế hoách của dự án. Chepter 17 và 19 cover các lỗi khác nhau và sẽ giúp bạn lên kế hoạch để trách sai xót

Trong các chapters OAuth 2.0 và OIDC, chúng tôi đã cung cấp samples về HTTP requests được tạo bởi một application.

## Sample Application

Chapter 9 giải thích sample application cách mà nó được thiết kế để sử dụng identity proptocols như là một phần của identity management solution

# Chapter 2 The Life of an Identity

## Terminology

Các khái niệm về một identity, identifer, và account có liên quan chặt chẽ với nhau nhưng thực chất là khác nhau. Chúng ta sử dụng khái niệm “identifer” để nhìn đến một thuộc tính duy nhất mục đích là để định danh một người, entity là duy nhất (nôm na thì thằng này là một thuộc tính chứa giá trị là duy nhất để phân biệt đối tượng này với đối tượng khác, ta hay lấy tên thuộc tính này là id), bên trong một context xác định. Email addresses, passport numbers, driver’s license numbers, and employee numbers và nhiều ví dụ khác về identifers được sử dụng cho mọi người. Các entities không phải người như là agents, bots, hoặc devices, có lẽ cũng được định danh bởi một alphanumberic string of characters được gán tại lần đầu tiên chúng được tạo ra hoặc đăng kí bên trong một context nơi mà chúng sẽ hành động. Identyfers cho phép chúng ta refer tới chính xác người hoặc entity không phải người và là bản chất để identity manager.

Khái niệm “identity” được định nghĩa là một nhóm các thuộc tính có mối quan hệ với nhau với một người hoặc một entity xác định trong một bối cảnh cụ thể. Một identity bao gồm một hoặc nhiều identifers và có lẽ chứa các thuộc tính khác giống như name, age, address, phone number, eye color, and job title. Các identities không phải người có lẽ có thêm các thuộc tính như một owner, IP address, một mô hình hoặc version number. Các thuộc tính tạo lên một identity có lẽ được sử dụng cho authentication và authorization cũng như truyền đạt thông tin về identity tới application (nôm na thì identity là bao gồm tất cả các thuộc tính tạo lên một entity, mô hình hóa entity với các thuộc tính cơ bản để nhận biết nó với loại entity khác trong một context cụ thể)

Một người có lẽ có nhiều hơn một identity. Cũng như một người có lẽ đóng là nhiều nhân vật khác nhau trong nhiều context khác nhau (ví dụ một người trong công ty anh ta có thể là giám đốc, trong gia đình là một người cha, trong thể thao là một đội trưởng), một person có thể có multiple identities. Người ta có lẽ có một identity về công việc được sử dụng để thực hiện các tasks. The identity attributes có lẽ bao gồm một identifer được phát bởi employer, một department name, building location, and manager. Một người cũng có lẽ có một vài identities nhân vật được sử dụng cho các mục đích khác nhau, bao gồm việc quản lý một team thể thao trẻ hoặc running a side business. Có nhiều ví dụ trong thực tế mô tả các khía cạnh do chúng ta định nghĩa. Một identity online bao gồm ít nhất một identifer và một tập các thuộc tính cho user hoặc entity trong một context cụ thể.

Một identity có mối quan hệ với một account trong mỗi bối cảnh. Chúng ta định nghĩa một account như là một local construct (kết cấu cục bộ) bên trong một application cái mà được sử dụng để thực hiện các hành động bên trong context này. Identity attributes có lẽ được chứa bên trong một application’s account object hoặc chúng có lẽ được chứa tách biệt và được tham chiếu từ account object. Một account có lẽ có identifer của chính nó để identity có mối liên hệ với nó. Việc có một account identifer tách biệt với identity đã liên kết với account cung cấp mức độ tách biệt. The account identifer có thể được sử dụng trong hồ sở application khác để làm nó dễ dàng hơn cho các users thay đổi username hoặc identifer khác có mối quan hệ với account.

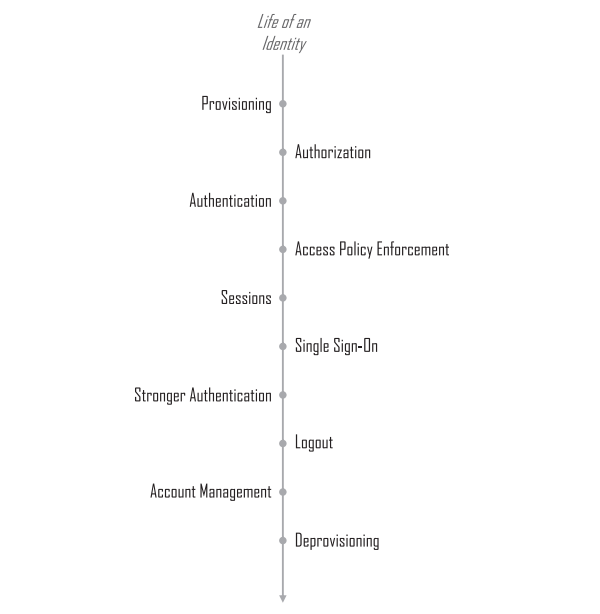
Chúng ta sẽ sử dụng khái niệm “identity” khi đặc biệt referring tới online identities như là một set of attributes về một người hoặc một entity. Chúng ta sẽ sử dụng khái niệm account khi referring tới một account là một kết cấu bên trong một application hoặc service cái mà có một identity đã liên kết với nó. Chú ý rằng một account có thể có nhiều hơn một identity liên kết với nó thông qua account linking cái sẽ được giải thích nhiều hơn trong Chapter 18. Tổng kết lại, một person logs in để sử dụng một account cái mà có vài identity có liên kết với nó và cho phéo chúng thực hiện các hành động bên trong một hệ thống.

Các yếu tố không phải người có thể có identities. Software components phục vụ như là agents hoặc bots và các thiết bị thông minh có thể có identities và có thể tương tác với software hoặc thiết bị khác trong nhiều cái mà yêu cầu authentication và authorization giống như các yếu tố con người. Trong cuốn sách này chúng ta chỉ thảo luận về online identities

Có lẽ như bạn đoán, một Identity Management (IdM) System là một set of service cái mà hỗ trợ creation, modification, and removal of identities và liên kết các accounts, cũng như authentication and authorization được yêu cầu để access resouruces. Identity Management systems được sử dụng để bảo vệ tài nguyên

## Events in the Life of an Identity

Với định nghĩa cơ bản trên một cách, chúng ta có thể chuyển tới các events cơ sở trong life of an identity được minh họa trong hình sau.



### Provisioning

Bước đầu tiên trong life of an identity là its creation. Việc tạo một account và liên kết identity infomation thường được refered tới như là provisioning. Provisioning có lẽ đã được hoàn thành bới việc có users register, nhập identity info từ hệ thống di sản hoặc tận dụng một external identity service. Bất kể cơ chế đã sự dụng là gì thì mục tiêu của giai đoạn the provisioning là để thành lập một account với liên kết identity data. Điều này liên quan đến việc có đưaọc hoặc việc gán một identifer của identity.

Ví dụ, một user có tên là Alice mong muốn sử dụng các dịch vụ banking online. Alice có lẽ thành lập một account online tại ngân hàng bởi việc điền một account registration form. Alice sẽ cung cấp identity infomation bao gồm username, password, her name, home address,... Dữ liệu này sẽ được sử dụng để provision một account online tại ngân hàng có liên kết với Alice’s personal identity.

Alice có thể tạo nhiều tài khoản online khác nhau. Trong the personal account, Alice có thể thiết lập danh tính thứ hai là một small business với tài khoản thứ 2 sử dụng cho business identity của cô ấy và taxID.

### Authorization

Khi một account đã được tạo, nó cần chỉ định là account này có thể làm gì. Chúng ta sử dụng khái niệm authorization cho việc trao đặc quyền (the granting of privileges) cái mà quản lý một account được cho phép làm gì.

Khi Alice tạo account của cô ấy xong, ngân hàng cấp quyền cho account của cố ấy để có thể truy cập vào application tới view kiểm tra các accounts. Nếu cô ấy không có tài khoản môi giới tại ngân hàng, cô ấy không thể được cấp quyền để truy cập vào ứng dụng giao dịch của ngân hàng. Cũng có thể nói, tài khoản của cô ấy không được quyền để xem thông tin account các khách hàng khác của ngân hàng. Alice’s authorization chỉ ra rằng đặc quyền tài khoản của cô ấy đã được cấp. Authorization cho một account tại một thời điểm đó một account được tạo và có thể được cập nhật theo thời gian.

### Authentication

Để truy cập vào nội dung trực tuyến cái mà không public, một user cần authenticate. Một user cung cấp một identifer để đăng kí account họ sử dụng và login credentials cho account. Nếu mấy cái này hợp lệ credentials đã được đăng ký từ trước trong giai đoạn the account provisioning. The credentials liên quan something the user knows, something the user has, and/or something the user is. Ví dụ, password là something the user know. A numeric code generated từ thiết bị đã đăng kí trước đó là something the user has. Sinh chắc học như là vân thay thì là something the user is. Authentication với một hoặc nhiều credentials cái mà đã hợp lệ đối với các thông tin đã đăng ký từ trước. Một user đúng sử dụng account để access tài nguyên được bảo vệ.

Sau khi Alice thiết lập identity và acount tại ngâng hàng, cô ấy có thể truy cập các services của ngâng hàng. Để truy cập tài nguyên được bảo vệ, như là việc cô ấy kiểm tra số dư tài khoản, cô ấy cần authenticate bởi việc nhập username và pass đã thiết lập trong bước đăng kí. The username chỉ ra tài khoản cô ấy muốn sử dụng và pass mô tả rằng cô ấy là đúng người để sử dụng tài khoản

### Access Policy Enforcement

Khi mà user đã xác thực và đã liên kết với một tài khoản, nó cần tuần theo chính sách truy cập để chắc chắn rằng bất kì hành động nào được tạo bởi user thì được phép bởi các đặc quyền họ đã được trao. Chúng ta sử dụng khái niệm access policy enforcement cho việc thực thi chính sách access được chỉ định bởi authorization. Nói cách khác, authorization chỉ định cái mà user hoặc entity được phép làm, và access policy enfor kiểm tra request của user được cho phép bởi các đặc quyền mà họ đã được cấp để sử dụng.

Khi Alice logs in vào ứng dụng bán lẻ của ngân hàng và request, ứng dụng sẽ kiểm tra cô ấy có quyền để tạo request này không. Nếu cô ấy bị giới hạn access tới các services trading, cô ấy sẽ bị từ chối cũng như là không được cấp quyền để truy cập. Trong trường hợp này, ứng dụng có lẽ hiển thị message chỉ ra rằng cô ấy không được phép.

### Sessions

Khi mà user đã xác thực và đã được cấp quyền, họ sẽ thực hiện các hành động bên trong ứng dụng. Nhiều ứng dụng chỉ cho phép một user tồn tại hoạt động trong một khoảng thời gian giới hạn trước khi yêu cầu user xác thực lại. Họ làm điều này bằng việc quản lý một session cho user. Một session theo dõi thông tin như là liệu user đã được xác thực chưa, thường là cơ chế xác thực hoặc strength level đã sử dụng. Cái này cho phép một ứng dụng biết khi nào user sẽ được thông báo là xác thực lại.

Đội dài thời gian user được cho phép hoạt động trước khi cần xác thực lại được biết như là một session limit hoặc session timeout. The sesion timeout setting sẽ thường khác nhau bởi độ nhạy cảm dữ liệu của ứng dụng. Session limits giúp bảo vệ và chống lại những users người mà đi ra khỏi màn hình của họ mà không cần logging off và identity infomation có thể đã thay đổi . Một Session limit cái mà bắt user xác thực lại cung cấp kiểm tra rằng nó vẫn là user hợp pháp tại bàn phím.

### Single Sign-On

Sau khi user truy cập một ứng dụng, họ có lẽ muốn làm nhiều cái liên quan đến application khác. SSO là khả năng log in một lần và sau đó access thêm protected resources hoặc các applications với cùng authentication mà không cần xác thực lại.

Khi Alice access web site ngân hàn của cô ấy, signle sign-on sẽ cung cấp tiện nghi truy cập tới các services khác của banking. Nếu Alice signed up cho services bản tin đầu tư của ngân hàng cô ấy, cô ấy có thể log in để access đầu tiên là vào ứng dụng bán lẻ của ngân hàng để xem số dư của tài khoản của cô ấy, sau đó truy cập tới ứng dụng thông tin đầu tư mà không cần sign in lần nữa.

Signle sign-on có sẵn khi một set of applications đã ủy quyền xác thực tới cùng entity. Một authenticated sesion trong entitu có thể được sử dụng để truy cập nhiều tài nguyên thông qua sign-on thì thường được gọi là SSO session

### Stronger Authentication

Step-up authentication và muilti-factor authentication (MFA) liên quan đến việc xác thực một user với stronger forms của authentication. Nhiều forms của authentication giống như là username-password authentication được xem xét là yếu bởi vị họ liên quan tới một single factor (yếu tố duy nhất), the password, cái này có thể bị chụp lại và dễ dàng được sử dụng bởi thằng khác. Stronger forms của authentication liên quan đến các yếu tố khác như là user has, user is. Authentication cái mà yêu cầu multiple factors tại cùng một thời điểm được biết là multi-factor authentication. Multi-factor authentication thường liên quan tới một password cũng như chiếm hữu của thiết bị như laptop hoặc mobile phone hoặc yếu tố sinh trắc học như vân tay hoặc giọng nói.

Step-up authentication thì là hành động nâng cao một authentication session đã có tới level cao hơn đảm bảo bởi việc xác thực với stronger form của authentication. Ví dụ một người có thể sử dụng username và password để thiết lập một authentication session, sau đó, để truy cập vào các tính năng nhạy cảm của application ta cần xác thực cấp cao hơn, the user sẽ được thông abo cho việc thêm credentials như là one-time password đã generated ở mobile phone. Step-up authentication có lẽ được yêu cầu khi user access tới nhiều ứng dụng nhạy cảm hjoawcj khi học bị hạn chế thực thi các giao dịch nhạy cảm.

### Logout

Khi một user đã hoàn thành công việc với một application, họ sẽ chấm dứt session của họ với logging out. Hành động logging out sẽ chấm dứt session của ứng dụng. Nếu nọ quay lại application, họ cần xác thực lại trước khi có thể access. Trong trường hợp single sign-on được sử dụng, có nhiều sessions để chấm dứt, và nó là quyết định thiết kế để các sessions này sẽ bị chấm dứt khi user logs out.

Hành động logging out thì khác một chút sơ với session timing out. The user yêu cầu rõ ràng rằng session của họ bị chấm dứt. Khi một session times out, một application có lẽ chọn để giữa trạng thái của session và phục hồi session nếu người dùng xác thực lại

### Account Management and Recovery

Account management bao gồm các tính năng cho phép users và administrators xem và update user profile attributes liên kết với một identity. Và các tính năng liên quan đến quên pass các thứ.

# Chapter 3: Evolution of Identity

Qua các năm, có một sự tiến hóa đang diễn ra trong cách mà identity infomation được lưu trữ và được sử dụng để cho phép users access tới applications và chức năng họ cung cấp. Bạn sẽ nhìn trong chapter này cái mà mỗi phần của tiến trình đã giải quyết các vấn đề nhưng đã phát triển cái mới như thách thức công nghệ và bảo mật. Chúng ta sẽ mô tả các cách tiếp cận cũ đã được sử dụng để manage identity infomation và cung cấp authentication and authorization. Chúng ta đã chọn các công nghệ xác định để làm nổi bật ưu điểm và khuyết điểm của mỗi cách tiếp cận cái mà có lẽ giúp bạn đánh giá solutions cho project của bạn. Chúng ta sẽ khảo luận tại sao bạn sẽ sử dụng một industry standard protocol thay vì tự phát mình solution cho chính bản thân mình.

## Identity Management Approaches

Nó quý giá để hiểu hạn chế và lợi thế của các cách tiếp cận.

### Per-Application Identity Silo

Trong thời kì đồ đá của các ứng dụng máy tính, mỗi ứng dụng thường thực thi kho chứa identity, authentication, authorization của chính nó. Một doanh nghiệp lớn thường có các application business hard core như là hệ thống kiểm soát tài chính và hàng tồn kho. Mỗi ứng dụng thường có database của chính nó hoặc lưu trữ khác với user identities, credentials, and user profile data được lưu, mỗi ứng dụng thông báo tới user log in và sau đó validated user’s credentials. Điều này có nghĩa rằng một employee có lẽ có các username, pass để nhớ cho mỗi application. Nó cũng có nghĩa rằng nếu các element của một user’profile thay đổi, chúng ta phải thay đổi hàng loạt. Dĩ nhiên điều này không xảy ra trong thực tế nếu một công ti có nhiều applications vì user data thường được đồng bộ trong hệ thống. User sẽ gặp nhiều vấn đề và cực kì khó chịu với kiểu này, khi số lượng ứng dụng trong doanh nghiệp tăng lên thì hệ thống này không thể bảo vệ cho các doanh nghiệp.

Cách tiếp cận ngớ ngẩn này vẫn được sử dụng cho đến ngày nay trong nhiều kịch bản consumer-facing nơi mà user signs up bởi việc cung cấp một username password cho ứng dụng xác định. Nếu user sử dụng lại cùng pass trên các sites khác, một thỏa hiệp trên bất kì một site cũng có thể đặt user’s data vào tình trạng rủi ro. Nếu user chỉ định một diffirent password cho mỗi application, họ có thể nhớ nhưng nhớ quá nhiều pass. Đại loại là dùng 1 mk thì không an toàn mà dùng nhiều thì khó nhớ

### Centralized User Repository

Theo thời gian, ngày càng nhiều phần mềm được viết cho rất nhiều chức năng của business. Điều này dẫn đến một nhu cầu về cách tiếp cận tốt hơn identity management. Nhiều công ty thực thi các directory services để lưu trữ và tập chung user identity infomation. Directory services được tối ưu cho thông tin cái mà đọc thường xuyên nhưng không được cỉnh sửa thường xuyên, cái này thường là trường hợp cho user identity data. Các ứng dụng có thể sử dụng directory serivce để lưu trữ user data và credentials. Nó cũng có khả năng cho một ứng dụng thông báo tới một user log in và validate các chứng chỉ đã nhập bằng việc sử dụng thông tin trong một directory service.

The centralization of identity administration and access với directory service cung cấp nhiều cải tiến. Directory nhân rộng các tính năng cho phép applications được hosted trên toàn thế giới để tận dụng cùng identity infomation, loại bỏ vấn đề không thống nhất dữ liệu. Cùng username và pass có thể được sử dụng trên nhiều applications. Một centralized directory service cũng cung cấp một điểm đơn để control cái thực tho password policy hoặc hủy bỏ identity nếu cần thiết.

Hạn chế của thằng này là không bảo trì bất cứ sort of sesssion cho một user. Tuy thằng này tập chung một chỗ thông tin nhưng nó vẫn cần nhập username pass trên mỗi application và giống hệt nhau. Rồi điều này còn rủi ro nữa khi tất cả đều tập chung trên một công ty mạng, nên công ty đó mà bị tấn công là ăn lz. Khi các công ty bắt đầu sử dụng clound applications, việc lộ directory passwords tới clound applications đã sở hữu bởi thằng khác là không thể chấp nhận được.

### Early SSO Servers

Một vài kiểu đã biết như là identity and access management (IAM) hoặc single sign-on (SSO) servers đã có cải tiến nhiều hơn. ban đầu SSO Servers tận dụng the identity infomation trong directory service, nhưng việc cung cấp một layer bên trên the directory service cái mà duy trì session để nhớ user đã xác thực. Nhưng với cách tiếp cận này, một application có thể chuyển hướng user’s browser tới một SSO Server để có the user authentucated và nhận được the authentication kết quả được bảo vệ. Nếu user đã truy cập vào application thứ 2, sau khji họ đã xác thực ở ứng dụng thứ nhất, ứng dụng thứ 2 chuyển hướng browser tới SSO server và SSO server sẽ detect session đã tồn tại và chuyển hướng lại tới application với một success status mà không cần bắt user credentials lại.

Với single sign-on servers có nhiều cải tiến trên directory services. Users có thể truy cập nhiều applications với một lần xác thực. directory password chỉ bị phơi ra trên SSO server thay vì mỗi application mà user đã truy cập.

Thật không may, có nhiều hạn chế với early SSO servers trong thực tế. Tình tương tích giữa các applications và SSo servers thường độc quyền và các sản phẩm SSO thường tốn nhiều thời gian để thực hiện. Một hạn chế đáng kể là single sign-on dựa trên cookie, với cookie thì external Software-as-a-Service applications sẽ bị hạn chế.

### Federated Identity and SAML 2.0

Sự bùng nổ của SaaS applications kiểu mới đã tạo ra thách thức cho managing identities. Trong chớp mắt, các teams business ở khắp mọi nơi chê sự lạc hậu của phòng IT và sign up cho SaaS applications với một credit card. Không may rằng, đây không phải là cách tốt đển quản lý employee identities trong SaaS applications. Nó khó cho company ddeeer theo dõi accounts của employees đã tạo trong SaaS systems, và user một lần nữa phải nhớ một password cho mọi application. The signle sign-on mà họ thích chỉ cho internal applications không mở rộng tới external SaaS applications trong các domain khác.

May mắn thay, một tiêu chuẩn công nghiệp mới, SAML 2.0 (Sercurity Assertion Markup Language) đã được phát hành trong năm 2005. Nó cung cấp một giải pháp cho web sign-on across domain và federated identity (liên kết identity). Nó là hoàn hảo cho các doanh nghiệp với SaaS applications. SAML 2.0 cung cấp giải pháp xuất xắc cho doanh nghiệp cần để control employee identities trong SaaS applications

Với SAML 2.0, SaaS applications có thể điều hướng các users cùng công ty back to corporate authentication service, được biết như là một identity provider (IdP) cho việc xác thực.

Identity federation đã cung cấp một cách để link một identity đã sử dụng trong một application với một identity tại the identity provider. Các công ty có thể cải tiến single sign-on với 2 internal và SaaS applications. Các users chỉ cần một username/password để nhớ. Doanh nghiệp có a centralized control point cho cả 2 internal và external identities và có thể shut off accesss nhanh chóng tại corporate identity provider nếu cần. Password policy và multi-factor authentication có thể được thực thi ở một nơi.

Mặc dù đã được áp dụng một cách rộng rãi, tuy nhiên, SAML 2.0 không phải là viên đạn bạc. The protocol được thiết kế để cover nhiều kịch bản. làm nó phức tạp để config và thực thi. Trong khi SAML 2.0 được sử dụng rộng rãi ở môi trường doanh nghiệp, nó không khả thi mô hình business với các kịch bản consumer-facing. Các users không thích trả tiền cho một consumer-facing identity service, với lại thằng này chỉ giải quyết vấn đề authentication còn authorization API thì không.

### WS-Fed

The Web Service Federation Language (WS-Fed) federation framework đã được tạo bởi một liên minh industry như là một phần của một tập hợp lớn của các protocols đã biết như WS-\*. The WS-Fed 1.2 được phát hành như một OASIS standard trong năm 2009 và đã cung cấp cơ chế “authorized access to resource managed in one realm can be provided to security principals whose identities are managed in other realms”. Nó được hỗ trợ bởi Microssoft’s ADFS server cũng như nhiều sản phẩm SSO thương mại khác và đã cung cấp các tính năng tương tự SAML 2.0 web single sign-on and federation. Nó dùng trong nhiều môi trường doanh nghiệp như SAML 2.0 và vẫn sử dụng trong nhiều công ty ngày nay.

### OpenID

Với SAML 2.0 chỉ được áp dụng trong employee-facing scenarios, consumer users vẫn buộc phải đăng kí một lần tại mỗi consumer-facing web site. Một nhóm đã phát triển một solution cho nó khái niệm “user-centric” identity, và thằng này nổi lên và được gọi là OpenID. Để tổ chức và controll identity providers đã sử dụng với SAML 2.0 và WS-Fed, OpenID bao gồm ý tưởng của user-controlled identity cho consumer use cas. Consumer users có thể set up identity provider cho chính họ và là điểm để các applications xác thực. Thằng này không được sử dụng rộng nhưng nó đã làm nổi bật lên user-centric identity.

### OAuth 2.0

Với Web 2.0 và sự trỗi dậy của social media, nhiều consumer-facing web sites đã được tạo ra cái mà cho phép user upload content giống như là picture. Điều này dẫn đến các trường hợp sử dụng nơi mà một application cần lấy được content đó trên danh nghĩa người dùng. Ví dụ, một person người là đã uploaded photos tới một social media site có lẽ muốn muốn một web site khác có thể access tới photos của họ tại social media site. Trong tường hợp không có giải pháp tốt hơn, user có thể share credentials social media của họ với photo printing site. Nếu photo printing site được thỏa hiệp, nó có thể đặt acount social media account vào trường hợp rủi ro. User cũng không control được rằng the photo printing site có thể làm gì khi mà nó đã có user’s password từ social media site. Một giải pháp cần thiết cái mà cho phép user cấp quyền tới một application tại một web site để trích xuất nội dung của họ từ web site’s API khác, không cần user phải phơi credentials của họ tới first site. The OAuth protocol cung cấp giải pháp cho trường hợp sử dụng này. The OAuth 2.0 cho phép user cấp quyền cho một application, được biết như là một client (the photo printing site), để gửi một request tới một API, được biết như là resource server (the social media site), đại diện cho user để trích xuất dữ liệu tại resource server được sở hữu bởi user. Để làm điều này, the application tương tác với authorization server cái mà xác thực user như là một phần của sự đồng ý của họ cho appication để truy cập vào tài nguyên của họ. the application nhận được một token cái mà cho phép nó gọi đến resource server. OAuth 2.0 giải quyết trường hợp API authorization quan trọng. Tại thời điểm này, có vài social media sites trên Internet như Google và LinkeIn thực thi OAuth 2.0 để cho phép consumer-facing applications lấy được thông tin từ một user của Google hoặc LinkedIn profile.

Social media đã cung cấp để xác thực users cho việc truy cập tới site của họ và khi đó một OAuth 2.0 authorization request nhận được cái mà đã yêu cầu đã xác thực được sự chấp thuận của user. Nếu họ cung cấp một vài authentication service, nó cõ lẽ thu hút nhiều người dùng hơn tới nền tảng của họ, và một consumer-facing authentication service có lẽ hiệu quả và sẽ được thanh toán bởi các quảng cáo cho the social media sites. Có vài vấn đều, OAuth 2.0 không được thiết kế như là authentication service chung và có thể không được an toàn cho mục đích này

### OpenID Connect (OIDC)

Thằng này được thiết kế để cung cấp một tính năng quan trọng cần cho authentication service. Thậm chí nếu OAuth 2.0 authorization servers có khả năng authenticating users, the framework không cung cấp một cách tiêu chuẩn để truyền tải an toàn the identity of an authenticated user to an application. OIDC đã cung cấp giải pháp cần thiết này. Nó đặt ra một layer phía trên của OAuth 2.0 protocol để cung cấp thông tin trong một format chuẩn tới applications về identity của một user đã được xác thực. Cái này đã cung cấp một giải pháp cho các applications cho user authentications cũng như API authorization. Sự thực thi OIDC đã được sử dụng rộng rãi trên các social media/serivce provider như Google, Paypal và Yahoo. OIDC có lợi tới các users, application developers, và identity providers. Web site developers có thể ủy thác việc thực thi authentication và password reset logic tới một OIDC provider. Users có lợi bởi vì họ có thể tận dụng một tài khoản để log-in tới nhiều sites mà không cần phơi tài khoản chứng chỉ của họ tới các sites khác. Các users có vài username và passwords sẽ thích single sign-on. Nói chung thằng này là thằng cực kì hiệu quả với single sign-on

## Standard Protocols

Các chapter tiếp theo sẽ mô tả 3 thằng được sử dung phổ biến nhất industry standard identity protocols và cách chúng làm việc. Nhưng đầu tiên, tại sao sử dụng một industry standard protocols? Đầu tiền, các protocols này đã được xem xét kỹ lưỡng bởi nhiều người, vì thế nó có ít khả năng bị tổn thương hơn cái mà ta tự phát mình ra. Thứ hai, các protocols này được sử dụng rộng rãi, cung cấp khả năng tương tác giữa application và service providers của bạn. Thứ 3, nếu bạn muốn truy cập user profile data từ các services như google, bạn sẽ sử dụng the standard protocols như là thằng thực thi của các services này. Tương tự, nếu application của bạn được sử dụng bởi các doanh nghiệp, doanh nghiệp có lẽ mong muốn application của bạn sử dụng một trong các protocols này. Thứ 4, các protocols này được thiết kế để hỗ trợ authentication signle sign-on. Cuối cùng, sử dụng protocol đã tồn tại có thế tiết kiệm thời gian vì nhiều ngôn ngữ hỗ trợ SDKs.

Nếu bạn là người mới, nó có lẽ khó học nhưng từ những thằng này ta có thể mix để tạo ra các giao thức phù hợp với chính solution của ta

# Chapter4: Identity Provisioning

Bước đầu tiên trong cuộc sống của một identity là việc tạo ra nó. Provisioning là hành động thiết lập các identities và accounts cho application của bạn. Như đã đề cập ở chapter 2, một identity có ít nhất một identifer và nhiều user profile attributes. Một online account thì được liên kết với một identity và có thể được sử dụng để truy cập tới protected resources. Mục tiêu của gia đoạn provisioning là tạo hoặc chọn một repository cho các user accounts và identity information cái mà sẽ được sử dụng trong authentication và authorization của các users.

## Provisioning Options

Cho một developer application, the identity provisioning phase bao gồm việc nhận được các users và việc tạo accounts và identity profiles cho họ. Một cách tiếp cận clear là có một users sign up cho một local application account, nhưng nó không phải là khả năng duy nhất.

* Một user tạo một identity mới bởi việc điền toàn bộ vào một self-registration form
* Một trường hợp đặc biệt của self-registration là gửi lời mời tới users được chọn để đăng kí
* User identities được chuyển từ một repository user có sẵn
* Một identity service với một repository của user identities có sẵn được tận dụng
* Một administrator hoặc automated process tạo identities

### Self-Registration

Một option bên trên là để users tạo một account mới cho application và xác định identity infomation của họ thông qua self-service sign-up. Điều này đòi hỏi phải lôi kéo users vào site của bạn và để họ điền vào registration form, sau đó lưu lại các thông tin đã thu thập được. Đây là cách tiếp cận phổ biến cho customer-facing sites và yêu cầu bạn thiết kế và tạo sign-up form. Nó cũng yêu cầu là phải thông báo đến user về việc sử dụng các thông tin của họ cho các kế hoạc về sau. Bạn nên giữa thông tin được yêu cầu ở mức tối thiểu để tránh rườm rà trong quá trình đăng kí nếu không người dùng sẽ bỏ đéo đk nữa.

Với self-registration có khả năng mở rộng, ít nhất là so với administrators accounts.

#### Ưu điểm:

* Có khả năng thu thập thông tin người dùng cái mà không có sẵn
* Kiểm soát được toàn bộ trải nghiệm người dùng về việc registration
* Có khả năng mở rộng

#### Nhược điểm:

* Có lẽ cản trở nhiều users mới tiềm năng từ việc sign up
* Chịu trách nhiệm pháp lý với việc lưu lại login credentials ( lưu lại visa là cực kì nguy hiểm)

#### Progressive Profiling

Bạn có thể giảm thông tin một user phải nhập khi sign-up băng việc sử dụng progressive profiling, thực tế việc xây dựng user profile attributes cho một identity, thay vì bắt họ điền toàn bộ trong một lần, với progressive profilling, một user được hỏi để cung cấp attributes ít nhất khi họ sign-up. Nếu user thực hiện các transaction muộn hơn họ cần hoàn thiện nhiều hơn về các thông tin. Ngoài ra việc bổ sung thông tin sau một thời gian hoặc sau một số lần đăng nhập. Về cơ bản thì chia việc thu thập thông tin thành nhiều giai đoạn, khi đăng kí tài khoản không nhất thiết cần thu thập hết mà về sau tùy vào chức năng người dùng mà cần nhiều thông tin hơn thì ta bắt họ điền thêm, đây là lý do tại sao ta gọi là progressive profiling (cải thiện profile)

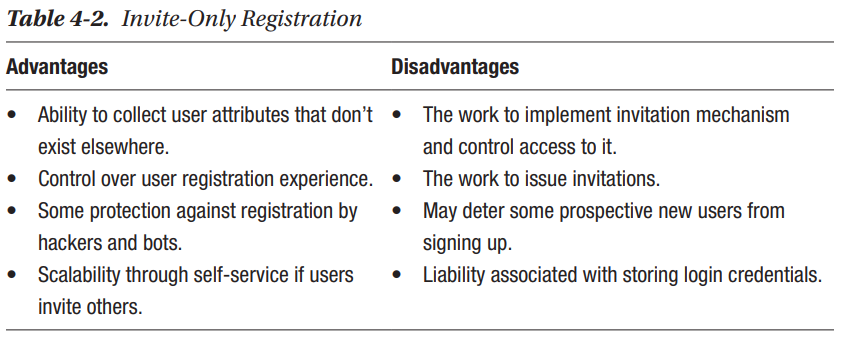
#### Invite-Only Registration

Một biến thể khác của self-service registration là the invite-only registration flow. Trong tình huống này, chỉ các users được mời để sign up. Việc mời này có lẽ được phát đọng bởi user khác. Nhiều social networking sites sử dụng cái này để tiếp cận các users invite bạn của họ để join vào site. user được mời sẽ nhận một link cái mà đưa cho họ một sign up form nơi mà họ có thể register. Việc invite cũng có thể được phát động từ một administrator của site. Trường hợp này có lẽ bao gồm một registration form cho user hoặc nếu administrator đã cung cấp tất cả account dât cần thiết, nó có lẽ chỉ bao gồm email address validation hoặc một password reset. Kỹ thuật này có lẽ hữu dụng để mời những users nhất định test giai đoạn early access version của application

Với một invite-only sign up, truy cập tới registration form thì bị hạn chế để chọn nhóm users người mà nhận được invitation. The registration page có lẽ khóa lại email address hoặc phoen number để sử dụng nó trong invitation vì nó không thể thay đổi tại thời điểm registration, nó quan trọng để loại trừ người không được invite đánh cắp invitation từ người khác và đăng ký như chính họ. The link trong invitation cũng có hể có thời hạn liên kết với nó, nếu cần thiết, mỗi invitations được theo dõi để nó chỉ có thể được sử dụng một lần.

Một invite-only flow cũng có thể được sử dụng trong các tình huống nơi mà bạn cần taonj một tài khoản để gán đặc quyền tới nó trước khi gửi lời mời. Cách tiếp cận này có thể được sử dụng để thiết lập employee accounts cho nhân viên mới hoặc custommer accounts cho giai đoạn early access application environment. Một administrator hoawhc automated process có thể tạo account, asign đặc quyền và kích hoạt việc gửi link mời tới một new user. The user click vào link và cung cấp thêm thông tin trong regisstration form nếu cần. Thông tin đã nhập của user có thể liên kết với account đã tạo từ trước. The account này sau đó sẵn sàng cho person để sử dụng và có các đặc quyền được gán từ trước tới account bởi administrator gửi tới ( tức là trường hợp này là trường hợp thêm đặc quyền).

The invite-only flow tương tự the self-registration option mô tả trước. Nó có thể bảo vệ registration chống lại hackers và bots... Một invite-only registration flow có lẽ yêu cầu làm việc bởi một administrator. nhiều ưu và nhược điểm được đề cập dưới bảng sau (thực ra phần này chưa cần tìm hiểu)



### Identity Migration

Nếu identities đã tồn tại ở nơi nào đó, chúng có thể được di chuyển từ một repository, như là legacy database tới repository khác cái mà có thể được sử dụng trong application mới. Lợi thế là các users không cần cung cấp thông tin mà đã nhập và repository mới có thể được cư chú một cách nhanh chóng từ legacy repository. Trong khi phần lớn profile attributes có thể được trích xuất và di chueeyrn thì với password là một thách thức. Passwords bình thường được lưu dưới dạng một hashed format. Hashing chuyển đổi chúng thành một string của các ký tự ngẫu nhiên và không thể reversed để nhận lại original value “cleartext”. Việc lưu trữ đươi dạng hashed format cho phép validation passwords đã nhập nhưng phòng tránh trường hợp administrators access tới password repositoryes và nhìn thấy cleartext passwords, cách này gây khó khăn về password khi kho chứa dữ liệu bị đánh cắp.

Có nhiều thuật toán cho hashing password. một Password hashed trong một hệ thống không thể imported và được sử dụng bởi hệ thống khác. Nếu 2 hệ thống sử dụng thuật toán hash khác nhau hoặc inputs khác nhau tới cùng thuật toán, nó không thể di chuyển hashed password từ một hệ thoongss tới hệ thống mới. Trong hoàn cảnh này có vài giải pháp để xem xét cho việc migrating identities tới một hệ thống mới.

#### Support Legacy Hashing Algorithm

Một giải pháp để update tới hệ thống mới là hỗ trợ thuật toán hashing được sử dụng bởi hệ thống di sản. Điều này yêu cầu việc thi thi trong hệ thống mới thuật toán hashing di sản từ hệ thống cũ. Cái này sẽ có thể di chuyển tất cả identity data và hashed passwords từ legacy system tới new system mà không cần yêu cầu user reset password

#### Bulk Identity Migration

Một giải pháp khác là trích xuất identity data của user trừ hashed passwords từ legacy system và import nó tới new system. Với thằng này bắt buộc phải reset password, và thông qua email để gửi link nên người dùng bắt buộc là đã xác nhận email

#### Gradual Migration of Users

Một cách thay thế tới bulk

### Administrative Account Creation

Chưa có giải pháp nào khác để xem xét việc tạo các accounts và identities là một administrator hoặc automated process tạo chúng. Cách tiếp cận tốt nhất trong tình huống này nên là

* Kích thước của một nhóm
* Tần suất mà các users mới được thêm vào
* Liệu provisioning cần làm trên domains

#### Manual Account Creation

#### Automated Account Creation

#### Cross-Domain Account Creation

### Leverage Existing Identity Service

Tận dụng một identity đã tồn tại cho một user trong identity provider service. Cái này cho phép users mượn một account đã có tại social provider như Facebook hoặc Google, một corporate identity provider service. Với option này, applocation của ta ủy thác trách nhiệm xác thực users tới một identity provider và nhận được một security token với thông tin về user’s authenticated session và tùy chọn, attributes về user.

Việc tận dụng accounts trong một existing identity provider service có lẽ khiến users ít phải làm việc nhất, nó giảm thông tin mà user phải nhập tới registration form. Nó cũng có nghĩa users không cần set up một password khác. Nó cũng giảm rủi ro khi mà user password không được lưu trữ dưới hạ tầng của bạn. Nếu một identity provider service chứa tất cả attributes application bạn cần về user đó, bạn có thể thu thập nhiều hơn về dữ liệu sau này.

#### Self-registered Identities

#### Govement Identities

#### Organization Identities

### Identity Provider Selection

Nếu ta tạo một customer-facing application cái mà không yêu cầu validate identity infomation, việc cho phép users xác thực thông quan một self-registerd identity cũng như thông quan social provider account thì là được đề xuất bởi tính tiện lợi của nó.

Việc dựa trên social identity provider accounts để access company applications có thể có vấn đề bởi vì the user tự sở hữu identity và account tại các providers này. Khi đó nếu thằng này rời khỏi công ty ta không thể nào khóa tài khoản của nó lại được. Mặt khác, một social provider account được linked tới một local application account để cho phép đăng nhập vào application thông qua social provider identity, the link có thể bị xóa và local account disable nếu nhân viên đó rời đi. Cho employee-facing applications, thì đây là cách tốt nhất để sử dụng identity serivce nơi mà tổ chức sử dụng lao động sở hữu các accounts.

Một organization-controlled identity service cung cấp một nơi duy nhất cái mà tổ chức có thể provision accounts cũng như shut off accounts nếu một nhân viên hoặc member rời khởi tổ chức. Nó cũng trao một điểm là nơi để enforce credential strength/policy và deploy multi-factor authentication cũng như log authentication activity. Có vài nhà cung cấp cloud cái mà đề xuất một identity service trên một subcription basic như Google Apps, Azure AD, Auth0, Amazon Cognito và Okta

### Choosing and Validating Identity Attributes

Topic về provisioning luôn đặt ra câu hỏi gây tranh cãi là làm thế nào để identity một user. Email được áp dụng rộng rãi là identifers, nhưng không phải là không có vấn đề. Việc sử dụng một email là một identifer có lợi thế, tuy nhiên các user có thể cần thay đổi email của họ vì một vài lí do và vẫn giữ lại khả năng truy cập vào account. Thêm nữa, một email provider có thể chỉ định một email đã sử dụng trước đó tới một chủ sở hữu user mới. Cho business-facing applications, nhiều business không cung cấp employees của họ với email accounts cái mà có thể gặp vấn đề nếu một application giả định là có sẵn một email address. Tương tự, applications được đánh mác cho trẻ em thì trẻ em không thể nào mà có một email được.

Việc sử dụng một user-selected username cũng có ưu và nhược điểm. một username sẽ dễ dàng hơn để ta set multiple accounts nếu cần và username thường ngắn hơn là email. Một user phải chọn một unique username, nó có lẽ khó để nhớ username trên mỗi site.

#### Suggestions

Nhiều nhược điểm đã liệt kê trước đó xuất phát từ việc ta dùng một attribute cho nhiều mục đích. Nên để tránh chúng thì việc tách rời và sử dụng các attribute khác nhau cho mỗi mục đích là cần thiết

* Identifer cho việc log in
* Display name
* Notification/ communication/ account recovery
* Thực thi Internal account
* Liên kết một identity/account với hồ sơ ứng dụng
* Capturing user activity trong log files
* identifer liên tục một user theo thời gian cho mục đích kiểm toán

Thằng thứ 3 trong list trên được sử dụng cho việc thực thi internal account và nên sử dụng một unique, internal account identifier cái mà không bị tác động bởi một user khi họ thay đổi profile attribute như email address, phone number, hoặc tên cũ của họ. Thêm nữa, các đề xuất sau để tránh hạn chế:

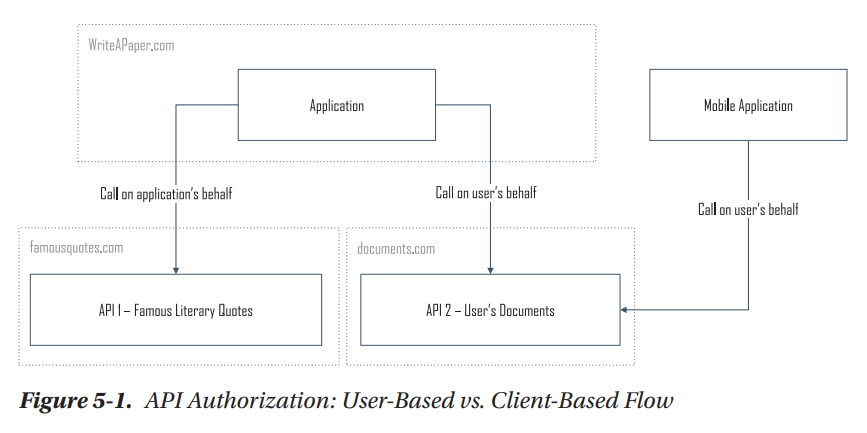
* Tránh phơi identifers cái mà chứa personal data
* Sử dụng một internal account identifer trong log files để tránh phơi personal data logs một cách trực tiếp
* Sử dụng một internal account identifier trong hồ sơ của application
* Cho phép các users xác định tên hiển thị để sử dụng trên screens/printouts để bảo vệ riêng tư
* Identifiers/attributes cho logging in, display, and notification nên khác biệt và có thể thay đổi
* Cho phép thiết lập nhiều attributes cho mục đích thông báo như là một primary and secondary email
* Cho phép một username dài với kí tự đặc biệt và có thể thay đổi bởi vì các users sẽ có thể linh hoạt. Một attribute profile riêng bên cạnh username nên được sử dụng cho thông tin để thông báo và liên hệ

#### Validating Critical Attributes

# Chapter 5: OAuth 2.0 và API Authorization

## API Authorization

Một applciation có lẽ cần gọi một API trên danh nghĩa một user, để truy cập tới nội dung mà user đó sở hữu hoặc đại diện cho chính application nếu application sở hữu nội dung đó.



Trong trường hợp này, WriteAPaper.com là application giúp users viết và chỉnh sửa tài liệu nghiên cứu. Nó gọi tới 2 APIs, 2 thằng này được sở hữu bởi các nhóm khác nhau. Đầu tiên là famousquotes.com cái này cung cấp validated các trích dẫn trong tài liệu. API thứ 2 là documents.com và cung cấp một document storage service. Có một thằng application thứ 2 là mobile, nó gọi đến documents.com API để truy cập tới documents.

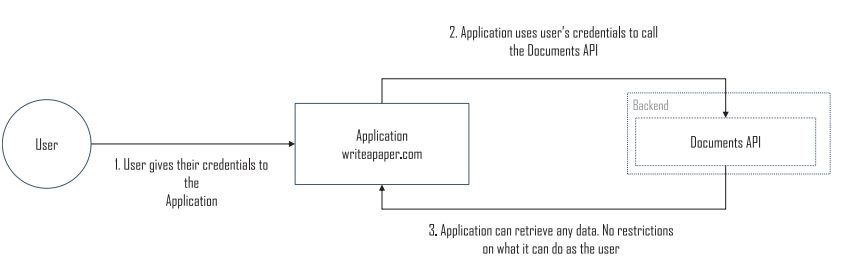
Khi mà Write application gọi đến API famous, nó đại diện cho chính bản thân thằng application. Các nội dung trích dẫn không được sở hữu bởi user, vì vậy nội dung của user không cần thiết phải truy cập. The application chỉ cần là một client đã được đăng ký và đã được cấp quyền là có thể gọi tới quotes API. Khi application gọi tới API documents thì ở đây là chứa documents của users, vì vậy việc request này phải dựa trên danh nghĩa của user. Và application phải được user chấp thuận mới có thể truy cập tới dữ liệu của user đó. The mobile application cung cấp read-only access tới user’s document và không access tới quote service. Nó yêu cầu việc cấp quyền từ user để gọi tới documents API để lấy được documents.

## OAuth 2.0

OAuth 2.0 được thiết kế để cho phép một application đạt được authorization để gọi APIs. Với OAuth 2.0 một application có thể lấy được sự đồng ý của user, và gọi API trên danh nghĩa user đó mà không cần phải đưa cho API credentials.

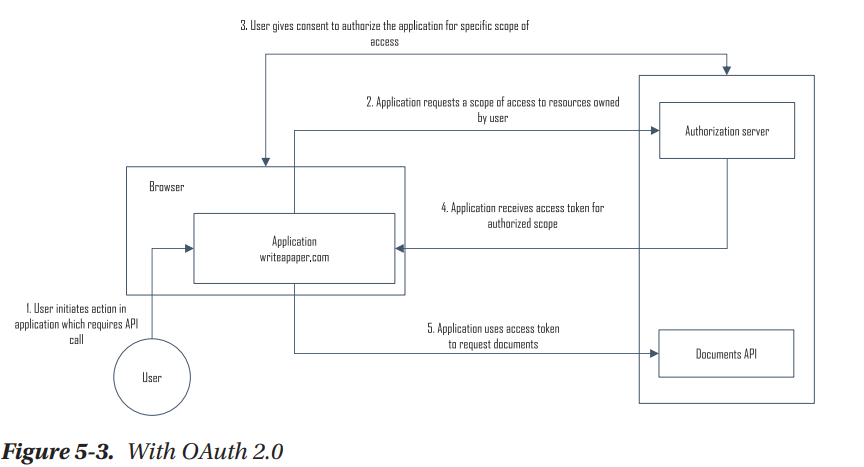
Trường hợp sử dụng nguyên thủy bao gồm một user hay còn được gọi là một resource owner, người đó muốn cho phép một application truy cập tới một protected resource được sở hữu bởi resource owner đó tại một site riêng biệt được gọi là resource server.

Trước kia khi chưa có OAuth 2.0



OAuth 2.0 được thiết kế để cung cấp giải pháp tốt hơn so với sơ đồ trên. Nó cho phép một user cấp quyền một cách rõ ràng tới application để gọi tới API trên danh nghĩa là chính user đó chứ không phải giao tất cả thông tin cho application rồi muốn làm gì gì làm. Với OAuth 2.0, khi một application cần gọi API trên danh nghĩa user, nó gửi một yêu cầu authorization tới một authorization server. Một authorization server xử lý các yêu cầu để có thể truy cập tới API, sau khi xử lý nó trả về một sercurity token cái mà có thể được sử dụng để access tới API. Trong một authorization request, the application cho vào request đó một sự chỉ định (được biết với cái tên là “scope”) của cái mà nó muốn yêu cầu từ API. The authorization serrver đánh giá yêu cầu, nếu được cấp quyền, nó trả về token cho application.

Nếu application hỏi về content được sở hữu bởi user, API’s authorization server xác minh user và sau đó hỏi user để trao sự đồng ý của họ tới application (authorization hỏi user là như thế nào ?). Bước xác thực đảm bảo rằng user cung cấp sự đồng ý là chủ sở hữu của tài nguyên đang được truy cập. Nếu user đồng ý, application nhận một token để gọi tới API trên danh nghĩa user. The token được gọi là access token, và nó cho phép application tạo yêu cầu tới API bên trong scope của cái mà user đã cấp quyền. Giải pháp này loại bỏ việc user phải để cho application giữ credentials nó chia sẽ để request về sau, và cung cấp cho user khả năng kiểm soát nhiều hơn đối với những gì application có thể truy cập.



## Terminology

### Roles

Có 4 roles trong OAuth 2.0

* Resource Server
* Resource Owner
* Client
* Authorization Server

### Confidential vs. Public Clients

* Confidential Client – Một application chạy trên một server được bảo vệ và có thể lưu trữ các bí mật nội bộ để xác thực chính bản thân nó tới authorization server
* Public client – Một application cái mà chạy trên thiết bị của user hoặc browser và không thể lưu trữ bí mật một cách an toàn.

### Client Profiles

OAuth 2.0 định nghĩa 3 profiles của client

* Web Application – Một Confidential client với code được chạy trên protected, bacn-end server. The server có thể lưu trữ bất cứ các bí mật nào một cách an toàn cho client xác thực bản thân nó cũng như bât kì tokens nó nhận từ authorization server
* User Agent-Based App – Giả định tới một public client với code chạy trên browser
* Native Application – Giả định tới public client cái mà được cài và được chạy trên thiết bị của user như moblie application hoặc desktop application

Trong thực tế, có vài định nghĩa có lẽ đã bị chồng chéo nhau bởi vì một web application có lẽ phục vụ HTML pages cái mà chứa nhiều JavaScript và single-page application . Chúng ta sẽ thảo luận vấn đề này nhiều hơn trong Chapter 6 về OIDC Hybrid flow

### Token và Authorization Code

* Authorization Code – một người trung gian, opaque code (mã mờ) trả vè tới application và được sử dụng để đạt được access token và tùy chọn một refresh token. Mỗi authorization code được sử dụng một lần
* Access Token
* Refresh Token

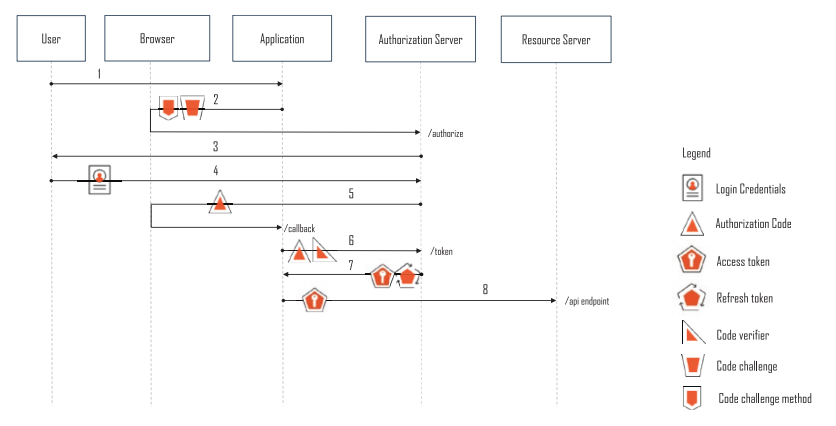
## How It Works

OAuth 2.0 định nghĩa 4 cách thức mà applications đạt được authorization để gọi API. Mỗi cách sử dụng kiểu credential khác nhau (credential cũng có nhiều loại) để đại diện authorization. Các credentials được biết như là

* Authorization code grant
* Implicit grant
* Resource owner password credentials grant
* Client credential grant

### Authorization code grant

sử dụng 2 request từ application tới authorization server để lấy access token ( tức là request 2 lần ms dc). Trong lần request đầu, the user’s browser chuyển hướng luôn tới authorization endpoint tại authorization server với một request để cấp quyền gọi tới API trên danh nghĩa user. The browser chuyển hướng tới authorization server để tương tác với user để xác thực họ và đạt được consent cho authorization request. Sau khi đạt được user’s consent, the authorization server chuyển hướng lại application với authorization code. The application sử dụng authorization code để gửi request thứ 2 tới authorization server’token endpoint để lấy access token. The authorization server phản hồi với một access token được phát tới application cái mà có thể được sử dụng để gọi API



The authorization code grant type được tới ưu cho confidential clients. Vậy vì sao lại phù hợp với thằng này, vì bản thân thằng này giữ các thông tin bí mật về chính bản thân của nó để có thể xác thực với authorization server, còn nếu client bình thường thì lộ sạch sẽ với các biến có thể xem được trên browser.

### Authorization Code Grant Type + PKCE

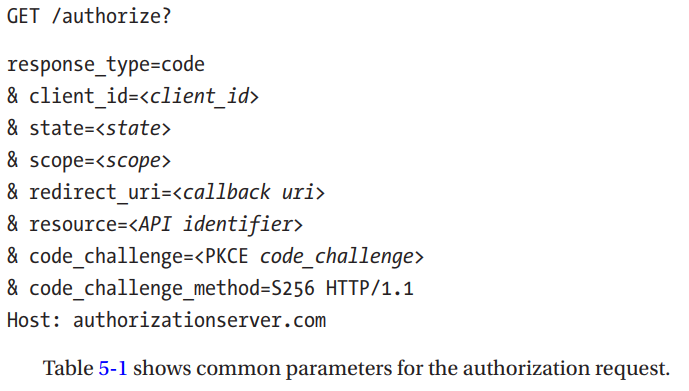
PKCE – Proof Key for Code Exchange là một cơ chế cái mà có thể được sử dụng với authorization và token request để đảm bảo rằng application cái mà request authorization code giống application sử dụng authorization code để lấy được access token. PKCE chống lại mã độc, đặc biệt trên thiết bị mobile và public clients cái mà có thể đánh chặn authorization code và sử dụng nó để nhận một access token.

Để sử dụng PKCE, the application tạo một mật mã random string được gọi là code verifier. The application sau đó tính toán một giá trị dẫn xuất được gọi là code chanllenge. Khi application gửi một authorization request trong bước 2 của diagram nó bao gồm code challenge, cùng với cách thức được sử dụng để dẫn xuất nó.

Khi application gửi authorization code tới token endpoint để nhận token trong bước 6, nó bao gồm code verifier. The authorization server chuyển đổi giá trị code verifier kiểm tra xem nó có khớp với code challenge đã gửi không. Cái này cho phép authorization server dect application độc hại đang cố để đánh cắp authorization code. Chỉ có application chính chủ mới biết code verifier để pass phần kiểm tra

#### The Authorization Request

Dưới đây là ví dụ application’s API authorization request với PKCE.

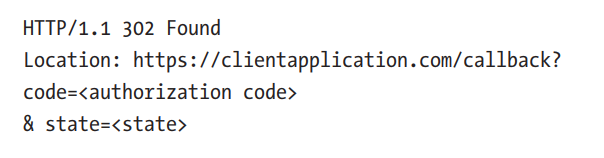


|  |  |
| --- | --- |
| Parameter | Meaning |
| response\_type | Chỉ ra OAuth 2.0 grant type |
| client\_id | Identifier cho application, được gán khi nó đã đăng kí với authorization server |
| state |  |
| scope | Chỉ ra scope của các đặc quyền được truy cập cho authorization đã yêu cầu |
| redirect\_uri |  |
| resource | Identifier của API xác định đã đăng kí tại authorization server cho token được yêu cầu access. Nhiều implementation sử dụng tên là “audience” |
| code\_challenge | PKCE code challenge dẫn xuất từ PKCE verify code sử dụng code challenge method |
| code\_challenge\_method | “S256” hoặc “plain” |

The resource parameter thì không phải là original OAuth 2.0. Tính từ thời điểm authorization servers được viết để xử lý nhiều request cho multiple APIs và các trường hợp, thì nó support thêm parameter để chỉ định API cho việc request. paramter này thường được gọi là “resource” hay “audience”

#### Response

The authorization server gửi phải hồi giống như ví dụ bên dưới tới application’s callback



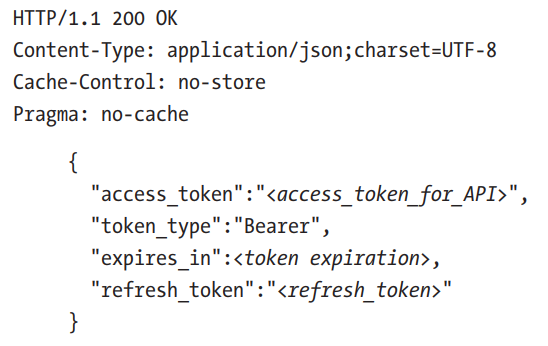
#### Calling the Token Endpoint

Sau khi nhận được authorization code, the application sử dụng nó để request lần thứ hai tới authorization server’s token endpoint để lấy được access token

POST /token HTTP/1.1  
Host: authorizationserver.com  
Content-Type: application/x-www-form-urlencoded

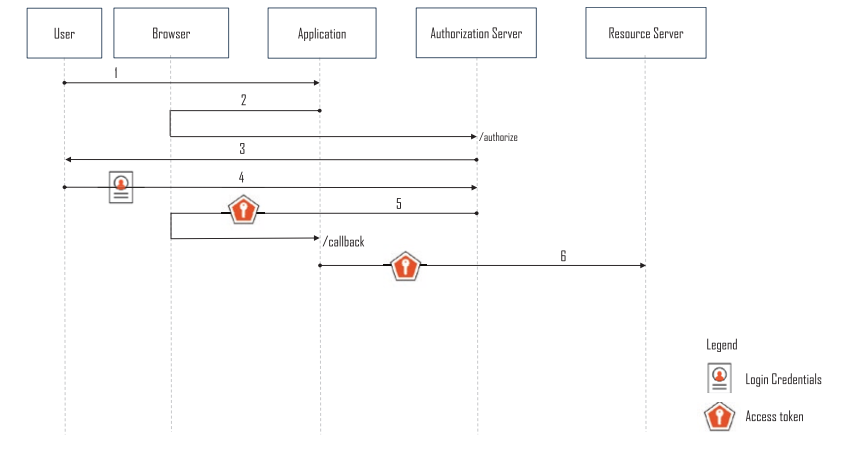
grant\_type=authorization\_code  
& code=<authorization\_code>  
& client\_id=<client id>  
& code\_verifier=<code verifier>  
& redirect\_uri=<callback URI>

Phản hồi từ token endpoint sẽ tương tự như sau



### Implicit Grant

OAuth 2.0 định nghĩa một implicit grant type, thằng này được tối ưu cho việc sử dụng public clients như là single-page applications. Sử dụng grant type này thì trả về một access token tới application trong một request. Nó được thiết kế tại thời điểm khi CORS (Cross-Orign Resource Sharing) standard không được hỗ trợ rộng rãi trên browser vid web pages chỉ có thể “phone home”. Nói cách khác, chúng chỉ có thể tạo cuộc gọi tới domain từ page đã được loaded, điều này có nghĩa rằng chúng không thể gọi một authorization server’s token endpoint. Để giải quyết hạn chế này, the implicit grant type có authorization server phản hồi tới authorization request bởi việc trả về tokens tới application trong một chuyển hướng với URL hash fragment



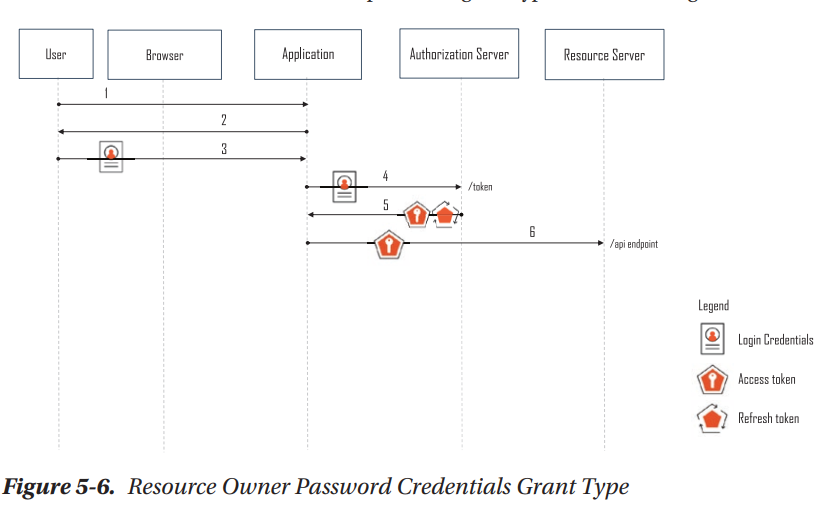
Khi mà OAuth 2.0 được phát hành, CORS được suport ở hầu hết browsers. Do đó implicit grant type không cần thiết nữa so với mục đích ban đầu của nó. Xa hơn việc trả về một access token trong một URL has fragment phơi ra access token thông quan browser history hoặc refer headers. The implicit grant type với access token được trả về trong một URL hash fragment thì không còn được đề xuất cho single-page applications cần để access token. PKCE sẽ được sử dụng thay thế

### Resource Owner Password Credentials Grant

The resource owner password credentials grant type hỗ trợ trong trường hợp một application được tin tưởng để xử lý credentials của end-user và không có grant type khác không khả thi. Với grant type này, application thu thập user’s credential trực tiếp thay vì chuyển hướng tới authorization server. The application truyền credntials thu thập được tới authorization server để validation như 1 phần request của nó để nhận được một access token.

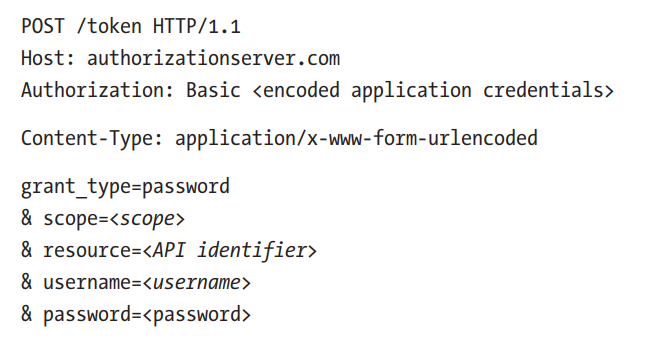
Thằng này thì không được khuyến khích dùng vì nó phơi ra credentials của user tới application. Loại này không có bước user consent, vì một application có thể request bất cứ truy cập nào nó muốn bằng việc sử dụng user’s credentials. The user không có cách để ngăn chặn việc lạm dụng credentials của họ

Thằng này được để xuất trong việc sử dụng cho user migration. Nếu users cần được di rời từ một identity repository tới thằng khác với password hash không tương thcihs. hệ thống có thể thông báo user về credentials của họ, sử dụng resuorce owner password grant để validate chúng với hệ thống cũ, và nếu hợp lệ, ta nhận user profile từ hệ thống cũ và lưu nó và credentials trong hệ thống mới. Điều này có thể tránh việc phải reset password với quy mô hớn khi migration. Nếu loại cấp này được sử dụng, khách hàng sẽ vứt bỏ thông tin đăng nhập của người dùng ngay khi nhận được access token để giảm khả năng thông tin bị xâm nhập



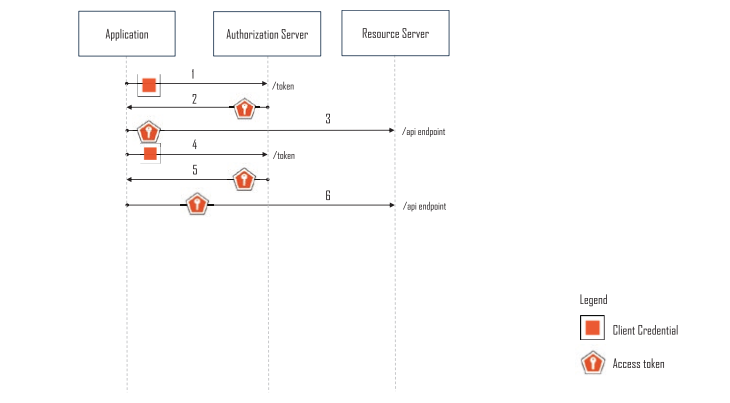
Kiểu này cũng được sử dụng trong quá khứ với mobile application gọi first-party APIs. Điều này thường được thực hiện vì login flows cái mà chuyển hướng thông qua browsers trên mobile devices thì rất cồng kềnh. Cái này được cải thiện, và RFC 8252, OAuth 2.0 cho Native Apps, bây giờ đề xuất sử dụng authorization code grant combined với PKCE cho nattive app sử dụng system browser

#### The Authorization Request

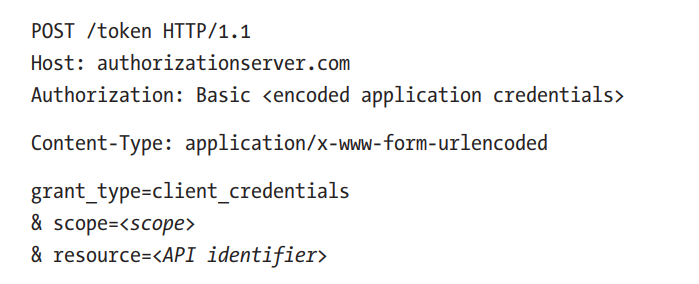


### Client Credentials Grant

Kiểu này được sử dụng khi một application gọi một API để truy cập resources mà application sở hữu. Ví dụ như hình dưới với việc gọi tới quote servier. Một quote thì không được bởi user cá nhân người mà cần quote, vì vậy việc gọi có thể dựa trên danh nghĩa application. The application sử dụng client credentials grant type và authenticates tới authorization server với credentials của chính bản thân nó để lấy được access token. Trình tự như diagram



#### The Authorization Request



#### Calling an API



## Refresh Token

OAuth 2.0 access tokens có hạn sử dụng. Khi một access token hết hạn, application có thể tạo một request authorization mới, nhưng OAuth 2.0 đã định nghĩa một cách tiếp cận thay thế với web app truyền thống và native clients bằng việc thêm refresh token. Một refresh token có thể đạt được từ authorization server và được sử dụng để đạt được một access token mới khi nó hêt hạn. Một refresh token có thể được sử dụng để kích hoạt truy cập API một cách liên tực từ native mobile application, ví dụ

Refresh token không được sử dụng trong tất cả các kịch bản. Không cần thiết một refresh token với client credentials grant bởi vì một application có thể request một access token tại bất cứ lúc nào mà không cần user tương tác. Static refresh token không được sử dụng với public clients bởi vì chúng là token nhạy cảm và public client không đủ khả năng bảo vệ chúng (chắc là chứa password, id mã hóa). The Oauth 2.0 Threat Model và Security Considerations document đề khai niệm xoay vòng refresh token để phát hiện nếu một refresh token bị ăn cắp và được sử dụng bởi 2 hoặc nhiều hơn các clients. This scheme có authorization server trả về một new refresh token với mỗi access token yêu cầu gia hạn. The OAuth 2.0 Security Best Curent Practice document chỉ định rằng authorization servers phải sử dụng refresh token rotation hoặc sender-constrained refresh tokens với public clients để giảm rủi ro

Refresh token cung cấp một cách thuận tiện cho web app truyền thống và native application để lấy được access token mới.

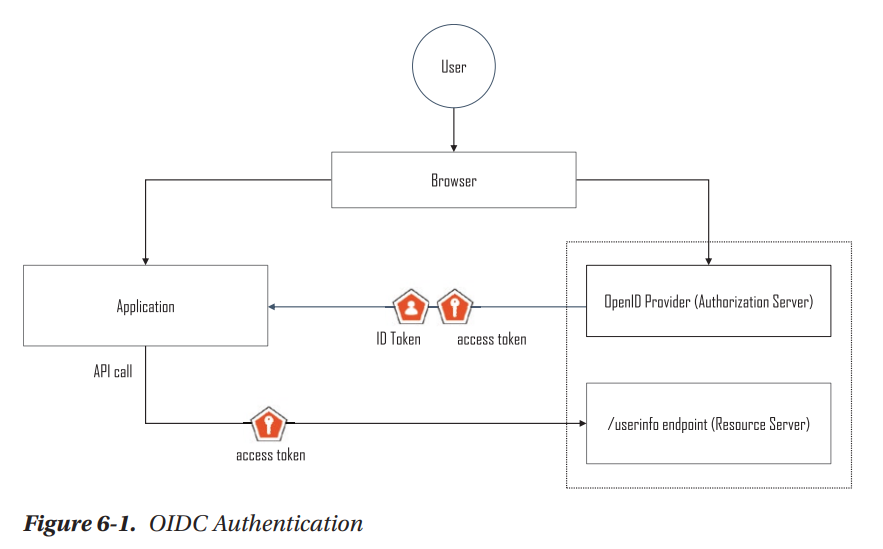
The OAuth 2.0 không bao gồm cơ chế cho application để request refresh tokens

# Chapter 6: OpenID Connect

Như đã được mô tả trong chapter trước, OAuth 2.0 cung cấp framework cho authorizing application để gọi APIs, nhưng mà không được thiết kế cho việc authenticating users tới applications. The OpenID Connect protocol cung cấp identity service layer bên trên OAuth 2.0, được thiết kế để cho phép authorization servers to authenticate users cho applications và trả về kết quả với cách tiêu chuaant.

## Problem to Solve

Kịch bản mà OIDC được thể kế là để giải quyết các thứ liên quan đến một user người mà cần được xác thực để truy cập vào application. OIDC cho phép application ủy thác user authentication tới một OAuth 2.0 authorization server và trả về tới application claims về user đã được xác thực và authentication event trong một standard format



Khi một user truy cập tới một application, nó chuyển hướng user’s browser tới một authorization server cái mà thực thi OIDC. OIDC gọi một authorization server là một OpenID Provider vì chúng ta sẽ sử dụng khái niệm này trong chapter này. The OpenID Provider tương tác với user để xác thực họ (giả định rằng họ chưa logged in). Sau khi xác thực, the user’s browser chuyển hướng lại tới application. The application có thể yêu cầu lên API với security token được gọi là ID Token. hoặc nó có thể yêu cầu một OAuth 2.0 access token và sử dụng nó để gọi tới OpenID Provider’s UserInfo endpoint để lấy được claims. Bởi vì OIDC là một layer trên top của OAuth 2.0, một application có thể sử dụng một OpenID Provider cho cả 2 việc là authentiaction và authorization tới để gọi OpenID Provider’s API.

## Terminology

### Roles

* End User
* OpenID Provider – The OpenID Provider là một OAuth 2.0 authorization server cái mà thực thi OIDC và có thể xác thực user và trả về claims về user đã xác thực.
* Relying Party – Một OAuth 2.0 client cái mà ủy thác user authentication tới OpenID Provider và request claims về user từ OpenID Provider. CHúng ta thường sử dụng khái nhiệm application

### Client Types

### Token and Authorization Code

OIDC sử dụng authorzation code, access token, and refresh token được mô tả trong chapter trước và định nghĩa một ID Token

* ID Token – một token được sử dụng để chuyên chở một authentication event và một authenticated user tới application

### Endpoints

OIDC tận dụng the authorization and token endpoints được mô tả trong chapter trước cho OAuth 2.0 và thêm nữa là UserInfo endpoint

### ID Token

Một ID Token là một token được bảo vệ sử dụng bởi OpenID Provider để chuyên chở claims tới application về authentication event và authenticated user. ID Tokens được mã hóa trong JSON Web Token (JWT)



JWT format được thiết kế để chuyên chở claim giữa 2 bên. Một JWT chứa header, payload và signature. The header chứa thông tin về type của JWT và chỉ định thuật toán được sử dụng để bảo vệ claims bên trong payload. The payload chứa the claims về một user và authentication event. The signature chứa một chữ ký dạng số dựa trên payload section của ID Token và một secret key được biết bởi OpenID Provider

## How It Work

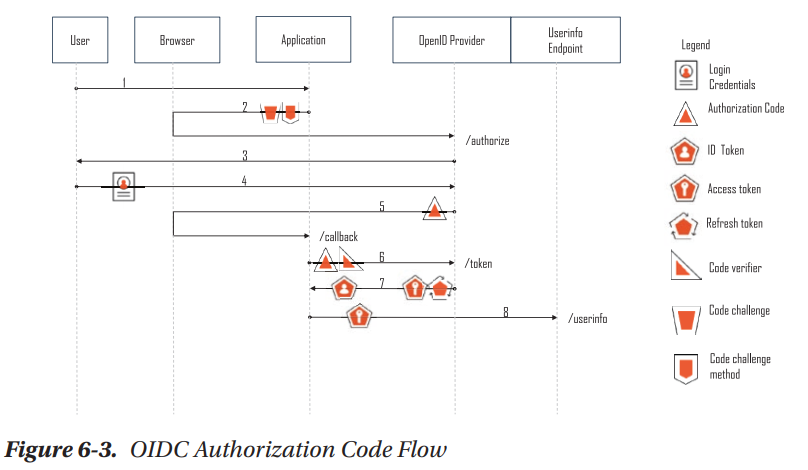
OIDC định nghĩa 3 flow khác nhau bởi một application tương tác với OpenID Provider để tạo một authentication request

### OIDC Flows

* Authorization Code Flow
* Implicit Flow
* Hybrid Flow

### OIDC Authorization Code Flow

Tương tự OAuth 2.0 authorization code grant. Để xác thực một user, một application chuyển hướng tới Provider. Provider xác thực user và chuyển hướng lại application với authorization code. The application sử dụng authorization code để lấy ID Token và access token có thể có refresh token



#### Authentication Request

Mootj application chuyển hướng tới the user’s browser với một authentication request tới OpenID Provider’s authorization endpoint như là:

GET /authorize?  
response\_type=code  
& client\_id=<*client\_id*>  
& state=<*state\_value*>  
& nonce=<*nonce\_value*>  
& scope=<*scope*>  
& redirect\_uri=<*callback\_url*>  
& code\_challenge=<*code\_challenge*>

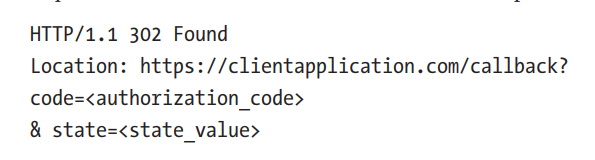
& code\_challenge\_method=<*code\_challenge\_method*> HTTP/1.1  
Host: authorizationserver.com

The response\_type parameter trong authentication reuqest được sử dụng để chỉ định OIDC flows. The response\_mode parameter tùy chọn, the method cái mà the response parameter được trả về tới application. Trừ phi không chú thích, ví dụ trong chapter này không bao gồm response\_mode thì trường hợp này mặc định response mode cho mỗi flow.

The scope trong OAuth 2.0 được sử dụng để request API đặc quyền được trao thông qua một access token. Với OIDC authentication requests, the scope được sử dụng để chỉ ra việc sử dụng của OIDC và request các thành phần claims về authenticated user. OIDC authentication requests phải bao gồm the “openid” scope value. Thêm “profile” tới cope value yêu cầu một tập các default của claims như name, family name, and give name. Thêm “email” yêu cầu email của user. Khi response\_type bên trong một đợt phát access token, the scope áp dụng tới claims được trả về bởi OpenID Provider’s UserInfo endpoint. Nếu access token không được phát, các yêu cầu claims sẽ được bao gồm trong ID Token.

Một optional parameter khác được gọi là “nonce”, nó quan trọng. The nonce value sẽ được included ID Token đã được yêu cầu. Khi một application tạo một application request tới OpenID Provider, nó sẽ chỉ định một unique, giá trị nonce không đoán được cái này được cột tới session application đã khởi động cho user. Khi application nhận được ID Token, nó phải kiểm tra cái mà token chứa có chính xác là nonce value đã được chỉ định trong authentication request không và nonce phải khớp với the hash của giá trị đã chứa trước đó trong session

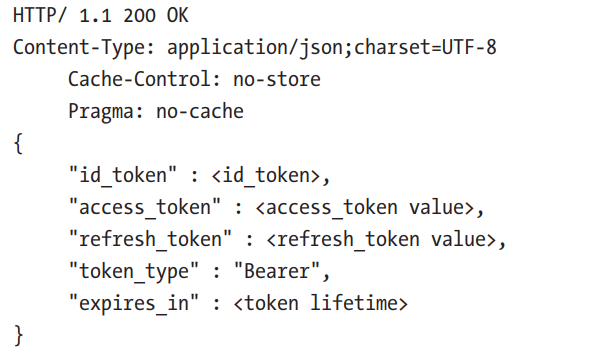
#### Authentication Response



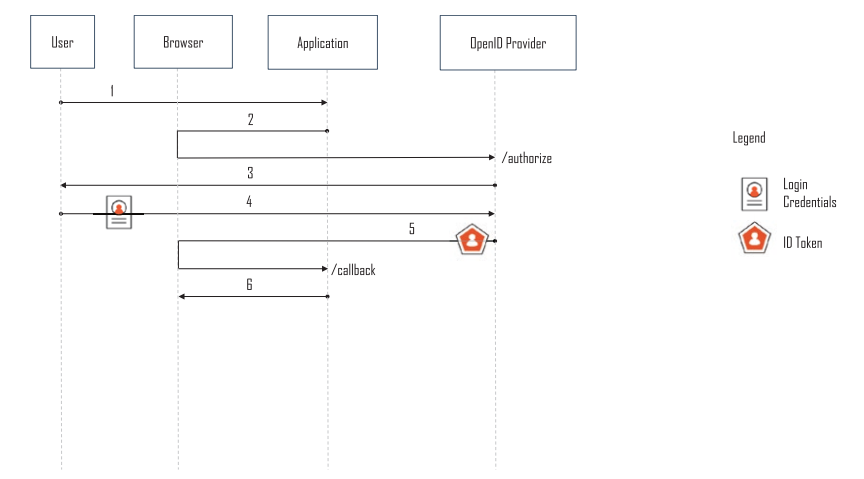
#### Token Request



Response



### OIDC Omplicit Flow



### OIDC Hybrid Flow

Hỗn hợp của Authorization Code Flow và Implicit Flow

# Chapter 8: Auhtorization and Policy Enforcement

## Authorization vs. Policy Enforcement

Trong thẩm quyền cái mà user hoặc application có thể làm, có 2 chức năng riêng biệt. CHúng ta sử dụng khái niệm auhtorization cho việc cấp các đặc quyền. Ngược lại, sự bắt buộc về chính sách truy cập được định nghĩa như hành động kiểm tra một ai đó đã được cấp đặc quyền cần thiết trước khi phản hồi tới request cho protected resource. Ví dụ, nếu bạn mua một vé xem phim, tấm vé đó cấu thành quyền của bạn để tham dự buổi chiếu phim. Trong đêm chiếu, người kiểm vé tại cửa thực thi policy kiểm tra để chắc chắn rằng chỉ có khách hàng đã được cấp quyền mới có thể đi vào.

Authorization có thể được cấp trước khi resource được request hoặc trong lúc request access. Nó có thể được thực hiển bởi entity đang chứa tài nguyên được yêu cầu hoặc bởi bên thứ 3 được tin tưởng với authorization information đáng tin cậy để thực thi chính sách.

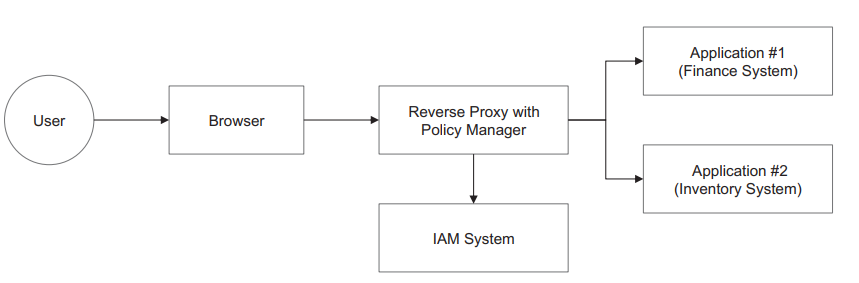
## Levels of Authorization and Access Policy Enforcement

Có các level khác nhau của authorization và access policy enforcement có thể được áp dụng

* Level 1 – Liệu một entity có thể access tới một application hoặc API hay không
* Level 2 – Những chức năng mà một entity có thể sử dụng trong application hoặc API
* Level 3 – Dữ liệu mà một entity có thể truy cập hoặc hoạt động bên trên

### Level 1 – Application or API Access

Tại level cao nhất, authorization và access policy enforcement có thể kiểm soát liệu một entity có sự cho phép để truy cập tới application hoặc API hay chưa. Trường hợp này thược được tìm thấy trong coporate setting. Ví dụ, Một employee trong một team maketing có lẽ chưa có business accssing the corporate accounting system. Level này có lẽ được xử lý bên trong một application hoặc bơi một entity đằng trước application đã được show hình dưới. The enforcement có thể là một external tới một application bởi các components như là một authentication broker hoặc reverse proxy cái mà làm việc với identity và access management system. Các hệ thống có thực hiện một high-level enforcement để điều hướng các user người mà chưa được cấp quyền để access tới một application. Một cách tiếp cận tương tự có thể được sử dụng với APIs và trong cả 2 trường hợp nó hiệu quả để giảm policy enforcement workload trên target system



### Level 2 – Functional Access

Đây là cái mà một entity có thể làm bên trong một application hoặc API. Ví dụ, một nhân viên kế toán mới trong phong tài chính có lẽ có thể truy cập tới coporate accounting system. Nó có lẽ tận dụng thông tin về user được lưu ở đâu đó như là roles hoặc group trong một directory service, nhưng thường bị bắt thực thi bên trong một application hoặc API

### Level 3 – Data Access

Trường hợp này dùng để truy cập cụ thể tới tập con của dữ liệu. Nếu functional định nghĩa các chức năng một entity có thể làm, data-level access xa hơn hạn chế truy cập đến dữ liệu xác định. Ví dụ, một ứng dụng nhập đơn đặt hàng, một user với role “sales manager” có thể được cấp quyền tại funtional level để nhìn thấy sales orders, nhưng data-level hạn chế họ để một region xác định đã chỉ định trong profile của họ “sales region” attribute (tức là chỉ xem được trong khu vực được phân thôi)

## User vs. Application Authorization

Chúng ta sẽ cover 2 tình huống yêu cầu authorization và access policy. Đầu tiên là một user (hoặc entity) có thể làm trong một application, và thứ 2 controls một application có thể request một API

Một user cần thực hiện một vài tính năng trong application, và tính năng này không thể display nếu một user không sử dụng

Một application truy cập tới API được bảo vệ, nên cần được cấp quyền

Có 3 bước tiêu chuẩn trong controlling access:

* Authorization và chỉ định access policy
* Vận chuyển thông tin authorization tới điểm thực thi bắt buộc
* Bắt buộc access policy bởi bởi enforcement point

### User Authorization

Các đặc điểm kỹ thuật của authorization là một chủ đề phức tạp. Chúng ta giải định rằng access policy được thể hiện trong một tập các attributes liên kết với user và mô tả cái cách mà OIDC có thể được sử dụng để chuyển authorization info về application và cách mà OAuth 2.0 có thể được sử dụng để authorize access tới API và cách thức thằng này support access policy. Các attributes được sử dụng để truyền đạt authorization cho users có thể khác nhua nhưng phần lớn rơi vào trong 2 trường hợp

#### User Profile Attributes

Một user’s identity có lẽ được cấp quyền dựa trên roles. Các thuộc tính này là các yếu tố tĩnh cái mà giữ nguyên bất kể người dùng ở đâu hoặc sử dụng thiết bị gì khi truy cập tài nguyên.

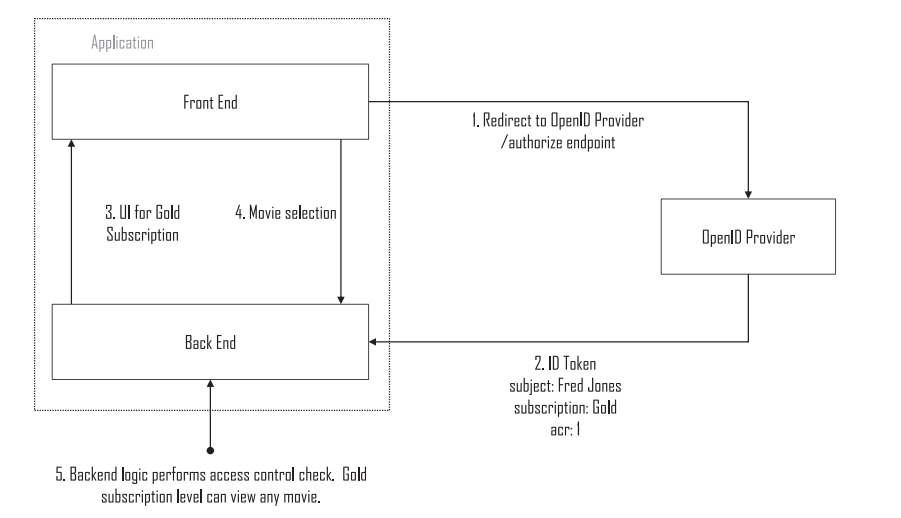
Bước cấp đặc quyền được làm trong lúc tạo user từ application. Một ứng dụng customer-facing, một user có thể được gán quyền liên qua tới profile khi họ mua subcription level

#### Transactional User Attributes

Việc cấp quyền cũng có thể dựa trên các thành phần vật lý môi trường tại thời điểm xác thực và access tới tài nguyên, như vị trí, ngày trong tuần,...Các yếu tố này được captured tại thời gian xác thực xa hơn thì đang là một phần của profile. Nếu được cpatured bởi identity provider, cú có thể được cung cấp tới application trong form của claims

### Delevery

application sử dụng OIDC, user authorization info có thể được vận chuyển toái application như là claims trong ID token.



### Enforcement

Trước khi dựa trên bất cứ một thông tin nào đó trong security token, the application phải validate nó. Các bước validate bao gồm

* Validate ID Token có đúng formatted JWT không
* Validate signature trên ID Token
* Kiểm tra token đã hết hạn chưa
* Kiểm tra issuer đã đúng OpenID Provider chưa
* Kiểm tra audience cho token là application

Khi mà ID Token đã validated, application có thể sử dụng claims bên trong token để access policy

## Application Authorization

Trường hợp đầu là User Authorization trường hợp thứ 2 là application authorization trong việc gọi APIs

### Application Attributes

Application request tới APIs trên danh nghĩa user có thể được cấp quyền bởi user, nhưng request API trên danh nghĩa chính application được cấp bởi một authorizxation server dựa trên config policy. Policy này được thiết lập một application trong việc gọi một API. Trư khi các application hoặc APIs khác lớn, the policy specifiation chỉ ra các application xác định được cấp để gọi tới APIs mà nó có thể được truy cập. Nếu OAuth 2.0 được sử dụng, the policy này có thể được chỉ định trong khái niệm scopes như là “get:documents”

### Authorization

# Chapter 9: Sample Application with Custom API

## Background

Application bao gồm một stateless back-end API serving một SPA và native app.

The application đề xuất theo các tính năng và services như sau:

* Cho phép user viết bài trong Markdown
* Render the final text from the Markdown
* Aritcles được sở hữu bởi a single user
* Người sở hữu article có thể share với người khác
* Sharing dựa trên một simple attribute-based access control scheme
* Có 2 levels access tới article, reader và editor:
* Readers có thể xem articles
* Editors có thể tạo và update articles
* Khi một version mới của article được tạo, nó được phát hành như một document mới

Application tương tự web sites code-sharing phổ biến như JSFiddle hoặc Codepen.

## Application Requirements

### Who Are Your Users: Employees or Consumers?

Vì mục đích đơn giản, thiết kế của chúng ta giải định một user ở đây là consumer, người mà muốn login thông qua social. Với câu hỏi này sẽ giúp ta hiểu người dúng đến từ đâu

### How Will Users Log In?

Để mô tả cho phần lớn các case, the application cho phép users sign up một local account và sau đó có khả năng thêm các liên kết tới social tới local account. Cái này cho phép users log in thông qua social. The application sẽ duy trì một local profile cho users và cho phép users thay đổi name, email, picture. Sử dụng OIDC cho authentication. Câu hỏi này để ta biết ta cần một local repository của user profile infomation

### Can Your App Be Used Anonymously?

Users có thẻ ẩn danh và tạo một document ẩn danh. Để giữa application đơn giản, nếu user sign in, documantes ẩn danh của họ không được converted tới documents đã sở hữu tên và không hiện tên account

### Web-Based or Native App Format or Both?

Câu hỏi này giúp ta hiểu scope của project và chỉ rõ việc tạo API cần cho cả 2 fronte-end applicaions

### Does Your Application Call APIs

# Chapter 10: Sessions

Sự tương tác của user với một application trong một khoảng thời gian thì được biết là session. người dùng mong muốn làm nhiều giao dịch trong phiên của họ mà không cần xác thực mỗi lần làm. Để làm điều này, một ứng dụng cần theo dõi user đã được xác thực

## Application Sessions

Trong một session (phiên) của user, một application có thể cần theo dõi thông tin cụ thể như các giao dịch hoặc một sesion có thể tiếp tục trong bao lâu. Với web truyền thống, session của user có lẽ được duy trì trên server bằng việc gán nó một identifier và lưu trong memory, một filesytem, database hoặc shared service như Redis. The session identifier có thể được lưu trong cookie bởi application, cái này sau đó gửi bởi browser với mỗi request tới application server. Khi một request được nhận, một server có thể sử dụng session identifier từ cookie để lấy thông tin về session của user để xử lý request. Thi thoảng, nếu session data đủ nhỏ, nó có thể được chứa toàn bộ trong cookie, loại bỏ sự cần thiết của server-side storage

Web truyền thống giới hạn thời gian của session vì nếu người dùng quên đăng xuất, bị mất kế nói thì dẫn tới tiêu phí về tài nguyên,...

Với SPA và stateless back-end API, server-side sessions cho user thì không được yêu cầu, nhưng nội dung của một session timeout vẫn tồn tại cho lý do khác.