TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

─────── \* ───────



TIỂU LUẬN MÔN HỌC

**AN NINH TRONG GIAO DỊCH ĐIỆN TỬ**

Đề tài: Đăng nhập một lần và ủy quyền

Học viên thực hiện: **Phạm Duy Hưng** - CA170259 - TTM17A

**Bùi Nguyên Tùng**  - CA170262 - TTM17A

Giáo viên hướng dẫn: **PGS.TS** **Nguyễn Linh Giang**

HÀ NỘI 10-2017

**MỤC LỤC**

[ĐẶT VẤN ĐỀ 3](#_Toc496644003)

[CHƯƠNG 1: GIẢI PHÁP OPENID 4](#_Toc496644004)

[1. Giới thiệu OpenID 4](#_Toc496644005)

[2. Các tính năng của OpenID 5](#_Toc496644006)

[3. Mô hình hoạt động của OpenID 6](#_Toc496644007)

[4. Nhận xét 9](#_Toc496644008)

[CHƯƠNG 2: GIẢI PHÁP OAUTH 11](#_Toc496644009)

[1. Giới thiệu OAuth 11](#_Toc496644010)

[2. Cơ chế hoạt động của OAuth 12](#_Toc496644011)

[3. Nhận xét 19](#_Toc496644012)

[CHƯƠNG 3: GIẢI PHÁP SINGLE SIGN ON (SSO) 20](#_Toc496644013)

[1. Giới thiệu Single Sign On 20](#_Toc496644014)

[2. Các loại Single Sign On 22](#_Toc496644015)

[3. Kiến trúc SSO 25](#_Toc496644016)

[4. Nhận xét 28](#_Toc496644017)

# ĐẶT VẤN ĐỀ

Hiện nay rất nhiều ứng dụng dịch vụ bắt buộc phải tạo tài khoản để thực hiện những chức năng mà nó cung cấp và đây là một việc làm bắt buộc để xác thực người dùng. Việc tạo tài khoản yêu cầu khách viếng thăm mất khá nhiều thời gian và có cảm giác phiền toái. Tránh né những website yêu cầu có tài khoản, đây là yếu tố mất điểm không thể tránh khỏi đối với các WebApp. Lượng đăng ký tỷ lệ nghịch với số lượng thông tin bắt buộc, số lần nhấp chuột vào các ô nhập liệu trên form của ứng dụng.

Về phía người dùng, khi tham gia vào mỗi ứng dụng dịch vụ có xác thực bằng tên đăng nhập và mật khẩu, họ đều phải ghi nhớ tên đăng nhập và mật khẩu, lượng tài khoản gia tăng làm cho việc ghi nhớ mật khẩu ở mỗi ứng dụng web trở thành vấn đề lớn. Chắc chắn mỗi người trong chúng ta đã từng click vào link *“Click đây nếu quên mật khẩu”* để tìm lại mật khẩu cho những ứng dụng ít sử dụng. Số ít lại dán đầy những mảnh giấy ghi thông tin tài khoản.

Một số website có các tiện ích được phát triển trên những nền tảng khác nhau, yêu cầu đặt ra là người dùng chỉ cần đăng ký một lần ở duy nhất một tiện ích nhưng có thể sử dụng được tài khoản đó cho tất cả các tiện ích trên website.

Những vấn đề đó trở thành vật cản cho sự phát triển của các ứng dụng dịch vụ đang ngày càng phát triển và mở rộng, khiến cho các ứng dụng khác nhau khó có thể liên kết được với nhau. Do đó cần có giải pháp thỏa đáng để có thể triệt để giải quyết các vấn đề, thúc đẩy các nhà phát triển đưa ra giải pháp hỗ trợ cho việc đăng nhập tại một địa điểm duy nhất và có thể sử dụng thông tin xác thực đó cho mọi nơi.

Các giải pháp nghiên cứu trong bài tiểu luận:

* OpenID
* OAuth
* SSO (Single Sign-On)

# CHƯƠNG 1: GIẢI PHÁP OPENID

## Giới thiệu OpenID

OpenID là một chuẩn mở và là một giao thức xác thực phân cấp.

OpenID được phát triển bởi tổ chức phi lợi nhuận OpenID Foundation, OpenID cho phép người dùng có thể được xác thực bởi rất nhiều website (replying parties) sử dụng service của bên thứ 3. Nó giảm được việc phải thiết lập riêng tài khoản cho mỗi website, cho phép các user có thể login tới nhiều website mà không cần phải có những định danh và password riêng cho mỗi site.

OpenID giúp người dùng và website xác thực quyền truy cập, cho phép người dùng đăng nhập vào những ứng dụng web khác nhau chỉ bằng một định danh số (digital indentity). Giúp thay thế các thủ tục đăng ký, xác thực, đăng nhập truyền thống chỉ bằng một bước đăng nhập duy nhất.

Bạn có thể chọn việc liên kết thông tin với OpenID của bạn – thông tin này sẽ được chia sẻ tới các website mà bạn đã truy cập (ví dụ như tên hoặc email). Với OpenID, bạn có thể kiểm soát được lượng thông tin mà bạn muốn cho phép các website bạn đến thăm biết được.

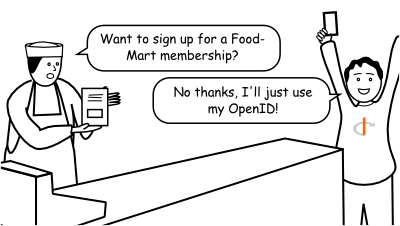
Với OpenID, chỉ duy nhất Identity Provider quản lý password, và provider này sẽ xác thực tài khoản của bạn tới các website mà bạn đến thăm, ko có một website nào có thể biết được password của bạn - đây là một yếu tố bảo mật rất cao.

OpenID nhanh chóng nhận được sự công nhận và sử dụng của cộng đồng web, với hơn 1 tỷ OpenID account được thiết lập và 50,000 webstie sử dụng phương thức OpenID để login. Một vài tổ chức lớn cho phép hoặc phát hành OpenIDs hoặc 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....



## Các tính năng của OpenID

1. ***Đăng ký thành viên nhanh và dễ dàng***



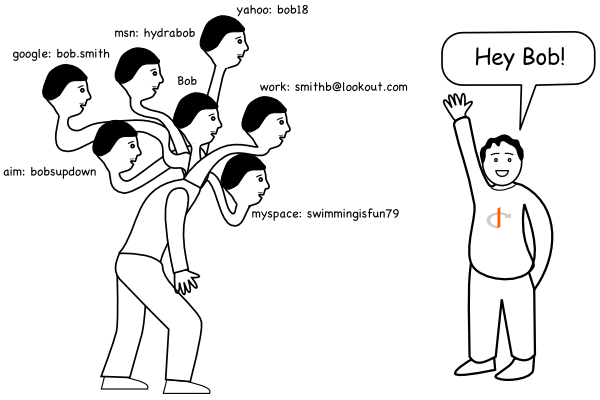
Với hình ảnh minh họa trên, ta có thể hình dung được sự tiện lợi này. Một người thu tiền hỏi anh khách hàng muốn đăng kí thành viên cho Food-Mart ko ? và khách hàng trả lời rất hào hứng "Không cần, Tôi sẽ sử dụng OpenID của tôi". Một OpenID chính là cách để định danh chính bạn ko quan trọng bạn ghé thăm website nào. Nó như kiểu là bằng lái xe cho toàn bộ Internet vậy. Nhưng còn hơn thế nữa là bởi vì bạn có thể quản lý được thông tin liên kết tới OpenID như là tên của bạn, địa chỉ mail của bạn, và bạn có thể chọn website nào được quyền thấy nó, cái nào ko. Điều này cũng đồng nghĩa với việc những website sử dụng OpenID sẽ ko hỏi đi hỏi lại bạn your name, your mail address lần nào cũng vậy.

1. ***Login nhanh và dễ dàng***

OpenID còn rất dễ dàng để login. Với OpenID bạn chỉ cần nhớ 1 username và 1 password. Bởi vì bạn login vào website với OpenID của bạn, cho nên bạn chỉ cần bảo mật với OpenID là đủ rồi. Password này của bạn thực tế là chỉ được quản lý bởi OpenID provider của bạn, ko phải là các website bạn đang login, khi bạn login vào OpenID provider sẽ bảo với các website đó là bạn đang login đấy và bạn là ai. Ko có bất kì một website nào nhìn được password của bạn, cho nên bạn ko cần phải quá bận tâm về vấn đề bảo mật khi sử dụng OpenID.

1. ***Tiệm cận đến khái niệm “web identity”***

Bởi vì OpenID định danh bạn là duy nhất trên Internet, một khi bạn thiết lập mình như là người sử dụng OpenID, bất cứ khi nào ai đó nhìn thấy OpenID của bạn được sử dụng, bất cứ nơi nào trên Internet, họ sẽ biết rằng đó chính là bạn. Tương tự như vậy, nếu bạn truy cập vào một trang web mới và thấy rằng một người nào đó với OpenID của người bạn của bạn, bạn có thể gần như chắc chắn rằng đó là bạn của bạn.



Điều đó có thể khiến bạn lo lắng rằng OpenID là sẽ làm cho tất cả các hoạt động trực tuyến của bạn trở nên quá rõ ràng và dễ dàng bị phát hiện. Hãy yên tâm, mặc dù OpenID thống nhất thông tin về bạn, nhưng nó chỉ hợp nhất thông tin mà bạn đã cho phép public. Bạn có thể chọn, sử dụng OpenID, thông tin nào và thông tin đó ai sẽ thấy được.

1. ***Tính an toàn bảo mật***

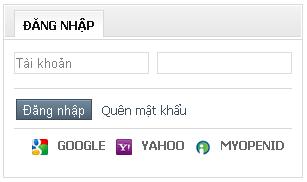
OpenID là thực sự cũng ko kém bảo mật hơn hay là an toàn hơn so với những gì bạn sử dụng ngay bây giờ. Sự thật là nếu ai đó có được tên truy cập và mật khẩu của OpenID của bạn, họ có thể chiếm đoạt định danh của bạn. Hầu hết các trang web cung cấp việc reset password thông qua email trong trường hợp nếu bạn đã quên nó, có nghĩa là nếu một người nào đó có được tài khoản e-mail của bạn, họ cũng có thể làm giống như trong trường hợp họ có được tên truy nhập và mật khẩu của OpenID của bạn. Họ có thể kiểm tra các trang web mà họ nghĩ rằng bạn có một tài khoản và yêu cầu thay mật khẩu bị quên. Tương tự như vậy, nếu một ai đó có quyền truy cập vào OpenID của bạn, họ có thể tìm kiếm trên Internet đến những nơi mà họ nghĩ rằng bạn có tài khoản và đăng nhập như bạn ... không có gì khác biệt về bảo mật ở đây.

Cho nên, ko liên quan đến việc bạn sử dụng OpenID hay ko, bạn nên cẩn thận về tài khoản của bạn - username và pwd. Khi bạn gõ username và pwd, hãy chắc chắn ràng bạn đang ở trên website mà bạn nghĩ là bạn muốn vào

## Mô hình hoạt động của OpenID

1. ***Đăng nhập bằng tài khoản OpenID***

Người dùng khi ghé thăm một website có hỗ trợ OpenID (vd: website.relying.com) lựa chọn một tác vụ yêu cầu phải đăng nhập. Không giống như cách đăng nhập thông thường là sử dụng “username” và “password”. Người dùng sẽ thấy một trường duy nhất là Các WebApp dựa vào OpenID URL để xác định người dùng vì OpenID URL là duy nhất cho một người dùng.



*Đăng nhập bằng Open ID*

Đồng thời cũng dựa vào đó để xác định Provider trong quá trình giao tiếp. Người sử dụng sau khi nhập OpenID URL của mình vào trường đăng nhập, sẽ có 2 trường hợp xãy ra.

*1: WebApp xác nhận được Provider và ngay lập tức chuyển người dùng tới một cửa sổ mới của Provider để Provider xác nhận lại người dùng.  
2: WebApp hỏi người dùng có muốn tiến hành xác nhận bằng Provider hay không.*

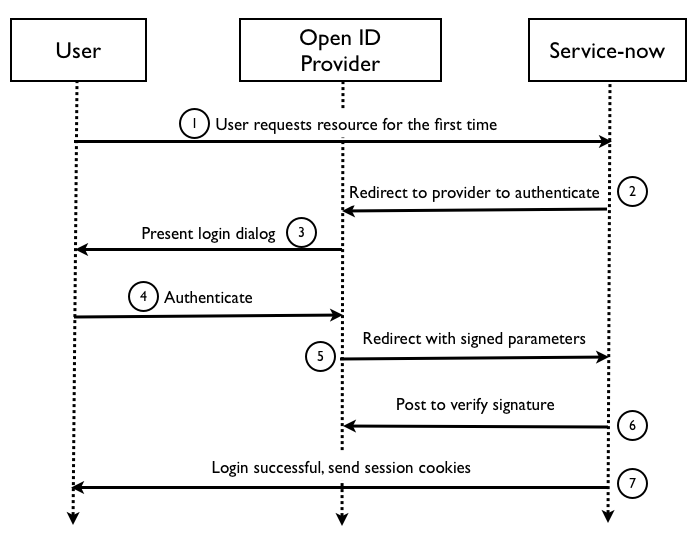
Sau khi đăng nhập bằng tài khoản của mình tại Provider, người dùng được Provider hỏi có đồng ý đăng nhập vào WebApp hiện tại hay không. Người dùng Click để xác nhận đồng ý và được chuyển trở lại WebApp. Người dùng tiếp tục những tác vụ yêu cầu phải đăng nhập sau khi đã được Provider xác thực.

Giữa WebApp và Provider tiến hành một số giao tiếp để Provider lưu giữ phê chuẩn cho WebApp và thuận tiện cho quá trình quản lý của người dùng. WebApp lấy một số thông tin về người dùng từ phía Provider.

1. ***Mô hình hoạt động của OpenID***

Giả sử một người dùng đang cố gắng truy cập tài nguyên là một phần RP của website, và RP sử dụng OpenID. Để truy cập vào các nguồn tài nguyên, người sử dụng phải xuất trình OpenID của mình trong một form có thể được công nhận (bình thường) như OpenID. OpenID được mã hóa với vị trí của OP. RP sau đó lấy định danh của người dùng và chuyển hướng người dùng đến OP, nơi đó user sẽ được yêu cầu để chứng minh yêu cầu của mình sử dụng ID đó.

Hãy cùng điểm qua các thành phần chi tiết của OpenID và mỗi vài trò của nó trong cả quá trình xử lý.



1. User requests resource for the first time: người dùng yêu cầu một tài nguyên từ RP (Service Now) lần đầu tiên chưa có xác thực từ OP (yêu cầu không chứa OpenID URL parameters)
2. Redirect to provider to authenticate: RP chuyển yêu cầu tới OP với yêu cầu xác thực
3. Present login dialog: OP đòi người dùng cung cấp tài khoản
4. Authenticate: Người dùng đăng nhập tài khoản và xác thực thành công
5. Redirect with signed parameters: OP gửi một yêu cầu, bao gồm cả OpenID URL tới RP
6. Post to verify signature”: RP xác nhận lại yêu cầu với OP
7. Login successful, send session cookies: RP đã được người dùng đăng nhập với tham số “user\_key”.

## Nhận xét

***Ưu điểm:***

***Đối với Website và nhà phát triển WebApp:***

* Tăng tỷ lệ “khách viếng thăm” trở thành “thành viên”:

Nghiên cứu của Google cho thấy tỷ lệ người viếng thăm (Lần đầu vào WebApp) kích hoạt tài khoản ở những trang có sử dụng OpenID là 90% do việc trở thành “thành viên” chỉ đơn giản thực hiện vài cái nhấp chuột thay vì phải điền quá nhiều thông tin và xác thực địa chỉ email như cách làm truyền thống.

* Giảm sự thất vọng của người dùng trong việc nhớ mật khẩu:

Đăng nhập bằng OpenID chỉ bằng một trường OpenID URL và người dùng chỉ cần nhớ thông tin này thay vì phải ghi tên sử dụng, mật khẩu ra giấy, note … không còn phải quan tâm tới link “click đây nếu quên mật khẩu” như thường gặp nữa.

* Kết nối website tới các mạng xã hội:

Các mạng xã hội như Yahoo Meme, Google Buzz, MySpace, Facebook … đều đã hổ trợ OpenID cho tất cả các tài khoản của họ, do đó bạn có thể tận dụng lượng người dùng này, quảng bá thông tin về WebApp để thu hút lượng người dùng từ các mạng xã hội. Hơn nữa, bạn còn có thể tiếp cận với danh sách liên kết của họ để tiến hành các chiến dịch SEO của mình tại đây.

* Mở ra khả năng kết nối với các ứng dụng khác nhau:

Ví dụ: Quá trình thâu tóm của Google đối với các ứng dụng như YouTube, OrKut, Blogger… phát triển trên các nền tảng khác nhau, nhưng các tài khoản lập tại Google hay các ứng dụng này đều có thể đăng nhập và sử dụng được mà không gặp trở ngại là nhờ cùng sử dụng cơ chế xác thực của OpenID.

***Đối với người dùng cuối:***

* Việc đăng ký trở nên dễ dàng hơn:

Bỏ qua một số bước bắt buộc và mất nhiều thời gian, cảm thấy đơn giản để vào tài khoản. Các thông tin của người dùng cần thiết cho ứng dụng có thể dần được bổ sung sau này.

* Đơn giản vấn đề mật khẩu:

Thay vì phải giấy bút và cố gắng nhớ hết mật khẩu ở tất cả các forum, blog, webapp… Người dùng cuối chỉ cần nhớ thông tin về OpenID URL.

* Có quyền quyết định nơi lưu trữ thông tin cá nhân:

Provider lưu trữ một số thông tin cá nhân và bạn có quyền quyết định lựa chọn Provider mình tin tưởng để lưu trữ thông tin.

* Giảm thiểu rủi ro và bảo mật:

Quá trình xác thực Provider yêu cầu người dùng xác nhận cho phép WebApp truy cập thông tin. Một số Provider tạo ra con dấu điện tử để đề phòng tấn công phising. Cùng với những biện pháp bảo mật khác nhau. OpenID đang ngày trở nên an toàn hơn so với cách lưu trữ mật khẩu truyền thống.

***Nhược điểm:***

* Triển khai OpenID:

+ Yêu cầu lớn đối với Provider: Với những public Provider, phải đảm bảo server đủ mạnh để quá trình xác thực diễn ra nhanh chóng, không mất quá nhiều thời gian của người dùng cuối.

+ Để triển khai Provider, đòi hỏi developer phải am hiểu về bảo mật và webservice. Đảm bảo các giao tiếp không bị cướp phiên giao dịch, an toàn trước các kiểu tấn công ăn cắp mật khẩu.

+ Các Provider khác nhau lưu trữ những thông tin khác nhau: OpenID ra đời năm 2005, trong khi các Provider lớn như Google, Yahoo, Windows Live ra đời từ khá lâu, do đó thông tin lưu trữ về người dùng của các Provider này cũng không giống nhau, OpenID Foundation đang dần nâng cấp các tiêu chuẩn và phiên bản để đồng bộ hóa tất cả các Provider này.

* Bảo mật đối với OpenID:

+ Một số ý kiến cho rằng OpenID là một điểm yếu trong lừa đảo phishing (một kiểu giả dạng website để đánh cắp mật khẩu). Ví dụ, lúc vào một WebApp có hổ trợ OpenID, ở đó chuyển đến trang yêu cầu xác thực bằng username và password của Provider giả. Vậy là họ bị hack pass ở Provider giả mạo này. Và trong nổ lực chống tình trạng phishing, nhiều phương án được đưa ra như xác nhận cho phép đăng nhập, con dấu cho Provider… tùy thuộc vào từng Provider.

+ Tháng 12/2008, Nhóm phát triển OpenID ban hành chính sách xác thực mở rộng, yêu cầu tất cả các Provider phải cho người dùng quản lý và phê chuẩn tất cả các kết nối đến OpenID URL của họ.

# CHƯƠNG 2: GIẢI PHÁP OAUTH

## Giới thiệu OAuth

Gần đây, khi đăng ký hoặc đăng nhập các website chúng ta thường thấy có thêm các chức năng như đăng nhập thông qua tài khoản Facebook, Google… chức năng này rất thuận tiện. Thực tế, mỗi người trong chúng ta đã từng có hàng vài chục tài khoản ở các website khác nhau đến nỗi phải đặt các tài khoản này giống hệt nhau. OAuth ra đời nhằm giải quyết vấn đề trên và xa hơn nữa, OAuth là một giao thức chứng thực nhờ đó một ứng dụng bên thứ 3 có thể đại diện cho người dùng truy cập vào tài nguyên người dùng nằm trong một dịch vụ nào đó.

OAuth là một phương thức chứng thực giúp các ứng dụng có thể chia sẻ tài nguyên với nhau mà không cần chia sẻ thông tin username và password. Từ Auth ở đây mang 2 nghĩa:

* Authentication: xác thực người dùng thông qua việc đăng nhập.
* Authorization: cấp quyền truy cập vào các Resource.

**Lịch sử phát triển OAuth:**

Chúng ta cùng điểm lại vài nét trong lịch sử phát triển của OAuth:

Năm 2006, Twitter phát triển hệ thống OpenID phục vụ cho đăng nhập các ứng dụng trong hệ thống của Twitter, tuy nhiên hệ thống này yêu cầu người dùng phải cung cấp tên đăng nhập và mật khẩu, đây là một điểm yếu. Cũng chính năm này, các ông lớn trong về mạng xã hội như Facebook, Google, Twitter… đã cùng ngồi với nhau để phác thảo ý tưởng về một hệ thống xác thực mới giúp các ứng dụng bên thứ ba có thể tích hợp.

Năm 2008, IETF – tổ chức quản lý tiêu chuẩn mạng Internet (tránh nhầm với IEEE tổ chức quản lý tiêu chuẩn trong lĩnh vực điện và điện tử) đã quyết định hỗ trợ tiêu chuẩn OAuth này và bắt đầu cho xây dựng RFC 1.0.

Năm 2010, IETF phát hành phiên bản chính thức đầu tiên của OAuth 1.0 (RFC 5849).

Sau đó, lỗi bảo mật nghiêm trọng được phát hiện với tên gọi Session Fixation xảy ra trên OAuth 1.0, cho phép các hacker lừa ứng dụng bên thứ ba trao quyền truy nhập vào tài khoản và dữ liệu người dùng.

Năm 2012, phiên bản OAuth 2.0 ra đời, tuy nhiên vẫn còn những lỗi bảo mật chết người các hacker có thể Hack Facebook thông qua trình duyệt Chrome. Từ đó đến nay chưa có thêm một phiên bản nào khác của OAuth ra đời thêm.

OAuth2 không chỉ đơn thuần là một giao thức (protocol) mà nó còn là một nền tảng (framework) giúp xây dựng ứng dụng cả ở client và server.

## Cơ chế hoạt động của OAuth

1. ***Lược đồ luồng thực hiện của OAuth:***

Các thuật ngữ hay sử dụng trong OAuth:

* Authorization Grant: cấp ủy quyền
* Resource Owner: Người sở hữu tài nguyên hay chính là người dùng (user)
* Authorization server: máy chủ ủy quyền
* Resource server: máy chủ tài nguyên
* Authorization code: mã ủy quyền.
* Access token: thẻ truy nhập, giống như thẻ ra vào tòa nhà, khi có thẻ này bạn có thể vào các vùng dữ liệu để sử dụng.
* Refresh token: thẻ yêu cầu cấp mới, khi các thẻ truy nhập hết hạn sẽ dùng thẻ yêu cầu cấp mới để hệ thống máy chủ ủy quyền cấp mới access token.

Chúng ta cùng bắt đầu tìm hiểu kỹ hơn về OAuth 2 thông qua lược đồ luồng thực hiện. OAuth có 4 đối tượng cần chú ý:

* Resource Owner: Chủ sở hữu dữ liệu
* Client: Ứng dụng, website bên thứ 3
* Authorization server: Máy chủ ủy quyền
* Resource server: Máy chủ tài nguyên, dữ liệu

Sơ đồ luồng hoạt động của OAuth2:



1. Application yêu cầu ủy quyền để truy cập vào Resource Server thông qua User

2. Nếu User ủy quyền cho yêu cầu trên, Application sẽ nhận được giấy ủy quyền từ phía User (dưới dạng một token string nào đó chẳng hạn)

3. Application gửi thông tin định danh (ID) của mình kèm theo giấy ủy quyền của User tới Authorization Server

4. Nếu thông tin định danh được xác thực và giấy ủy quyền hợp lệ, Authorization Server sẽ trả về cho Application access\_token. Đến đây quá trình ủy quyền hoàn tất.

5. Để truy cập vào tài nguyên (resource) từ Resource Server và lấy thông tin, Application sẽ phải đưa ra access\_token để xác thực.

6. Nếu access\_token hợp lệ, Resource Server sẽ trả về dữ liệu của tài nguyên đã được yêu cầu cho Application.

Luồng hoạt động thực tế có thể sẽ khác nhau tùy thuộc vào việc ứng dụng sử dụng loại ủy quyền (authorization grant type) nào, trên đây chỉ là ý tưởng chung để thực hiện.

1. ***Đăng ký ứng dụng***

Trước khi sử dụng OAuth do một dịch vụ nào đó cung cấp (ví dụ Facebook, Github, Google…) trong ứng dụng, cần đăng ký ứng dụng với dịch vụ đó. Đăng ký ứng dụng là điền các thông tin vào form đăng ký của dịch vụ, các thông tin này bao gồm:

* Application name: Tên ứng dụng
* Application website: Trang web của ứng dụng
* Redirect URI hay Callback URL: Đường dẫn trỏ về ứng dụng của bạn để nhận kết quả, khi đường dẫn này kích hoạt bạn có thể nhận được các thông tin như mã ủy quyền hoặc access token.

Khi ứng dụng được đăng ký, dịch vụ sẽ sinh ra cặp dữ liệu Client Id và Client Secret cho ứng dụng. Client ID dùng để dịch vụ nhận diện được ứng dụng, Client Secret là một mã bí mật được sử dụng để xác thực ứng dụng với client ID.

1. ***Cấp ủy quyền***

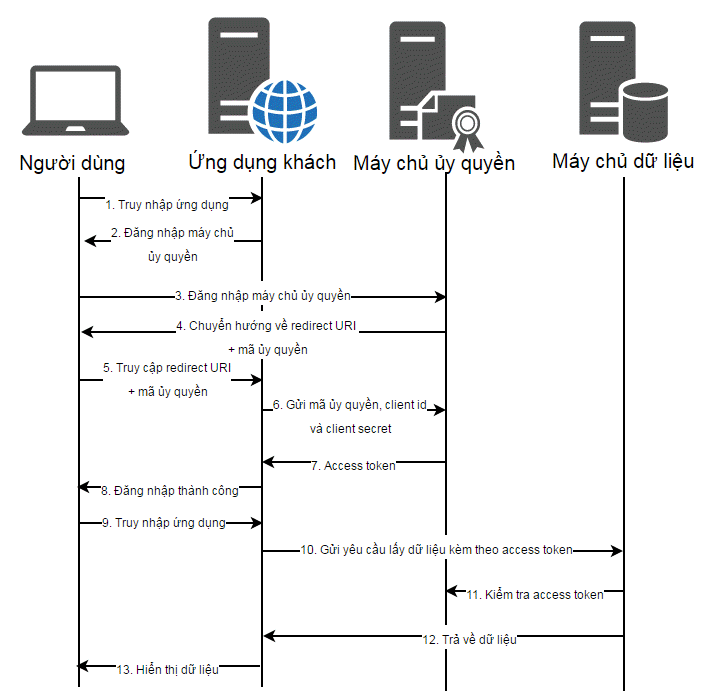
Trong mô hình cơ bản ở trên, bốn bước đầu tiên là để lấy được mã ủy quyền (authorization code) và thẻ truy nhập (access token). Mã ủy quyền sẽ được sử dụng bởi ứng dụng để thực hiện lấy access token, khi có được access token, ứng dụng có thể lấy được các tài nguyên thuộc về người dùng. Các dạng ủy quyền phụ thuộc vào cách thức ứng dụng yêu cầu xác thực và dạng ủy quyền được hỗ trợ bởi máy chủ ủy quyền. OAuth 2 có bốn dạng ủy quyền là:

* Mã ủy quyền: sử dụng với các ứng dụng phía máy chủ.
* Ủy quyền ngầm định: được sử dụng bởi các ứng dụng điện thoại và ứng dụng web (các ứng dụng chạy trên thiết bị người dùng cần xác thực).
* Ủy quyền bằng thông tin người dùng: sử dụng với các ứng dụng là các thành viên mở rộng của hệ thống trong đó máy chủ ủy quyền và máy chủ tài nguyên là các thành viên.
* Ủy quyền bằng thông tin ứng dụng: sử dụng với các truy nhập API ứng dụng.

***2.3.1 Cấp ủy quyền sử dụng mã ủy quyền***

Loại ủy quyền sử dụng 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 được công khai do đó các thông tin ứng dụng như client id và client secret được quản lý tốt. Để sử dụng được cách thức cấp ủy quyền này, ứng dụng phải có khả năng tương tác với người dùng (ví dụ, ứng dụng phải thiết lập được các giao diện yêu cầu người dùng đăng nhập máy chủ ủy quyền như “nút đăng nhập sử dụng Facebook”) và nhận mã ủy quyền từ người dùng thông qua các giao diện.

Chúng ta cùng xem xét Lược đồ cấp ủy quyền sử dụng mã ủy quyền như sau:



1) Người dùng truy nhập vào ứng dụng

2) Ứng dụng đưa ra màn hình có nút “đăng nhập bằng Facebook, Google, Twitter”

3) Người dùng đăng nhập máy chủ ủy quyền, tức là đăng nhập vào Facebook, Google, Twitter (người dùng để ý thì các đường link đăng nhập này chính xác là của Facebook, Google hoặc Twitter…).

4) Facebook, Google, Twitter… sau khi kiểm tra sẽ hiển thị các cảnh báo về quyền hạn của ứng dụng.

5) Người dùng xác nhận cấp quyền cho ứng dụng và cũng là thực hiện chạy redirect URI kèm theo mã ủy quyền.

https://application.com/callback?code=AUTHORIZATION\_CODE

6) Ứng dụng đã có mã ủy quyền và có thể thực hiện yêu cầu máy chủ ủy quyền cung cấp access token bằng cách gửi đi client id, client secret và mã ủy quyền ở dạng POST

https://authorization-server.com/oauth/token?client\_id=CLIENT\_ID&client\_secret=CLIENT\_SECRET&grant\_type=authorization\_code&code=AUTHORIZATION\_CODE&redirect\_uri=CALLBACK\_URL

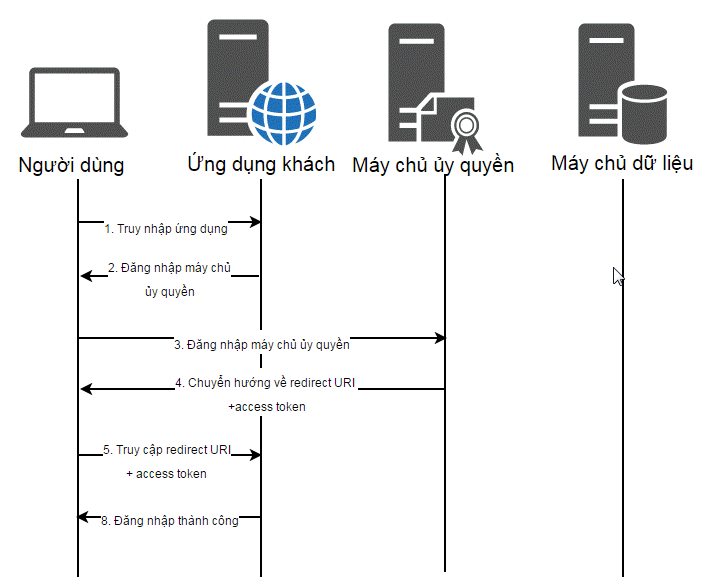
7) Máy chủ ủy quyền kiểm tra client id, client secret, mã ủy quyền (authorization code) nếu không thấy bất thường sẽ gửi trả lại ứng dụng access token.

8) Như vậy, người dùng đã được xác thực, các bước tiếp theo từ 9-13, người dùng yêu cầu dữ liệu nào, ứng dụng sẽ sử dụng access token để lấy dữ liệu từ máy chủ tài nguyên.

***2.3.2 Cấp ủy quyền ngầm định***

Cấp ủy quyền ngầm định thường sử dụng cho các ứng dụng điện thoại, đặc điểm là các ứng dụng này chỉ sử dụng bởi duy nhất một người là chủ sở hữu của chiếc điện thoại này. Ví dụ, bạn sử dụng điện thoại Android, khi thiết lập ban đầu bạn phải nhập vào tài khoản của Google, do đó tất cả các ứng dụng tích hợp xác thực Google sẽ tự động xác thực bạn luôn mà không cần phải thực hiện lại các bước như ở cấp ủy quyền bằng mã ủy quyền. Refresh token cũng không cần thiết trong cấp ủy quyền ngầm định.

Lược đồ cấp ủy quyền ngầm định như sau:



1) Người dùng truy cập ứng dụng

2) Ứng dụng hiển thị đăng nhập máy chủ ủy quyền cũng chính là gửi request đến máy chủ ủy quyền link có dạng như sau:

https://authorization-server.com/oauth/authorize?response\_type=token&client\_id=CLIENT\_ID&redirect\_uri=CALLBACK\_URL&scope=read

3) Người dùng nhập thông tin vào màn hình trên.

4) Máy chủ ủy quyền kiểm tra nếu không có bất thường sẽ chuyển hướng người dùng về redirect URI kèm theo access token.

https://application.com/callback#token=ACCESS\_TOKEN

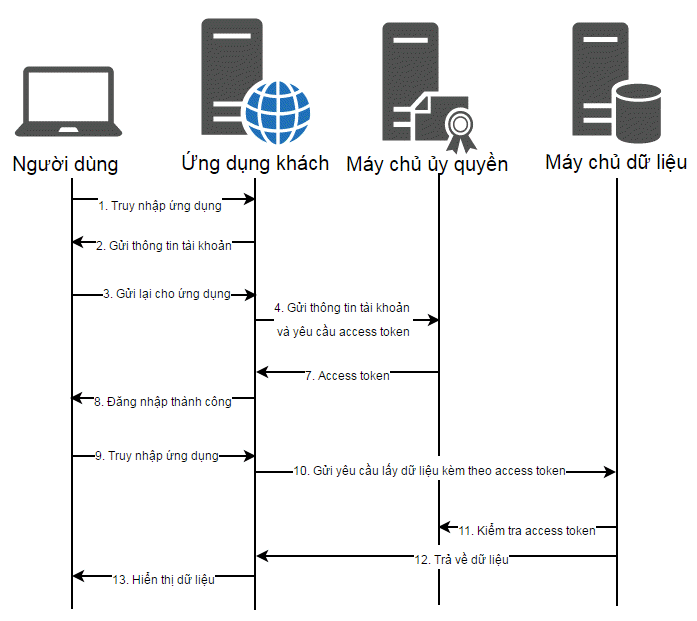
5) Ứng dụng nhận được access token do người dùng nhấn vào link trên.

8) Người dùng được xác thực

9 – 13) Tiếp theo cũng giống cấp ủy quyền bằng mã ủy quyền, ứng dụng sử dụng access token để lấy dữ liệu do người dùng ủy quyền thực hiện.

***2.3.3 Cấp ủy quyền bằng thông tin người dùng***

Người dùng cung cấp thông tin bao gồm username và password trực tiếp cho ứng dụng, khi đó ứng dụng sẽ dùng thông tin này yêu cầu máy chủ ủy quyền cấp access token. Loại ủy quyền này chỉ sử dụng khi các ứng dụng được người dùng tin tưởng hay ứng dụng là một thành phần trong hệ thống lớn trong đó máy chủ ủy quyền, máy chủ tài nguyên cũng là thành phần trong hệ thống đó.



Ví dụ, khi Facebook ra mắt một ứng dụng web A mới, ứng dụng A này là một thành phần của Facebook, do đó người dùng tin tưởng Facebook thì cũng tin tưởng ứng dụng A, do vậy người dùng có thể đăng nhập bằng username, password của Facebook vào ứng dụng A.

Ứng dụng A sẽ sử dụng thông tin người dùng (username, password) yêu cầu Facebook cấp access token, request này dạng POST và URL có dạng như sau:

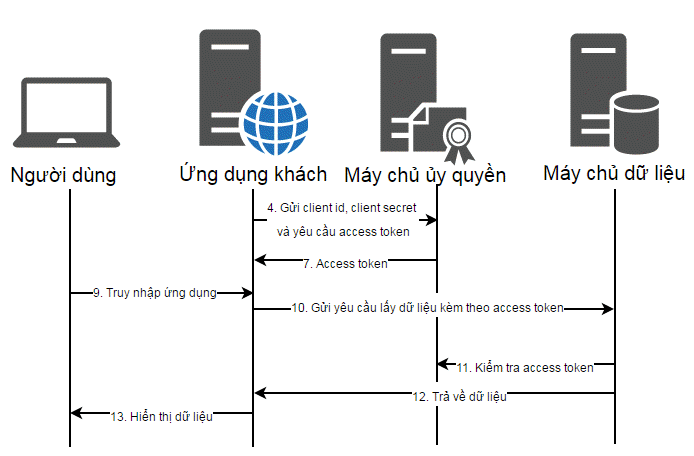
https://oauth.example.com/token?grant\_type=password&username=USERNAME&password=PASSWORD&client\_id=CLIENT\_ID

Máy chủ ủy quyền sẽ kiểm tra thông tin người dùng cùng với client id của ứng dụng, nếu không có vấn đề gì, nó sẽ cung cấp cho ứng dụng A access token. Như vậy ứng dụng A sẽ có quyền truy xuất tài nguyên trên máy chủ tài nguyên.

Các dịch vụ lớn như Facebook, Google, Twitter không bao giờ cung cấp kiểu ủy quyền bằng thông tin người dùng cho các ứng dụng ngoài hệ thống của họ do nếu cung cấp, các ứng dụng ngoài có thể lấy ghi nhận lại thông tin tài khoản người dùng và rất dễ dẫn đến các tình huống như lộ thông tin tài khoản, sử dụng thông tin này khai thác dữ liệu ngoài mục đích…

***2.3.4 Cấp ủy quyền bằng thông tin ứng dụng***

Trường hợp này thường dùng khi ứng dụng cần truy nhập tài nguyên hoặc gọi các phương thức từ máy chủ tài nguyên, các tài nguyên này không phụ thuộc vào người dùng xác định (tài nguyên dùng chung).



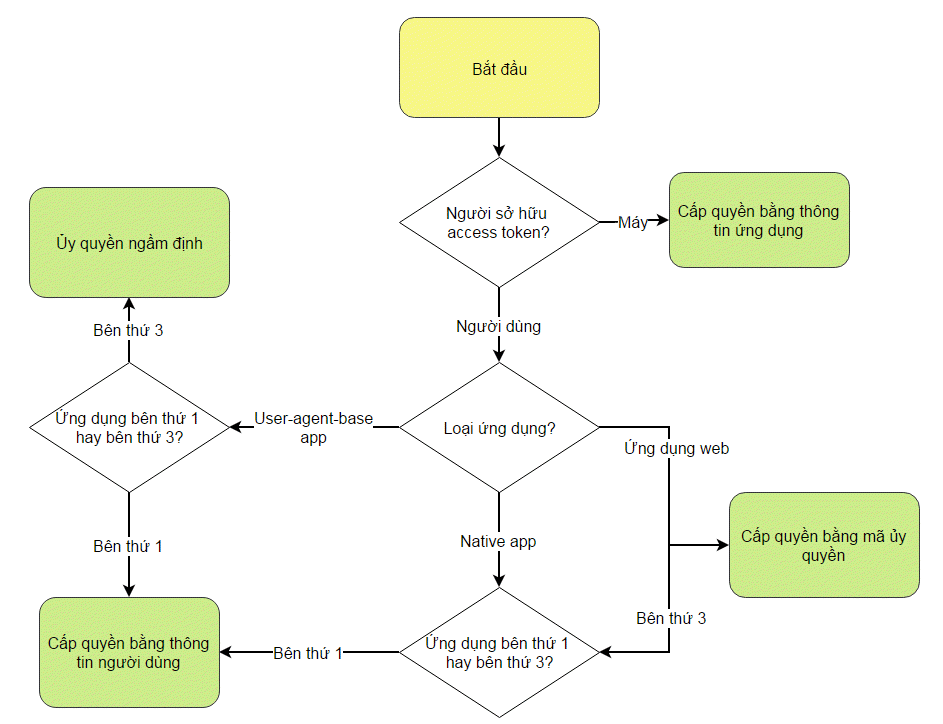
Ứng dụng yêu cầu access token bằng cách gửi đi các thông tin bí mật gồm client id và client secret đến máy chủ ủy quyền. Thực tế là ứng dụng khách gửi request dạng POST với đường link có kiểu cách như sau:

https://oauth.example.com/token?grant\_type=client\_credentials&client\_id=CLIENT\_ID&client\_secret=CLIENT\_SECRET

Khi đó, máy chủ ủy quyền kiểm tra thông tin bí mật của ứng dụng nếu đúng nó sẽ gửi trả access token. Lúc này ứng dụng được cấp quyền để truy cập các tài nguyên.

***2.3.5 Nên sử dụng loại cấp ủy quyền nào?***

Sơ đồ lựa chọn loại ủy quyền trong OAuth 2 dưới đây sẽ giúp ta lựa chọn chính xác loại ủy quyền mà ta cần sử dụng hoặc tự xây dựng một hệ thống khi sử dụng OAuth 2.



## Nhận xét

Nhiều người dùng có câu hỏi OAuth 2 có an toàn không? Chuyện gì xảy ra khi người dùng đăng nhập bằng nick Facebook, Google? Câu trả lời là rất an toàn. Chúng ta khi đăng nhập ứng dụng bằng tài khoản Facebook, Google… nhưng ứng dụng không hề biết các thông tin tên đăng nhập và mật khẩu này. Trừ khi dịch vụ đó sử dụng cấp quyền bằng thông tin người dùng, nhưng đã nói ở trên, không bao giờ các dịch vụ này cấp quyền bằng thông tin người dùng cho ứng dụng ngoài hệ thống của họ.

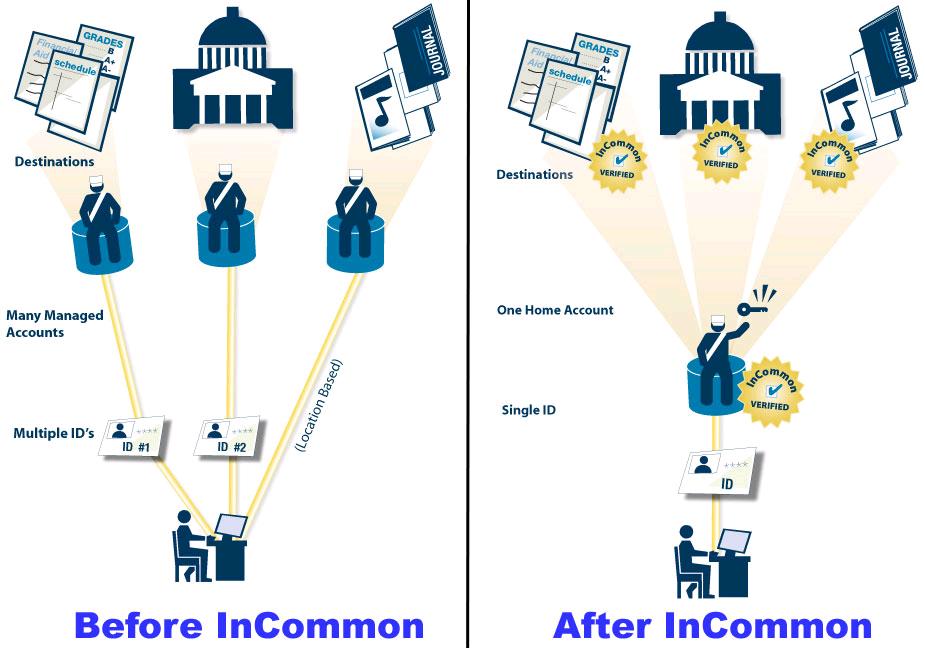
Oauth2 không chỉ là một "protocol" mà còn là một "framework", đòi hỏi bạn phải implement cả ở phía server và client. Ngoài ra việc oauth2 tách authorization server và resource server ra cũng khiến cho việc implement khá là mất công nếu làm từ đầu.

# CHƯƠNG 3: GIẢI PHÁP SINGLE SIGN ON (SSO)

## Giới thiệu Single Sign On

* 1. ***Khái niệm***

Single Sign On: Người sử dụng dùng tài khoản đăng nhập chỉ một lần và có thể sử dụng nhiều ứng dụng trong cùng một tổ chức một cách liên tục mà không cần phải nhập lại thông tin cho mỗi ứng dụng.



Ví dụ: Người dùng sử dụng nhiều dịch vụ: Email, forum, web…khi chưa có Single Sign On thì với mỗi dịch vụ ta phải nhập thông tin để xác thực.Khi một tổ chức đã thống nhất sử dụng Single Sign On cho hệ thống của họ thì người dùng chỉ phải đăng nhập một lần vào bất kì ứng dụng nào trong hệ thống thì khi dùng các ứng dụng khác người dùng không phải đăng nhập lại.

* 1. ***Tại sao sử dụng SSO***

Hệ thống CNTT ngày càng phát triển rộng khắp và ứng dụng vào tất cả các lĩnh vực. Người sử dụng phải đối mặt với vấn đề ngày càng phức tạp là phải nhớ nhiều tài khoản và mật khải để thực hiện công việc của họ.

Người dùng thường phải đăng nhập vào nhiều hệ thống, cần phải có một lượng tương đương số lần đăng nhập, mỗi dịch vụ trong các hệ thống đó có thể bao gồm tài khoản và mật khẩu khác nhau. Điều này dẫn đến người dùng phải đối mặt với các vấn đề:

* Quên tài khoản:Do sử dụng nhiều tài khoản và mật khẩu để đăng nhập các ứng dụng.
* Dùng một tài khoản và mật khẩu cho các hệ thống dẫn đến nguy cơ mất an toàn .

**Sử dụng Single Sign On:**

* Nhiều phương pháp xác thực: Là cần thiết để nhập thông tin đăng nhập của mình và ngoại trừ tài khoản và mật khẩu khi truy cập vào mỗi ứng dụng.
* Đa phương diện: Tài khoản người dùng là duy nhất trong tổ chức này và nó sẽ được mong muốn sử dụng nó truy cập vào tài nguyên máy tính từ các tổ chức khác có thể được sử dụng cùng một tài khoản.
* Quyền: Xác thực quyền hạn truy cập các tài nguyên của người dùng khi truy cập các ứng dụng khác nhau.
* Xác thực tập chung: Trên một máy chủ duy nhất để xác thực tài khoản người dùng thông qua giao thức đã được mã hóa.
* Trình duyệt sử dụng HTTP chuyển hướng trình duyệt của người dùng giữa các ứng dụng mà không cần đăng nhập tài khoản và mật khẩu.
  1. ***Các khái niệm quan trọng của SSO***

Xác thực: Quá trình xác minh danh tính của người dùng, làm cho chắc chắn rằng người dùng là họ.Điều này có thể dựa trên thông tin đăng nhập và mật khẩu kết hợp ,hoặc thẻ thông minh,sinh trắc học…

Ủy quyền: Là một quy trình nhằm xác minh rằng một người dùng biết trước, có quyền lực để thao tác một quá trình hoạt động nào đó hay không.

Giấy chứng nhận: Là các chi tiết được cung cấp bởi một người dùng trong quá trình xác thực vào ứng dụng.

Domain: Là sự nhận dạng vị trí của một máy tính trên mạng Internet nói cách khác tên miền là tên của các mạng lưới, tên của các máy chủ trên mạng Internet.

Bảo vệ dữ liệu: Dữ liệu của chúng cũng không cho tất cả mọi người có thể truy cập. Người dùng cần phải thông qua xác thực và ủy quyền trước khi truy cập vào một nguồn tài nguyên bảo vệ.

Cookie: Cookie là các thông tin lưu trong máy tính thường được dùng để nhận ra người dùng khi viếng thăm một trang web. Nó là những tập tin mà trang web gửi đến máy tính của người dùng. Các nội dung thực tế của một cookie có thể khác nhau tùy thuộc vào ứng dụng.Sau khi có một cookie, nếu người dùng duyệt để một ứng dụng khác nhau đó là một phần của Single Sign On,thông tin được lưu trong cookie sẽ được trình duyệt sử dụng để trực tiếp đăng nhập vào ứng dụng khác,nếu người sử dụng được phép truy nhập vào.

* 1. ***Ý nghĩa của SSO***
* Tránh việc nhớ nhiều thông tin đăng nhập (username & password) khi dùng nhiều dịch vụ.
* Tiết kiệm thời gian khi tái lập lại mật khẩu cho một người dùng (identity user).
* Bảo mật tất cả các cấp độ của việc thoát hay truy xuất vào hệ thống.
* Người phát triển ứng dụng không cần phải hiểu và thực hiện nhận dạng bảo mật trong ứng dụng của họ.

## Các loại Single Sign On

Hầu hết các sản phẩm SSO có sẵn trên thị trường có thể được phân loại thành hai loại dựa trên kiến trúc:

* 1. ***Trên web (Enterprise SSO)***

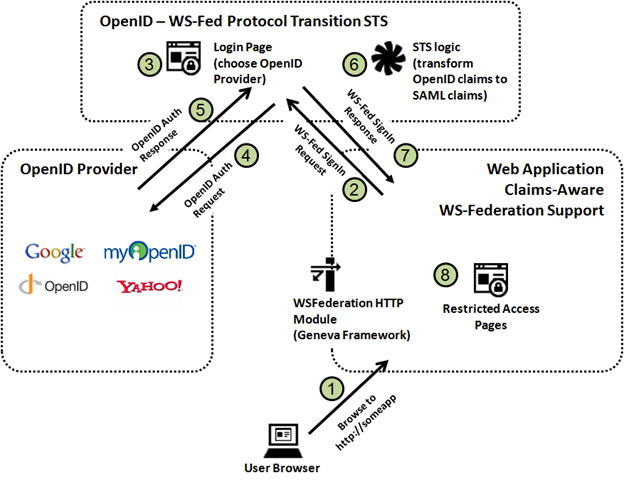
SSO trên web được sử dụng dưới các dạng:

Single Domain: Khi xác thực thành công vào domain.com, người dùng đồng thời được xác thực vào các sub-domain.domain.com tồn tại.

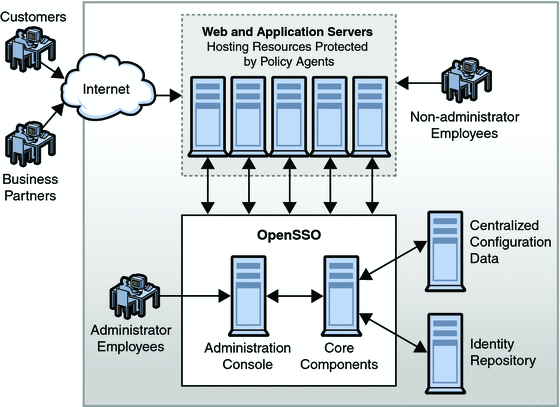
Multi Domain: Khi xác thực thành công vào facebook.com, người dùng đồng thời được xác thực vào example.com.

Các SSO được sử dụng phổ biến trên web hiện nay:

- OpenID: Hệ thống đăng nhập một lần không có tính tập trung. Đối với những trang web có sử dụng OpenID thì người sử dụng không cần phải nhớ các thông tin về username và password cho riêng trang đó nữa. Thay vào đó họ chỉ cần đăng ký trước 1 tài khoản OpenID tại một trong những nhà cung cấp OpenID, hay thường gọi là i-broker. 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. Các hệ thống lớn đang sử dụng là : Yahoo, google, facebook…



- Open Single Sign-On (OpenSSO) hoạt đông dựa trên Token: Là một sản phẩm open source của SUN. Nó là một sản phẩm đơn lẽ, kết hợp các tính năng của Sun Java System. Access Manager, Sun Java System Fedearation Manager và Sun System SAML v2 Java Plugin.



- Java Open Single Sign On ( JOSSO).

- Central Authenticate Service (CAS) hoạt đông dựa trên Ticket.

***2.1.1 Đặc điểm của SSO trên web (Enterprise SSO)***

Một WebSSO đúng nghĩa phải là một dịch vụ chứng thực tập trung đáp ứng đồng thời các yêu cầu sau:

- Đăng nhập một lần với một tài khoản duy nhất để chứng thực vào các ứng dụng web khác nhau sử dụng dịch vụ WebSSO. Các ứng dụng web có thể chạy trên nhiều domain khác nhau. Ví dụ: khi bạn đăng nhập vào hotmail.com thì bạn sẽ không cần đăng nhập một lần nữa khi vào msn.com.

- Web SSO yêu cầu một hệ thống network ổn định và đảm bảo vận hành xuyên suốt.

- Phải sử dụng SSL và các phương thức mã hóa bất đối xứng. Nếu một dịch vụ chứng thực mà không dùng SSL cho các phiên chứng thực thì đều bị coi là không an toàn. Và đây cũng là một mục tiêu quan trọng của một ứng dụng web khi dùng dịch vụ chứng thực tập trung.

- Đăng xuất toàn bộ hệ thống SSO khi click vào button [Sign Out] hoặc tự động SignOut khi tắt trình duyệt.

Trong SSO cũng như chứng thực thông thường, đều cần database để lưu trữ thông tin người dùng. Còn SSO hoàn toàn không dùng database để lưu vết session. SSO phải đảm nhiệm một nhiệm vụ rất lớn trong việc chứng thực toàn bộ người dùng cho các ứng dụng web. Do vậy vấn đề bảo mật luôn đặt lên hàng đầu, nếu SSO gặp một sự cố nào đó thì toàn bộ các partners (ứng dụng web dùng SSO) sẽ không thể đăng nhập được.

***2.1.2 Nguyên lí hoạt động của web SSO:***

- Bước 1: Người dùng đăng nhập vào tài khỏan tại url của dịch vụ SSO Authentication Service. Sau khi đăng nhập thành công -> bước 2

- Bước 2: Authentication Service sẽ thực hiện việc sinh ra Authentication Token. Đây là một chuỗi chứa thông tin chứng thực (không chứa password) và thông thường được mã hóa bằng DES hoặc RSA.

- Bước 3: Authentication Token sẽ được truyền tải trên http (không mã hóa SSL) giữa ứng dụng web (partner) và Authentication Service. Trên mỗi ứng dụng web sẽ có một SSOAgent, có nhiệm vụ xử lý Authentication Token và kiểm tra việc chứng thực. Trong quá trình này session cookie sẽ được tạo tại trình duyệt của người dùng đăng nhập nếu SSOAgent kiểm tra Authentication Token thành công.

- Bước 4:*[Sign Out]*: Khi chứng thực tự động vào một partner, Authentication Service sẽ lưu lại thông tin partner này, khi người dùng click [Sign Out] thì Authentication Service sẽ lần lượt Sign Out (xóa toàn bộ session cookie của partners).

* 1. ***SSO trên các ứng dụng không trên nền tảng web***

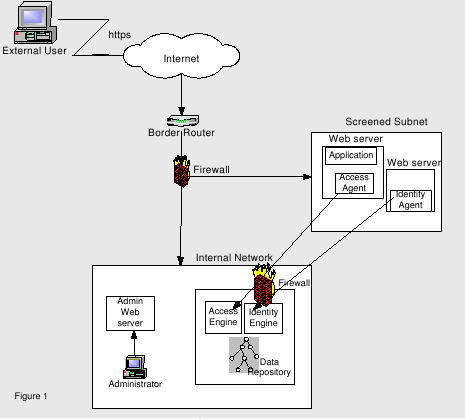
Các doanh nghiệp hiện nay thường sử dụng Microsoft Windows ® là hệ điều hành chính dành cho người dùng máy tính để bàn truy cập các ứng dụng doanh nghiệp đa dạng, ứng dụng thường có yêu cầu an ninh khác nhau và phải đăng nhập cho các dịch vụ khác nhau.

Giải pháp SSO sử dụng phổ biến nhất hiện nay là hệ thống của Microsoft sử dụng cho Windows, Microsoft Office SharePoint Server.



## Kiến trúc SSO

***3.1. Kiến trúc SSO cho các ứng dụng Internet (Architecture for Internet facing SSO)***



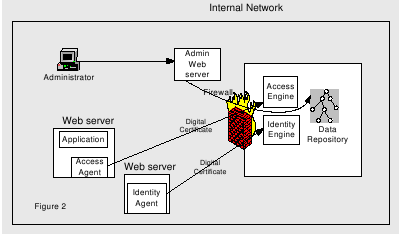
Kiến trúc này mô tả các vị trí thành phần khác nhau trong một đơn vị sử dụng các ứng dụng SSO. Ứng dụng ra ngoài Internet phải đối mặt với rất nhiều cuộc tấn công và do đó việc bảo vệ theo chiều sâu được sử dụng như một nguyên tắc trong kiến trúc SSO để bảo đảm an ninh tối đa cho các ứng dụng. Các web server và ứng dụng cho phép truy cập vào internet nên được đặt trong phân vùng riêng (DMZ). Các công cụ truy cập, quản lý máy chủ và lưu trữ dữ liệu đặt trong mạng nội bộ.

Các giao tiếp giữa người dùng và máy chủ web có thể được bảo mật bằng cách sử dụng SSL(Secure Socket Layer) tránh bị ăn cắp user đăng nhập và mật khẩu được truyền đi dưới dạng văn bản không mã hóa.

Sử dụng tường lửa chỉ cho phép các máy chủ ứng dụng giao tiếp với nhau tại các dịch vụ được chỉ định và từ chối tất cả các truy cập khác. Điều này làm giảm rủi ro các chương trình độc hại như Trojan gửi các lưu lượng giả vào các máy chủ ứng dụng từ mạng internet.

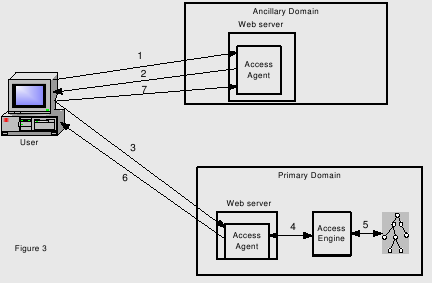
Thường xuyên cập nhật các bản vá cho các máy chủ một cách sớm nhất ngay khi các bản vá được phát hành.

***3.2. Kiến trúc SSO cho mạng nội bộ***



Tương tự như kiến trúc SSO cho các ứng dụng ra internet, nhưng tất cả các thành phần của kiến trúc được đặt trong mạng nội bộ.

***3.3. Kiến trúc SSO cho nhiều miền***



Sử dụng một miền chính và các miền phụ trợ, người dùng sẽ được xác thực chung nhưng sẽ chỉ truy cập được đến tài nguyên mà các miền ấn định chính sách riêng cho từng người dùng, nhóm người dùng cụ thể. Nếu người dùng truy cập vào các dữ liệu không được phép thì sẽ diễn ra theo các bước:

1. Người dùng sử dụng trình duyệt của mình để truy cập vào một nguồn tài nguyên được bảo vệ bởi một chính sách truy cập trong các miền phụ.

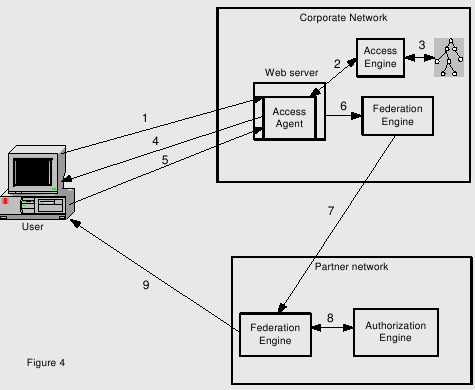
2. Miền phụ gửi một phản hồi đến người dùng để chuyển hướng đến miền chính để xác thực.

3. Truy cập miền chính qua thành phần xác thực người dùng, quyền hạn truy cập để kiểm tra và cấp phép.

4. Gửi thông tin xác thực và ủy quyền trở lại trình duyệt của người sử dụng, sau khi nhận được thông tin từ các công cụ truy cập, cookie được thiết lập trên trình duyệt của người dùng cho các tên miền chính.

Khi người dùng nhấp vào nút đăng xuất trên một trang web trong miền phụ các máy chủ web trong miền phụ sẽ hủy các cookie đặt cho người sử dụng và chuyển hướng người dùng đến miền chính với các thông tin hủy cookie người dùng. Các máy chủ web trong tên miền chính hủy các cookie của nó lưu cho người sử dụng và người sử dụng thoát khỏi ứng dụng.

***3.4. Kiến trúc SSO chéo***



Cũng được tổ chức như các SSO thông thường nhưng có thêm phần xác thực giữa các hệ thống SSO với nhau. Các bước :

- Người dùng truy cập một ứng dụng trên các trang web của ứng dụng trên hệ thống SSO của mình.

- Người sử dụng được xác thực và ủy quyền để truy cập ứng dụng và được thiết lập một cookie trên trình duyệt của người dùng.

- Người dùng sau đó nhấp chuột vào liên kết đưa người sử dụng một ứng dụng duy trì trên một trang web đối tác (hệ thống SSO khác)

- Các công cụ kết nối sử dụng cookie của trình duyệt người dùng để đóng gói nó như là một SAML (Security Assertion Markup Language) và sau đó chuyển hướng nó vào thành phần kết nối trong mạng SSO đối tác.

- Hệ thống kết nối bên SSO đối tác xác nhận SAML được gửi từ đối tác SSO đã được tin cậy, sau đó kiểm tra và cấp phép cho phép người dùng truy cập vào ứng dụng.

## Nhận xét

***Ưu điểm của giải pháp SSO:***

***Đối với người dùng:***

Hệ thống SSO cung cấp cho người dùng công cụ xác thực đăng nhập một lần. Công cụ này xác thực người dùng đăng nhập vào mọi hệ thống có tích hợp giải pháp SSO, vì vậy, người dùng chỉ cần nhớ một cặp tên truy cập/mật khẩu duy nhất.

Khi cập nhật mật khẩu và tên đăng nhập, người dùng chỉ cần cập nhật một lần cho toàn bộ các hệ thống có yêu cầu xác thực người dùng.

Khi đã đăng nhập vào một hệ thống chứa các hệ thống con có tích hợp giải pháp SSO, người dùng có thể dễ dàng truy nhập vào các hệ thống này mà không cần phải xác thực lại trong một khoảng thời gian nào đó, tùy thuộc vào chính sách quản trị.

***Đối với nhà cung cấp dịch vụ, quản trị hệ thống:***

Người quản trị chỉ cần bảo mật và quản lý thông tin đăng ký của người dùng một lần, vì vậy có thể giảm dung lượng cơ sở dữ liệu và tránh được các xung đột nảy sinh do phải xử lý mật khẩu của các hệ thống khác nhau, tăng khả năng mở rộng và triển khai các chiến lược bảo mật.

Người quản trị có thể thay đổi và cập nhật thông tin được bảo mật của người dùng khi cần thiết, một cách dễ dàng hơn so với việc thay đổi ở từng hệ thống riêng lẻ mà người dùng đó được phép truy cập. Điều này rất hữu ích khi người dùng thay đổi vị trí của mình với các cấp độ bảo mật khác nhau.

***Nhược điểm của giải pháp SSO:***

Việc xử lý một loạt các hệ thống con có tích hợp giải pháp SSO không đơn giản, bởi vì mỗi hệ thống con có thể hoạt động trên các nền phần cứng và phần mềm khác nhau. Vì vậy, khi cài đặt sẽ giải quyết nhiều vẫn đề liên quan đến tính tương thích và đồng bộ giữa các hệ thống.

Các kỹ thuật mang tính mở áp dụng với hệ thống hoặc chưa được chuẩn hóa, hoặc chưa được cung cấp, có thể gây mâu thuẫn và không tương thích với các sản phẩm khác.

Khi đã được xác thực thành công, người dùng có thể truy nhập vào nhiều ứng dụng trong hệ thống. Bởi vậy, nếu không được bảo mật tốt tại các hệ thống con, thì đối tượng tấn công khi tiếp cận vào một ứng dụng có thể tấn công vào toàn bộ hệ thống.

Thách thức đối với giải pháp SSO là làm việc theo kiến trúc, các cơ chế bảo mật độc lập đối với các nền ứng dụng. Để thuận tiện và đơn giản hơn khi giải quyết nhược điểm của SSO, người ta thường đi theo hướng xây dựng các ứng dụng sử dụng cơ sở hạ tầng bảo mật, xác thực người dùng chung, thông suốt giữa các ứng dụng. Điều này đòi hỏi phải có định dạng chung khi thể hiện các thông tin xác thực hoặc ủy nhiệm, sao cho tất cả các ứng dụng đều có thể hiểu và chấp nhận chúng. Đồng thời, phải đảm bảo được các ủy nhiệm là đúng đắn.

Mặc dù có một số nhược điểm, nhưng giải pháp SSO vẫn đang được triển khai rộng rãi ở nhiều nước trên thế giới, trong nhiều hệ thống thông tin có nhu cầu xác thực và bảo mật. Nó được đánh giá là một trong các giải pháp hiệu quả, tiện dụng và kinh tế khi triển khai trên diện rộng. Các vấn đề nêu trên sẽ được giải quyết trong thời gian tới khi chúng ta tăng cường chuẩn hóa và có các chính sách cụ thể cho vấn đề này.