PROTOCOLLI PER L'UTILIZZO DEI SERVIZI CLOUD

A. Cavaleri L.Sabatucci

Rapporto Tecnico N.: ***Data:***

RT-ICAR-PA-15-04 **Novembre 2015**

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Palermo, Viale delle Scienze edificio 11, 90128 Palermo.

Protocolli per l'utilizzo dei servizi cloud

Antonella Cavaleri and Luca Sabatucci ICAR-CNR, Palermo
Email: {a.cavaleri,sabatucci}@pa.icar.cnr.it

ABSTRACT

Il presente documento presenta una indagine sui principali protocolli per l'utilizzo dei servizi cloud. Ad ogni standard quindi corrisponde la descrizione dalla relativa architettura per l'interazione tra servizio locale e servizio remoto. In particolare si è analizzato il protocollo OAuth , utilizzato per garantire la sicurezza nell'accesso ai dati nelle applicazioni sul Cloud. Esso rappresenta uno standard di fatto dal momento in è adottato nei servizi Google e Dropbox. Infine il documento presenta i principi architetturali alla base del protocollo RESTFul per la definizione di servizi cloud.

OAuth

Il protocollo di comunicazione Open Authorization chiamato comunemente OAuth permette ad un applicazione o ad un servizio web di gestire in modo sicuro l'accesso autorizzato ai dati sensibili. Esso è stato ideato nel 2006 da Blaine Cook e si posto come alternativa ai protocolli proprietari aperti già esistenti quali Google AuthSub, AOL OpenAuth, Yahoo BBAuth, Flickr API e tanti altri. La prima versione 1.0 rilasciata nel 2007 è stata oggetto di numerose revisioni sviluppate a seguito del recepimento delle RFC(Request For Comments) proposte dai vari esperti. La versione più recente e attualmente utilizzata del protocollo è quindi la OAuth 2.0. Attraverso l'uso di questo protocollo viene garantito l'autorizzazione a terze parti a gestire documenti privati senza dover necessariamente condividere la password personale. Ciò ha notevoli vantaggi se si pensa che il dover condividere una password pone notevoli limiti sul livello di sicurezza non garantendo ad esempio dei privilegi esclusivi relative ad alcune operazioni su alcuni file o ad esempio non permette di poter revocare l'accesso se non modificando l'intera password dell'account. Tali limiti vengono superati grazie ad OAuth che garantisce un accesso delegato ad un cliente relativamente a delle risorse specifiche mantenute su un server per un tempo limitato che può anche essere alterato a causa della possibilità di revoca. Infatti la logica di funzionamento principale si basa sul fatto che viene delegata l'autenticazione dell'utente al servizio che ospita l'account dell'utente stesso è viene fornita l'autorizzazione ad applicazioni di terze parti a poter accedere all'account dell'utente. All'interno del protocollo possono essere individuati tre principali attori convolti :

- Service Provider: ovvero il servizio web che detiene le risorse personali dell'utente che è quindi in grado di fornirle a terze parti, è colui quindi che è capace di accettare e rispondere attraverso l'uso di access tokens alle richieste di utilizzo provenienti dall'esterno.
- Client: l'applicazione che richiede l'accesso alle risorse protette dell'utente per conto del proprietario della risorsa e con dopo aver ottenuto la sua autorizzazione.
- Proprietario della risorsa: utente dell'applicazione che essendo registrato ad una piattaforma del service provider vuole garantire l'accesso alle risorse personali che in esso sono presenti, esso è colui che concede l'accesso alla risorsa protetta e nel caso in cui si tratti di una persona fisica viene indicato come utente .
- Server di autorizzazione : il server che rilascia l'access tokens al client dopo che l'utente proprietario della risorsa ha effettuato con successo l'autenticazione e ha concesso l'autorizzazione al client.

Il server di autorizzazione può essere lo stesso del service provider oppure una risorsa distinta, un singolo server di autorizzazione può emettere access tokens accettati da più risorse server.

Quando un' applicazione client vuole usufruire del servizio fornitor da un Service Provider è necessario che essa effettui una registrazione, tramite questa fase iniziale il service provider fornisce il client delle chiavi segrete che saranno poi necessarie per lo scambio di dati. Nella Figura 1 viene mostrato quali i passi principali attraverso i quali avviene l'interazione tra i vari attori convolti nel protocollo.

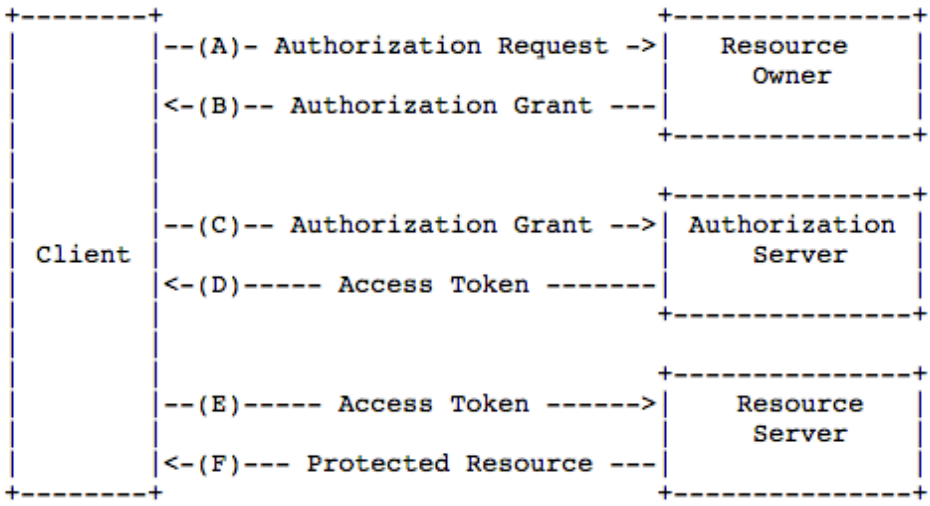

Figura 1- Flusso di interazione tra gli attori coinvolti [3]

I passi attraverso i quali il protocollo opera che vede coinvolti i vari ruoli possono essere cosi dettagliati:

(A) Un applicazione client effettua una richiesta di autorizzazione al proprietario di una risorsa. Tale richiesta può essere effettuata direttamente come mostrato in figura oppure mediata da un authorization server in maniera indiretta.

(B) Successivamente il client riceve in risposta le credenziali che rappresentano l'autorizzazione da parte del proprietario della risorsa sotto forma di un authorization grant. La specifica del protocollo definisce quattro tipi di authorization grant predefiniti che possono essere utilizzati oppure si può utilizzare un estensione. Tale tipo dipende dal metodo utilizzato dal client per richiedere l'autorizzazione e dal tipo supportato dall'authorization server.

(C) Il client inoltre deve effettuare la richiesta di un access token autenticandosi presso l'authorization server a cui fornisce l'authorization grant ricevuto nella fase precedente

(D) L'authorization server fornisce l'access token dopo aver validato e verificato l'autenticità dell' authorization grant

(E) Attraverso l'access token ottenuto il client può quindi richiedere al server che detiene la risorsa l'accesso ad essa

(F) Il server se l'access token fornito è valido processa la richiesta del client.

Nel seguito di questo paragrafo vedremo con maggiore dettaglio gli elementi principali che vengono utilizzati nell'interazione tra gli attori per il corretto funzionamento del protocollo: authorization grant e access token

Registrazione dell'applicazione

Prima di poter usare OAuth con una specifica applicazione , è necessario registrare l'applicazione con il servizio. Questo generalmente avviene attraverso un form di registrazione presente nella sezione "developer " o "API " del sito web del servizio stesso, attraverso tale form verranno fornite le seguenti informazioni circa l'applicazione:

- Nome dell'applicazione
- Sito web dell'applicazione
- Uri di re-indirizzamento o di callback, che rappresenta dove il servizio reindirizzerà l'utente dopo che esso ha autorizzato l'applicazione, è quindi la parte della nostra applicazione che permette di gestire i codici di autorizzazione o gli acccess token.

Non appena l'applicazione è stata registrata , è possibile ottenere le "credenziali del cliente" sotto la forma di un client identifier e di un client secret. Il Client ID è una stringa che può essere resa pubblica e che è usata dal servizio del server della risorse per identificare

l'applicazione, viene inoltre utilizzata per costruire un URL di autorizzazione che viene fornita all'utente. Il valore Client Secret è usato per autenticare l'identità di un applicazione nel server della risorsa del servizio quando l'applicazione richiede di accedere all'account dell'utente, tale valore po' quindi essere mantenuto provato tra l'applicazione e il server della risorsa.

Authorization Grant

Si tratta delle credenziali che rappresentano l'autorizzazione ottenuta da parte del proprietario di una risorsa e che vengono utilizzate dal client per ottenere un access token. La specifica del protocollo OAuth definisce i seguenti quattro tipi di grant authorization:

- Codice di autorizzazione
- implicita
- password del proprietario della risorsa
- credenziali del client

Inoltre è possibile definire delle tipologie aggiuntive attraverso l'uso di un estensione che il protocollo stesso supporta.

Codice di autorizzazione

Tale tipo di codice è ottenuto utilizzando un server di autorizzazione come intermediario tra il client e il proprietario della risorsa. Infatti il client ,invece effettuare una richiesta direttamente al proprietario della risorsa esso si rivolge ad un server di autorizzazione che a sua volta ottiene il codice di autorizzazione dal proprietario della risorsa e lo fornisce al client. Quindi il proprietario della risorsa si autentica solo per con il server di autenticazione in questo modo le sue credenziali non vengono condivise con il client. Ciò garantisce un elevato livello di sicurezza in quanto attraverso la capacità di autenticare il client e di trasmettergli direttamente l'access token senza dover necessariamente passare attraverso l'user –agent del proprietario della risorsa si evita l'esposizione potenziali ad altri client.

Implicito

Si tratta di una versione semplificata del flusso eseguito dal codice di autorizzazione ottimizzato per le implementazioni client implementate in un browser che usano dei linguaggi di scripting. Con un flusso implicito, anziché di fornire un codice di autorizzazione, viene rilasciato al client direttamente un access token, quindi non vengono richieste delle credenziali intermedie cosi come necessario per il codice di autorizzazione. Quando rilascia

un access token il server di autorizzazione quindi non autentica il client , tuttavia in alcuni casi l'identità del client può essere verificata attraverso un re-indirizzamento URI usato per fornire al client l'access token richiesto. L'access token può essere reso disponibile dal proprietario della risorsa o da applicazioni che hanno accesso all'user-agente del proprietario della risorsa. Questo tipo di interazione sebbene migliori notevolmente la capacità di risposta e l'efficacia di alcuni client poiché riduce il numero delle interazioni necessarie per ottenere un access token , pone alcuni limiti nei livelli di sicurezza in quanto l'access token essendo trasmesso all'interno del frammento URI , può essere potenzialmente esposto all'accesso da parte di entità non autorizzate

Credenziali del proprietario della risorsa

Le credenziali del proprietario di una risorsa (ad esempio , username e password) possono essere direttamente come un authorization grant per ottenere un access token. Questo tipo di interazione può essere usata solo se vi è un elevato grado di fiducia del proprietario della risorsa nei confronti del client e quando non sono disponibili altre tipologie di authorization grant. Anche se questo tipo di autorizzazione permette l'accesso diretto del client alle credenziali del utente, esse sono utilizzate per una singola richiesta e sono modificate per un access token. Quindi il client non deve memorizzare le credenziali per un eventuale uso futuro, ma possono essere sostituite le credenziali con un access token di lunga durata o con un aggiornamento del token.

Credenziali del cliente

Le credenziali del client possono essere utilizzate come un authorization grant quando l'ambito di autorizzazione è limitato a delle risorse che sono sotto il controllo del client stesso, oppure è stata concessa precedentemente da parte di un server di autorizzazione l'accesso a delle risorse protette. Questo tipo di interazione è generalmente usata quando il client è anche il proprietario delle risorse oppure quando la richiesta di accesso alle risorse protette è basata su un'autorizzazione precedentemente concordate con il server di autorizzazione.

Access token

Gli access tokens sono le credenziali utilizzate per accedere alle risorse protette. Si tratta di una stringa rilasciata al client . il contenuto di tale stringa è generalmente poco chiara per il client. I vari tokens rappresentano specifici campi e tempi di accesso concessi dal proprietario della risorsa ed eseguiti dal server della risorsa e dal server di autorizzazione. Il token può

indicare un identificativo utilizzato per recuperare le informazioni di autorizzazione, oppure possono contenere al loro interno tali informazioni necessarie in una modalità verificabile (ad esempio associando una firma alla stringa di dati contenuta). Un access token fornisce un livello di astrazione elevato in quanto i diversi costrutti di autorizzazione (ad esempio username e password) vengono sostituiti con un singolo token che è interpretato e compreso dal server delle risorse. Tale astrazione permette di concedere degli access token più restrittivi rispetto all'authorization grant usata per ottenerli, inoltre ciò elimina la necessità da parte del server delle risorse di comprendere una vasta gamma di metodi di autenticazione. Gli access token possono avere differenti formati, strutture e metodi di utilizzo a seconda dei requisiti di sicurezza del server delle risorse.

Refresh token

I refresh token sono credenziali usati per ottenere gli access token. Questo tipo di token sono rilasciati al client da un authorization server e sono usati per ottenere un nuovo accesso quando l'access token corrente è considerato invalido o è scaduto., oppure al fine di ottenere ulteriori access token con un ambito uguale o più ristretto. L'emissione di un refresh token è facoltativa e a discrezione dell' authorization server. Qualora esso venga rilasciato dal server è incluso nell'access token relativo fornito. Il refresh token è una stringa che rappresenta l'autorizzazione concessa al client dal proprietario della risorsa. Il token denota un identificativo usato per recuperare le informazioni di autorizzazione. A differenza dell'access token , il refresh token sono destinati ad un uso esclusivo con gli authorization server e non sono mai inviati ai server delle risorse.

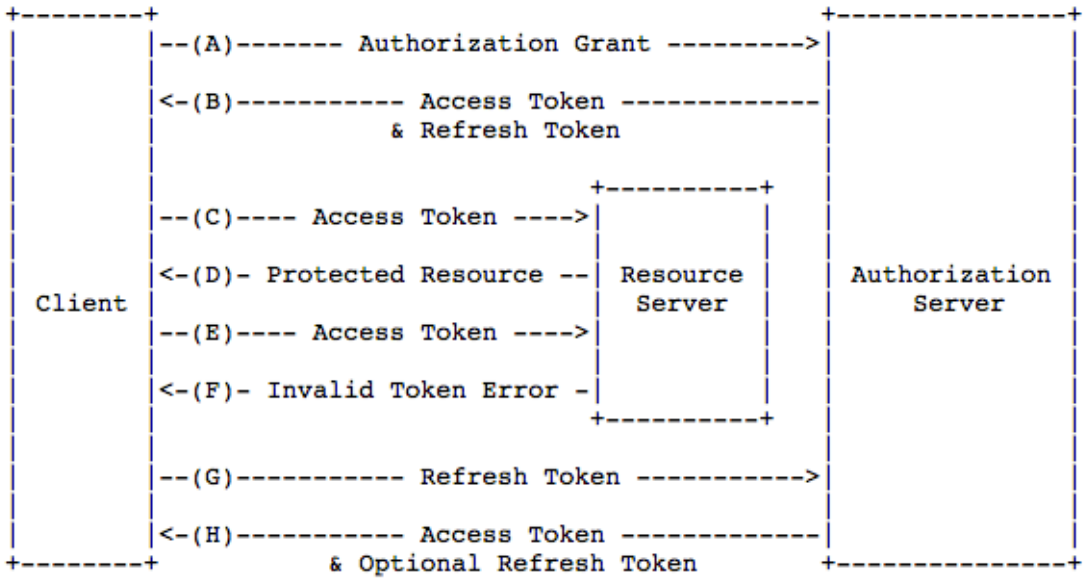

Figura 2 - Aggiornamento di un access token scaduto [3]

Per comprendere meglio il flusso che viene eseguito quando deve essere utilizzato un access token si consideri la Figura 2 ; come si può vedere dopo che il client ha avanzato una richiesta di authorization grant ottiene un access token e il relativo refesh token. Ogni volta che il client richiede una risorsa protetta passa al server della risorsa l'access token (C)ricevuto il quale se è correttamente validato dal server permette al cleint di ottenre la risorsa richiesta (D). tali passi vengono ripetuti fino a quando l'access token passato non è scaduto. Non appena il client è messo al corrente del verificarsi di tale situazione (F), allora esso richiederà un nuovo access token all'authorization server al quale passa il refresh token posseduto che gli permette di ottenere un nuovo access token valido per la risorsa considerata e può cosi avviarsi una nuova sezione di richieste con il server delle risorse.

Utilizzo di OAuth 2.0 per accedere alle API di Google

Le API google utilizzano il protocollo OAuth 2.0 per realizzare l'autenticazione a l'autorizzazione. Per poter utilizzare le Api di google è infatti necessario che ottenere inizialmente le credenziali client OAuth attraverso la Google Developers Console. A questo punto può essere richiesto un access token fornito dal Google Authorization Server, estratto tale token dalla risposta eesso può essere inviato alla Google API alla quale si intende accedere. Tutte le applicazionei che accedono alle Api di google usanto il protocollo OAuth 2.0 operano secondo i seguenti quattro passi:

1. ottenere le credenziali OAuth 2.0 : attraverso la Google Develeopers Console vengono forniti un clientID e un client secret noti sia a Google che all'applicazione considerata. L'insieme di tali valori cambia a seconda della tipologia di applicazione che si intende creare.
2. Ottenere un access token dal Google Authorization Server: prima che l'applicazione che si intende realizare possa accedere ai dati privati usando le API di Google è necessario ottenere l'access token che permetta di accedere a tali API. Un singolo access token può fornire diversi permessi di accesso a differenti APIs. Attraverso il parametro *scope* viene realizzato il controllo sull'insieme di risorse e operazioni che sono permesse da uno specifico access token. Durante la fase di richiesta dell'access token, l'applicazione infatti invia uno o più valori all'interno del parametro *scope.* Alcuni richieste di servizi Google richiedono una fase di autenticazione attraverso la quale l'utente deve effettuare il login nel suo account Google. In seguito al login viene chiesto all'utente di concedere all'applicazione i permessi che essa richiede. Tale processo chiamato consenso dell'utente se non va a buon fine genera un errore, mentre se l'utente

fornisce il permesso verrà inviata dal Google Authorization Server un access token all'applicazione considerata

3. Invio dell'access token all'API: dopo che un applicazione ha ottenuto un access token, tale token va inviato a Google API tramite un http authorization header. È anche possibile inviare i tokens come parametro nella stringa di richiesta URI, ma ciò non è raccomandato poiché non è completamente sicuro. Bisogna sottolineare come l'access token sia valido solo per l'insieme delle operazioni e delle risorse descritte nello scope di una richiesta di un token.
4. Riaggiornare il token: poiché gli access token hanno un tempo di vita limitato, qualora l'applicazione considerata necessita di accedere ad un api di google oltre il tempo di vita di un access token allora può ottenere un refresh token attraverso il quale è possibile ottenere un nuovo access token.

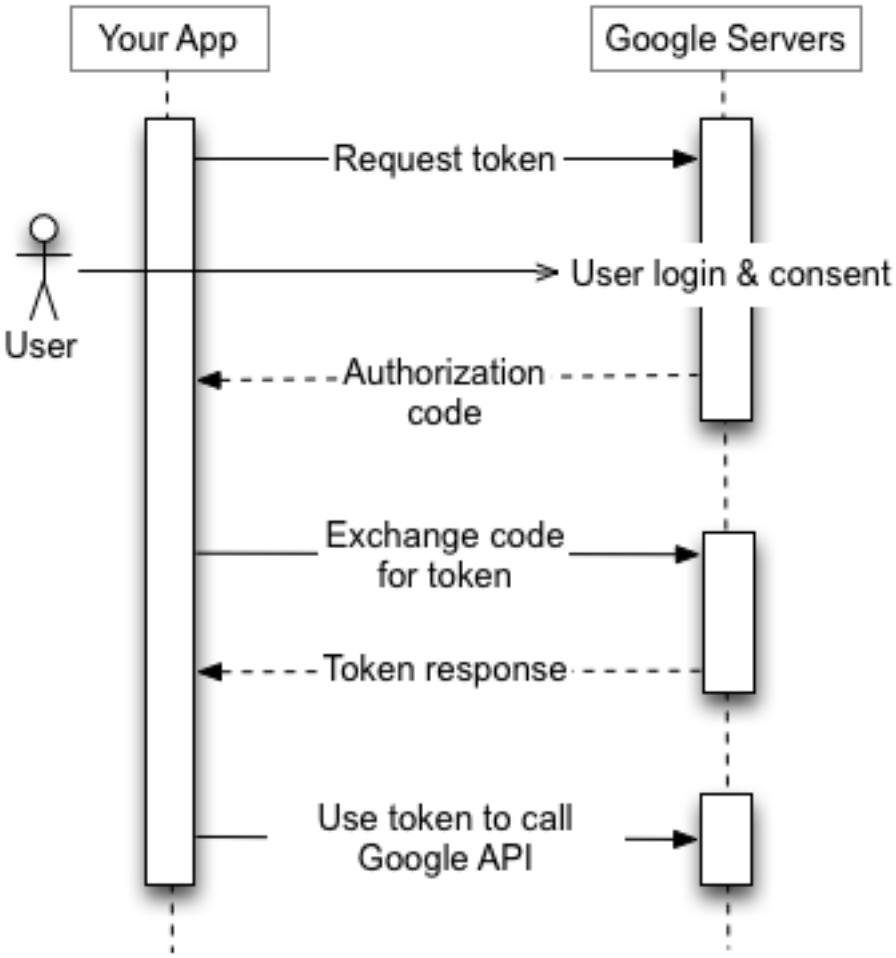

Figura 3 - Scenario di utilizzo del Protocollo OAuth 2.0 nell Google API [1]

Un access token può smettere di funzionare per uno dei seguenti motivi:

- L'utente ha revocato l'accesso precedentemente concesso

- Il token non risulta più utilizzato da oltre sei mesi
- L'account dell'utente ha superato un certo numero di richieste di token

Il limite attuale per ciascun account utente è di 25 token.

Dropbox e OAuth 2.0

Il protocollo OAuth viene utilizzato quando si intendono utilizzare le APIs di Dropbox per accedere al servizio Dropbox per conto di un utente. È necessario quindi avere per ogni utente dell'applicazione una fase di autenticazione tramite il servizio Dropbox al fine di verificare la loro identità e poter ottenere il permesso per l'applicazione considerata di accedere allo spazio cloud dropbox dell'utente. La fase iniziale per poter utilizzare le Api di Dropbox necessita che venga registrata l'applicazione che si intende realizzare con Dropbox, ciò può essere fatto creando una nuova app tramite la finestra App Console (Figura 4). Attraverso l'app console l'utente è guidato nella definizione dei permessi necessari per l'applicazione considerata e di tutte le informazioni utili, dopo aver specificato il nome associato all'applicazione.

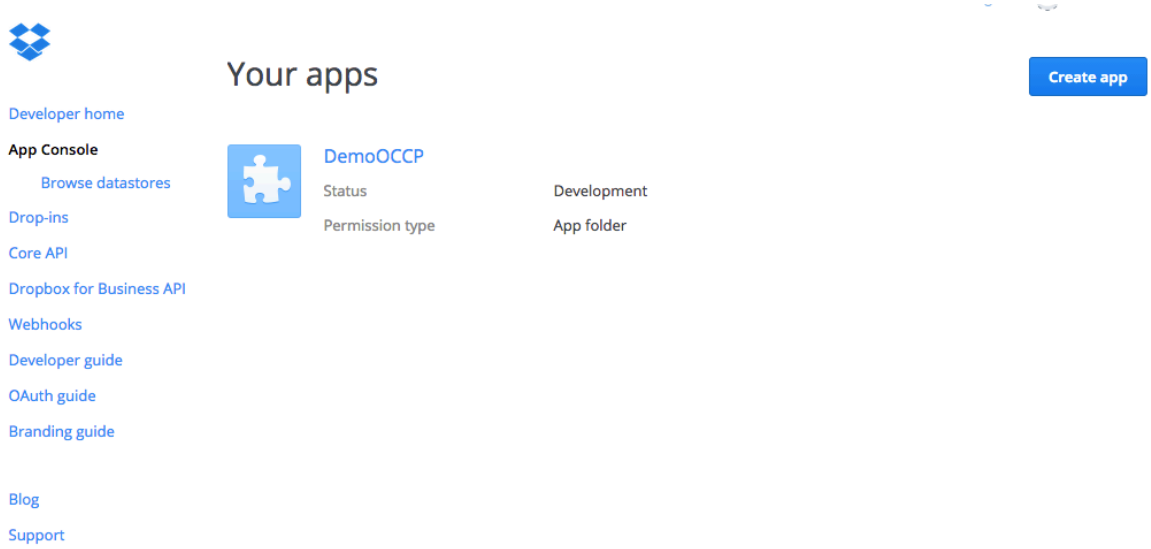

Figura 4 - Registrazione di una nuova applicazione [8]

Se si intende realizzare un applicazione web il primo passo all'interno del processo OAuth è ridirigere l'utente in una pagina web di Dorpbox. Generalmente ciò si ottiene imponendo all'utente di cliccare du un bottone che effettua il collegamento con Dropbox, l'applicazione avrà bisogno di reindirizzare l'utente verso una particolare URL di autorizzazione di Dropbox. Tale URL di autorizzazione è specifica per l'applicazione considerata ed è composta dalle seguenti informazioni:

- Chiave dell'applicazione
- URI di re-indirizzamento
- Tipo di risposta
- Stato

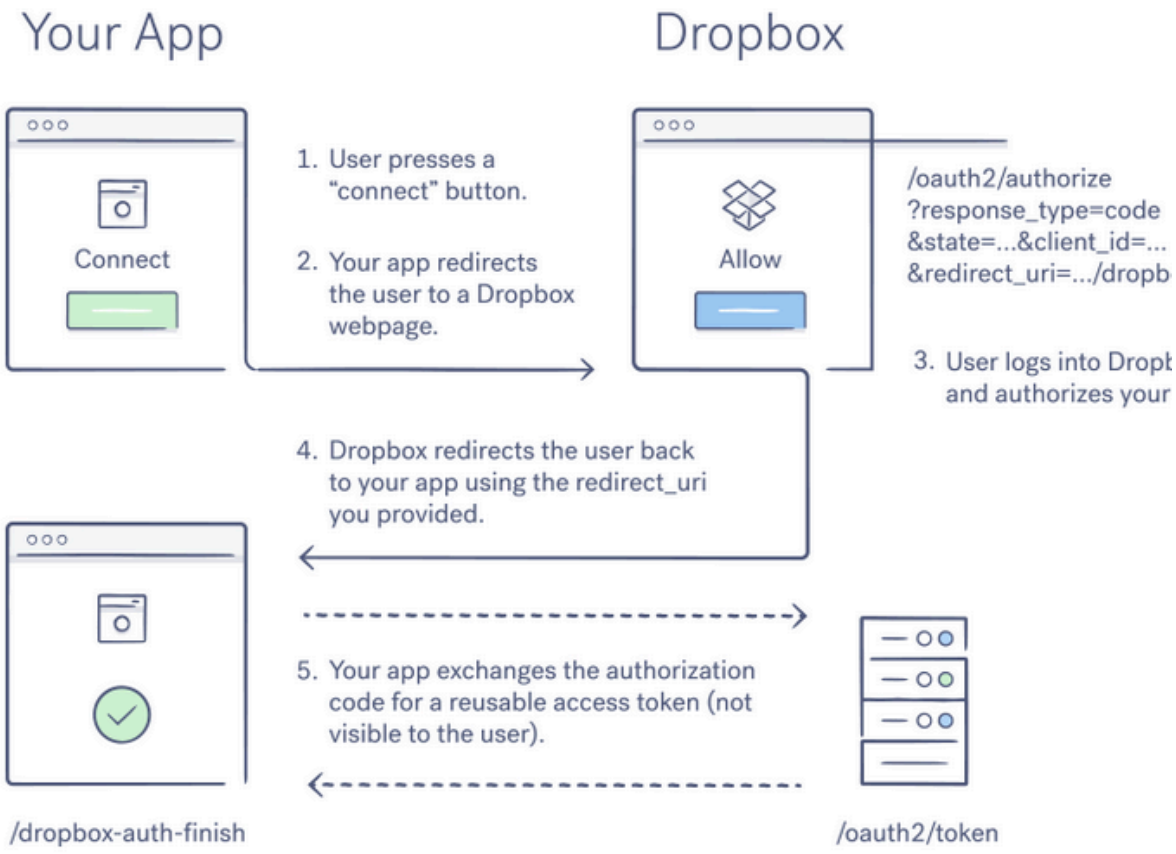

Figura 5 - Flusso di esecuzione del protocollo OAuth in Dropbox [9]

Ad esempio: https://www.dropbox.com/1/oauth2/authorize?client_id=...&redirect_uri=. Ciascuno delle SDKs di Dropbox contiene un metodo che può aiutarci a generare tale URL quando è necessario. Dopo aver generato l'authorization URL è richiesto all'utente di effettuare il login o di creare un account dropbox, non appena l'utente inserisce le credenziali personali gli verrà mostrata una finestra attraverso la quale viene richiesto ad esso se si vuole

concedere all'applicazione in questione l'autorizzazione ad accedere ai suo dati presenti su dropbox. Non appena esso fornisce il consenso verrà re-indirizzato da dropbox nuovamente verso l'applicazione utilizzando la redirect URI contenuta nella Dropbox authorization URL. Per sicurezza dropbox ammette come possibili sono le redirect URI che sono state definite precedentemente nell'App console dell'applicazione. Infatti se si tenta di effettuare un re-indirizzamento attraverso un URI che non è stata per quell'applicazione, verrà mostrato uno specifico messaggio di Dropbox al fine di impostare la corretta URI. Attraverso il redirect verso l'applicazione, Dropbox fornisce un authorization code, l'applicazione dovrà effettuare una richiesta a Dropbox per poter trasformare l'authorization code in un access toke riusabile. Tale richiesta non è visibile agli utenti finali ed avviene tramite una chiamata ad un token Api endpoint, la struttura dell'URL è la seguente: https://api.dropboxapi.com/1/oauth2/token che necessita di due parametri obbligatori :

- Code : ovvero il codice di autorizzazione che è stato precedentemente ottenuto attraverso il re-indirizzamento dell'utente all'URL: /oauth2/authorize?response_type=code.
- Grant_type: il tipo di grant che deve essere impostato come *authorization_code*

Mentre i parametri opzionali che posso essere presenti sono:

- Client_id : corrispondente al valore dell'app's key presente nella console dell'applicazione
- Client_secret: corrispondente al campo app's secret generato per l'applicazione e presente nella pagina di amministrazione della stessa (App Console)
- Redirect_uri: : una stringa contenente l'uri usata per validare se corrisponde a quella originale impostata per l'applicazione

L'invocazione di tale servizio permette di ottenere come valore di ritorno un file Json che contiene i valori per tre elementi chiave : access_token (il valore dell'access token) token_type (il tipo di token , di solito impostato a bearer), uid (l'user ID di Dropbox)

OpenID

OpenID è un semplice strato di identificazione che si pone sopra il protocollo OAuth. Permette ad un Client di verificare l'identità di un end-user basandosi su un sistema di autenticazione realizzato da un server di autorizzazione, al fine di ottenere le informazioni di base del profilo circa l'end user in modo interoperabile basato sull'invocazione di servizi REST.

RESTFul

REST, Representational State Transfer è uno stile architetturale per i sistemi software distribuiti. Tale termine introdotto e definito nel 2000 da Roy Fielding, uno dei principali autori del protocollo http. Attraverso tale stile vengono definiti una serie di principi architetturali per la progettazione di Web Service. L'idea alla base del REST è quella di vedere il Web o qualsiasi sistema come una piattaforma per l'elaborazione distribuita di dati. In tal senso quindi i servizi realizzati attraverso la tecnologia RESTful possono rappresentare i componenti critici principali per qualsiasi servizio cloud. Infatti quando si costruiscono servizi per il cloud generalmente essi sono costruiti sopra una struttura di tipo Iaas (Infrastructure as a Service) oppure come PaaS (Platform as a Service) provviste ed integrate con uno o più SaaS (Software as Service) forniti. La maggior parte dei servizi cloud open e proprietari espongono le loro API utilizzazndo dei servizi RESTFul. Inoltre essendo il cloud un ecosistema eterogeneo che connette differenti servizi cloud appartenenti a differenti compagnie e scritti utilizzando differenti tecnologia, risulta necessario come debba essere ridotta la complessità dei protocolli sottostanti in modo da astrarre la logica di business al fine di permettere a tale ecosistema di servizi di interagire in maniera semplice. Inoltre è bene sottolineare come l'utilizzo del protocollo RESTFul garantisca agli utenti di poter utilizzare un determinato servizio attraverso l'utilizzo di differenti dispositivi (web, mobile,tablet, ecc..). Considerando che l'infrastruttura cloud è virtuale e dinamica, ovvero nuove risorse vengono continuamente inserite mentre altre possono essere rimosse dall'infrastruttura, è necessario che in un infrastruttura cloud si faccia a meno dello stato, attraverso i servizi RESTFull si riesce in effetti a garantire ciò perché sfruttano l'ipermedia come un motore di monitoraggio dello stato. Infatti lo stato è rappresentato da una serie di link che (URIs) che permettono si indirizzare quindi le risorse cloud ottenendo le loro istruzioni dall' URI e processando la richiesta. Lo stile architetturale REST è costituito da client e servers. I client avviano una richiesta verso i servers i quali processano tali richieste e ritornano le risposte basandosi su tali richieste. Le richieste e le risposte sono costruiti attorno al trasferimento di rappresentazioni di queste risorse. Una risorsa può essere qualsiasi concetto coerente e significativo che può essere indirizzata, mentre una rappresentazione di una risorsa è un documento che cattura lo stato previsto di una risorsa. Fondamentalmente in REST ogni risorsa viene identificato con un URL e viene creata una nuova risorsa per ogni servizio richiesto. I dati restituiti dal servizio possono essere collegate

agli altri dati, creando quindi a una rete di informazioni. Lo stile architetturale REST si basa sui seguenti principi fondamentali:

- identificazione delle risorse
- utilizzo esplicito dei metodi http
- risorse autodescrittive
- collegamenti tra le risorse
- comunicazione senza stato

Come è stato precedentemente accennato qualsiasi elemento che è oggetto di elaborazione viene identificato in maniera univoca tramite un URI, il beneficio principale nell'adottare tale schema consiste nel fatto che in ambito Web esso risulta già ben definito e collaudato, essi inoltre risultano generalmente abbastanza auto esplicativi. Le principali regole che bisogna tenere presente nella definizione di un URI sono:

- utilizzare i nomi piuttosto che i verbi
- contenere la lunghezza di un URI
- preferire un percorso con una struttura gerarchica piuttosto che la presenza di più argomenti concatenati (es *http://www.myapp.com/client/1990/12/01*)
- non utilizzare estensioni che mostrano la tecnologia con cui è stato implementato il servizio

I metodi che il protocollo http fornisce e che permettono di effettuare delle operazioni sulle risorse presenti nel Web sono: GET,POST,PUT e DELETE. Nel contesto RESTful è possibile individuare una corrispondenza univoca tra le tipiche operazioni CRUD(create, read , update e delete) e i metodi forniti da http ciò permette quindi di poter effettuare su una risorsa tutte le principali operazioni.

Tenendo presente come un approccio RestFull le risorse cono concettualmente separate dalle rappresentazioni che sono restituite dal client, sebbene con REST non viene posto alcun vincolo relativo alla modalità di rappresentazione della risorsa e potrebbe quindi essere utilizzato il formato che si preferisce senza dover necessariamente seguire uno standard, è bene tuttavia al fine di semplificare le interazioni con il client adottare standard o fornire rappresentazioni multiple di una risorsa per soddisfare client di tipo diverso. Tramite il tipo *MIME* presente nella risposta http viene indicato il tipo di rappresentazione inviata dal Web Service, a sua volta il client può richiedere che una risorsa venga inviata in uno specifico formato definendo ciò nell'attributo *Accept.*

Il collegamento delle varie risorse presenti nel web avviene tramite link ipertestuali, tale principio è noto come HATEOAS (*Hypermedia As The Engine Of Application State*) e evidenzia quale debba essere la modalità di gestione dello stato dell'applicazione. Ovvero tutte le informazioni di cui ha bisogno un client su una risorsa sono contenute nella sua rappresentazione o deve poter essere accessibile tramite collegamento ipertestuale alle risorse ad esso collegate.

Le interazioni tra il client ed il server avvengono attraverso una comunicazione stateless. Infatti la gestione dello stato qualora necessari avviene sul client, ciò permette di migliorare la prestazioni del server il quale non deve prendersi cura dello stato di una sessione. Tuttavia ciò non deve far erroneamente intendere che i servizi RESTful siano completamente senza stato, ma bisogna sottolineare come lo stato da prendere in considerazione sia quello delle risorse, dato dall'insieme dei valori delle caratteristiche di una risorsa in un dato momento, quello del client, rappresentato dall'insieme del contesto e delle risorse ottenute in uno specifico momento, e infine quello relativo all'applicazione cioè che risulta dall'interazione tra client e server, attraverso il quale si possono determinare le modalità di modifica dello stato delle risorse e del client.

RIFERIMENTI

1. OAuth2- https://developers.google.com/identity/protocols/OAuth2
2. OAuth2- http://oauth.net/2/
3. OAuth2-https://tools.ietf.org/html/rfc6749
4. OPenId-http://openid.net/
5. OPenId -https://openid.net/get-an-openid/what-is-openid/
6. RestFul-http://searchsoa.techtarget.com/definition/REST
7. RestFul- http://rest.elkstein.org/
8. Dropbox Developer Console- https://www.dropbox.com/developers/apps
9. Dropbox Developer OAuth Guide https://www.dropbox.com/developers/reference/oauth-guide
10. Google Developers -https://developers.google.com/+/web/api/rest/oauth?hl=it