OAuth

Pier Paolo Pittavino

Definizione 0.1: OAuth

OAuth è un protocollo di rete (o meglio un framework di autorizzazione) aperto e standard, progettato specificamente per lavorare con HTTP.
Essenzialmente consente l'emissione di un token di accesso da parte di un server autorizzativo ad un client di terze parti, previa approvazione dell'utente proprietario della risorsa cui si intende accedere

Caratteristiche

Questo meccanismo, detto di *"access delegation"* (delega di accesso):

1. consente all'utente di condividere informazioni circa il proprio account con altre applicazioni o siti web
2. consente ad applicazioni ed ai siti web l'accesso alle proprie informazioni senza fornire loro alcuna password
3. garantisce ai *service provider* l'accesso da parte di terzi (*consumer*) ai dati degli utenti proteggendo contemporaneamente le loro credenziali

OAuth (1.0)

- **Motivazioni:** garantire l'accesso delegato ad un client specifico per determinate risorse sul server per un tempo limitato, con possibilità di revoca
- **Limiti:**
  1. complicato
  2. presenta problemi intrinseci di sicurezza

OAuth (2.0)

La versione 2 (RFC 6749) di questa specifica è più semplice delle precedenti, ma presenta limiti analoghi:
Nelle implementazioni più comuni fa uso di uno (o due) tokens:

- **access token:** utilizzato come un *API key* o più spesso un *Bearer Token* consente all'applicazione che lo utilizza di accedere ad i dati dell'utente ed eventualmente può avere una scadenza
- **refresh token:** utilizzato opzionalmente per richiedere nuovi token di accesso quando questi sono scaduti (parla solo con l'*Authorization Server* e non con il *Resource Server*)

OAuth 2.0 combina *autenticazione* ed *autorizzazione* ed è la scelta più matura per identificare utenti e garantire i permessi corretti

OAuth (2.0): building block

oAuth 2 divide i ruoli e prevede per ognuno un compito ben definito:

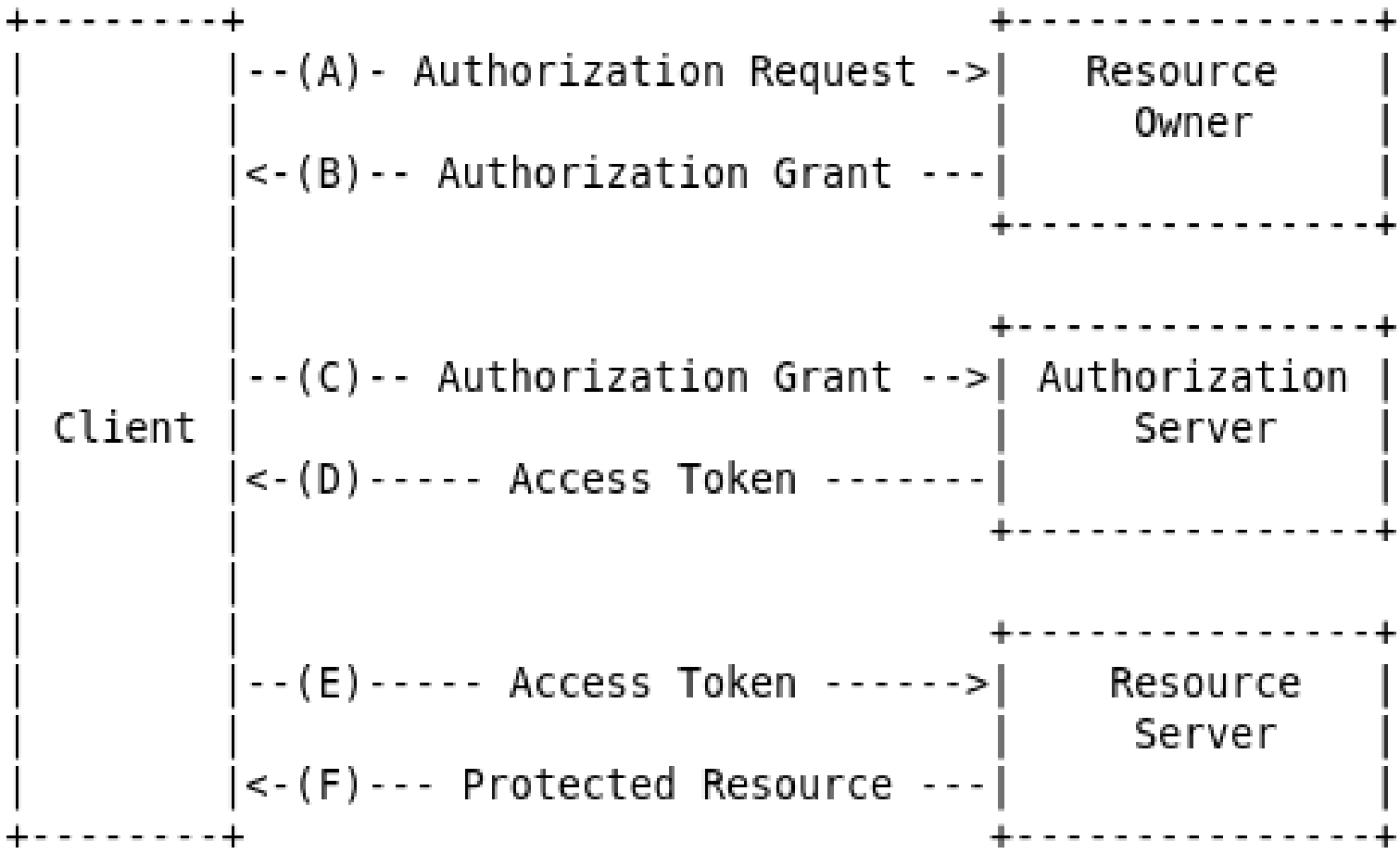
- **Resource Owner**: è il proprietario dell'informazione esposta via HTTP (l'utente) che può garantire l'accesso alle risorse protette presenti sul*"Resource Server"*
- **Client o Consumer**: è l'applicazione che richiede l'accesso alla risorsa HTTP al *Resource Owner*
- **oAuth 2 o Authorization Server**: è il modulo che firma e rilascia i token di accesso e, se necessario, richiede la login al *Resource Owner*, può coincidere con il *Resource Server*
- **Resource Server**: è il server utilizzato per accedere alle risorse protette del *Resource Owner* che detiene l'informazione esposta via HTTP

OAuth (2.0): funzionamento di massima

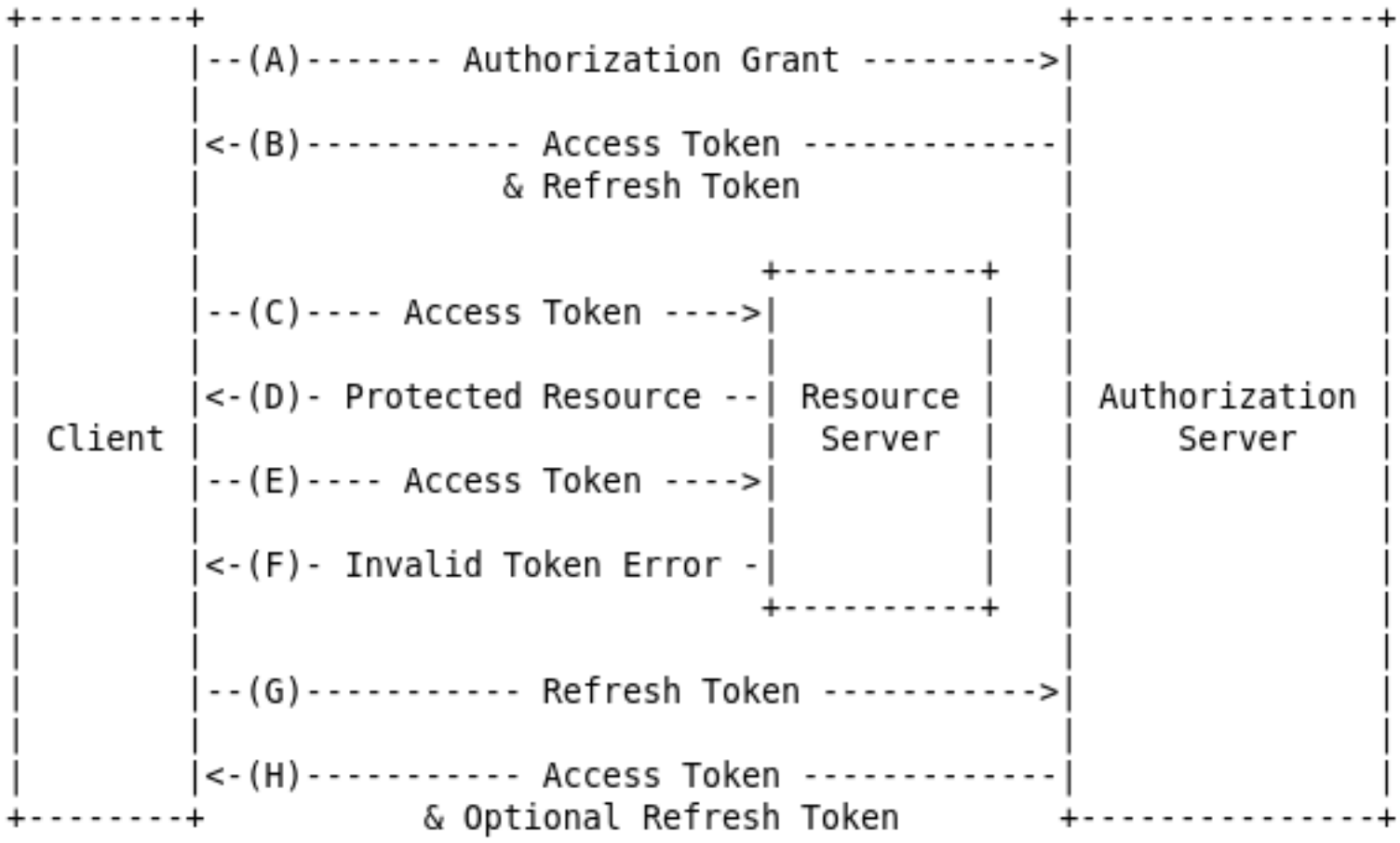
A livello astratto:

1. Il *Client* richiede l'autorizzazione direttamente al *Resource Owner* o mediante l'intermediazione di un *Authorization Server*
2. il *Client* riceve un *authorization grant*(in 1 dei 4 modi possibili in base a quelli supportati dall'*Authorization Server*)
3. Il *Client* richiede un *access_token* all'*Authorization Server* autenticandosi con l'*authorization grant*
4. Una volta ottenuto l'*access_token* può accedere e interagire con le risorse stabilite per un breve periodo, e quando scaduto ottenere un nuovo token con il *refresh token*

Abstract Protocol Flow



Refreshing an Expired Access Token



Access Token Scope

Gli **scope** possono essere considerati come una lista di permessi:

- Il *Client* può richiedere degli *scope* specifici nella richiesta dell'*access_token*, altrimenti l'*Authorization Server* o fallisce o a una lista di *scope* predefiniti
- L'*Authorization Server* può ignorare o tenere conto solo in parte di questa richiesta
- Se gli *scope* concessi sono dieversi da quelli richiesti l'*Authorization Server* **deve** includere gli *scope* concessi nella *response*

OAuth (2.0): tokens

Solitamente si utilizza un access token (e un refresh token) autoreferenziale, ovvero firmato digitalmente:

- può essere validato dal Resource Server senza dover chiedere una verifica all'Authorization Server
- La chiave privata è in possesso solo dell'Authorization Server
- la chiave pubblica è in possesso del/dei Resource Server

OAuth (2.0): grant type

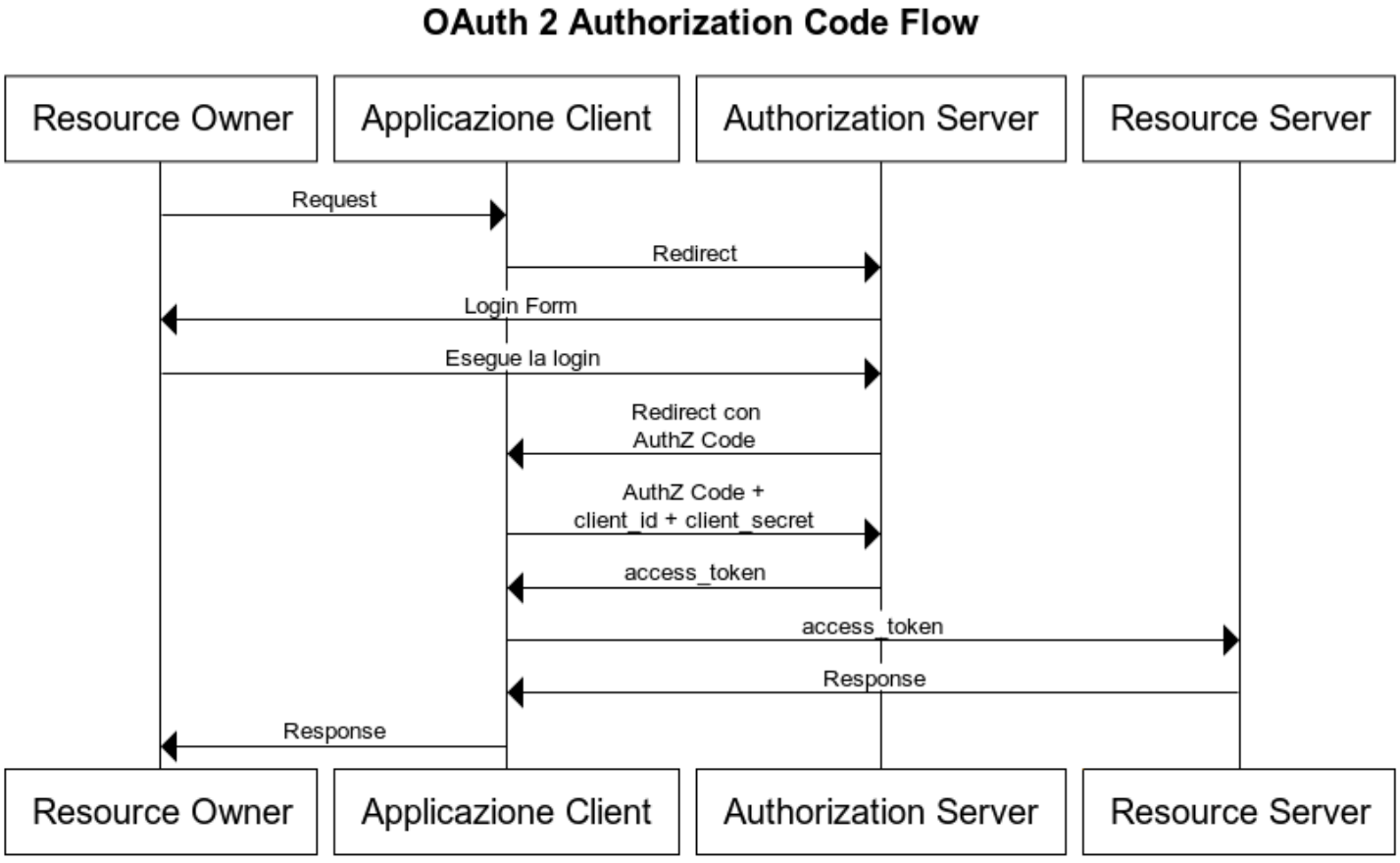
Un Grant Type è il processo da seguire per ottenere il cosiddetto Authorization Grant, ovvero la prova inoppugnabile dell'avvenuta autorizzazione da parte del Resource Owner, il titolare dell'informazione, a cui l'applicazione Client (colei che chiede l'accesso alla risorsa ad esempio un app) sta cercando di accedere. 4 diversi tipi:

1. Authorization Code
2. Implicit Code
3. Resource Owner Password Credentials
4. Client Credentials

1. Authorization Code Grant Type

1. Il Resource Owner accede all'applicazione Client
2. L'applicazione Client esegue il redirect all'Authorization Server (oAuth 2 Server)
3. L'Authorization Server chiede al Resource Owner di autenticarsi
4. A valle della login andata a buon fine, l'Authorization Server consegna all'applicazione Client l'Authorization Code
5. L'applicazione Client riconsegna all'oAuth 2 Server l'Authorization Code appena ricevuto, unitamente a client_id e client_secret (potremmo definirle l'username e la password dell'applicazione Client)
6. L'Authorization Server consegna all'applicazione Client un access_token per consentire all'applicazione Client di venire autorizzato presso il Resource Server

1. Authorization Code Grant Type Flow



1. Authorization Code Grant Type

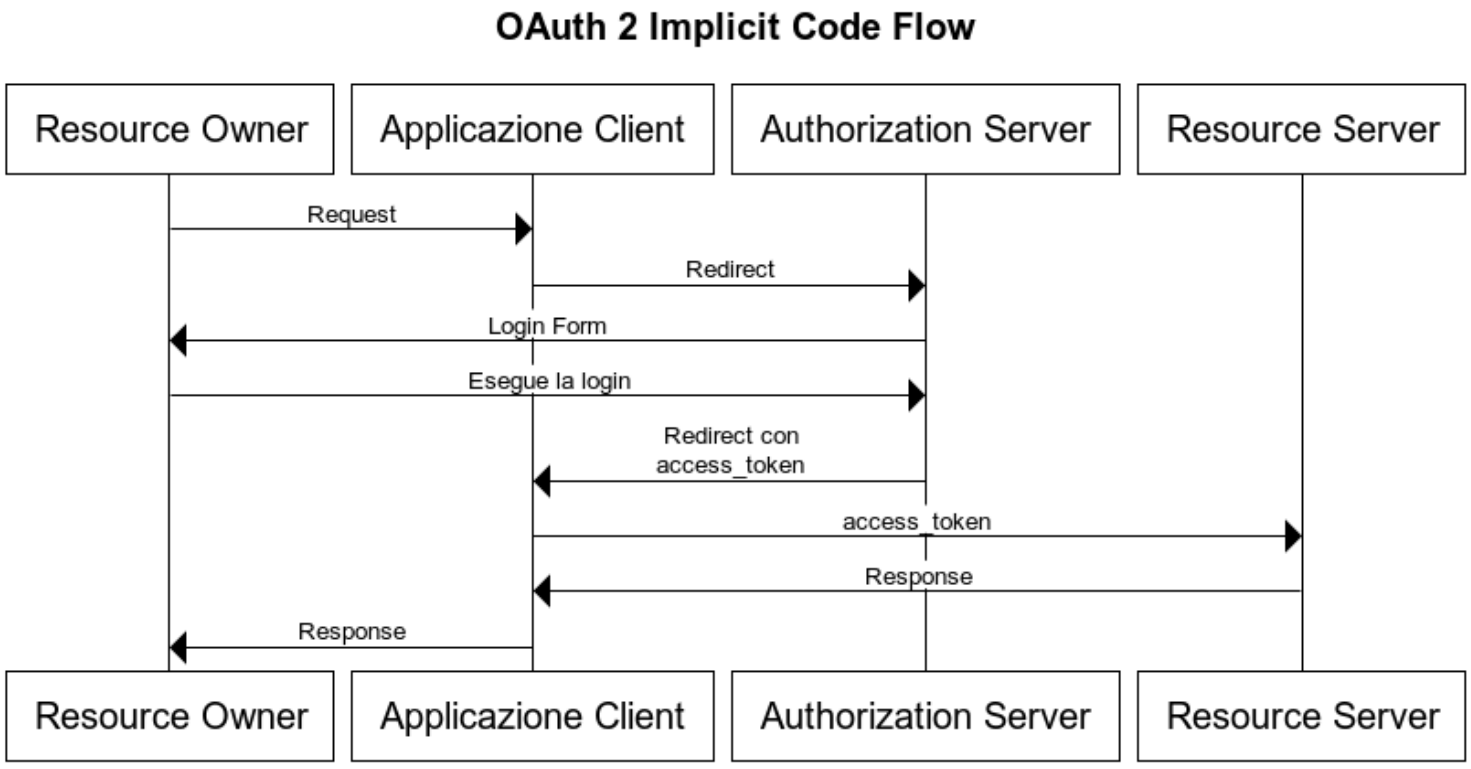
Il *Resource Server* non ha bisogno di contattare l'Authorization Server per verificare la validità dell'*access_token* perché solitamente il token è autoreferenziale, ovvero firmato con crittografia asimmetrica dall'*Authorization Server* (JWT).
Il *Resource Server* deve quindi avere la chiave pubblica dell'Authorization Server per validare il token
Lo scambio tra *Authorization Code* e *access_token*, permette di tenere segreto (scambiato tra i server)*access_token*

2.Implicit Grant Type

Minore sicurezza, maggiore semplicità, sconsigliato:

- l'*access_token* non viene mantenuto segreto
- permette di saltare un passaggio e far vivere questo colloquio completamente lato client

2.Implicit Grant Type Flow

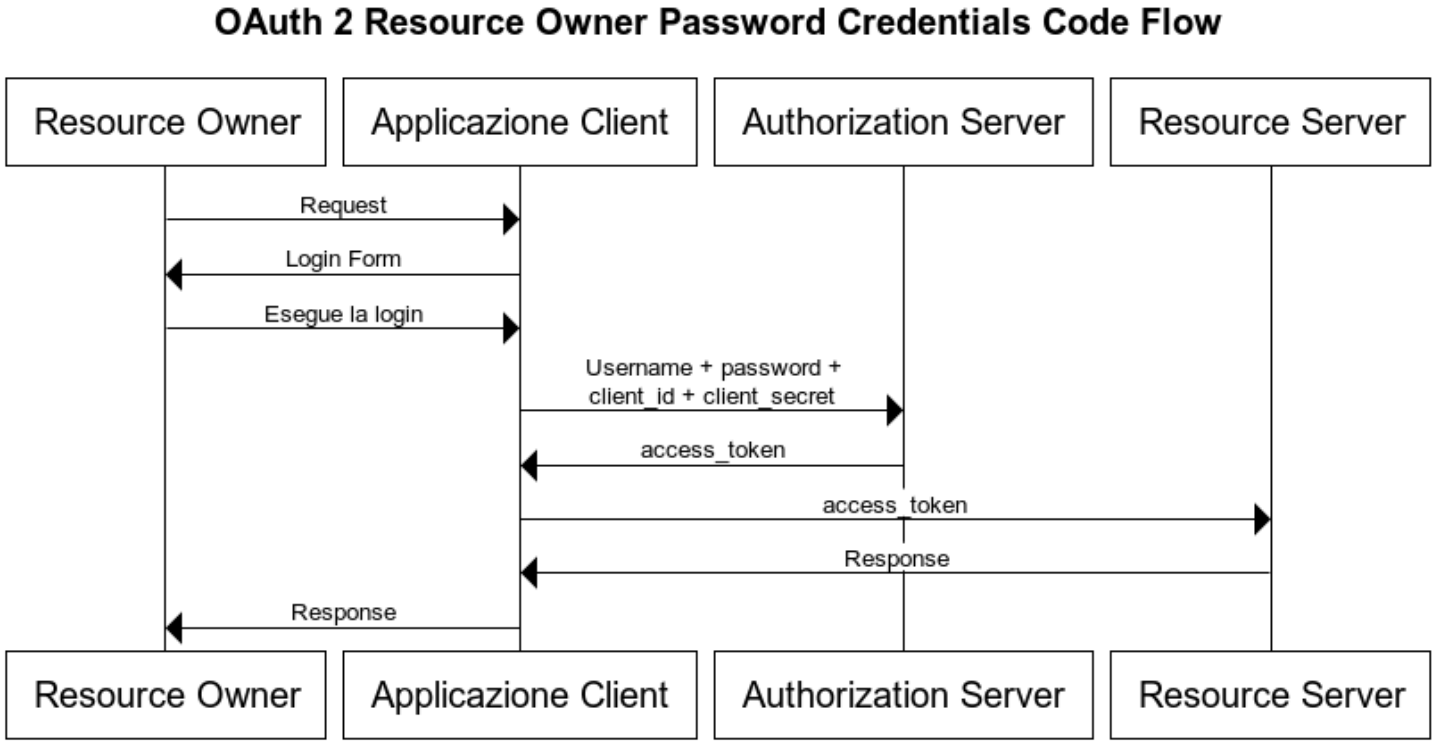


3.Resource Owner Password Credentials

Sconsigliato:

- le credenziali utente transitano dall'applicazione Client
- questo va assolutamente evitato
- da usarsi esclusivamente se c'è assoluta fiducia tra *Resource Owner* e *Client*

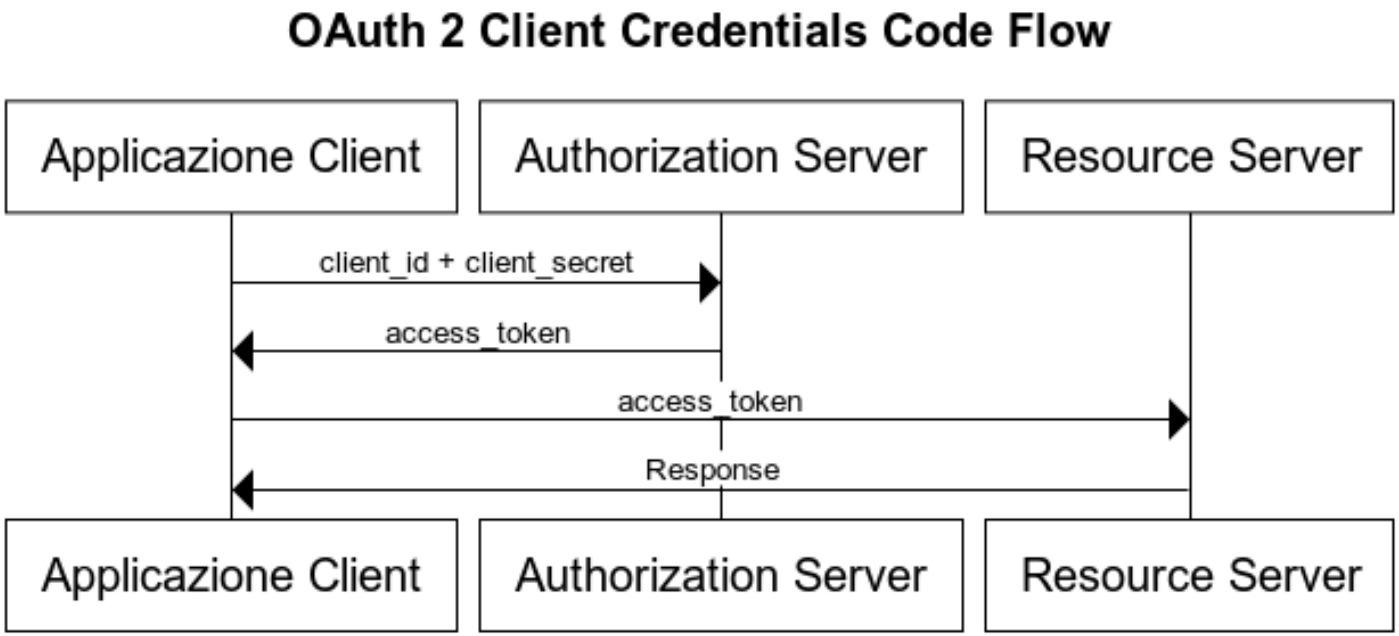
3.Resource Owner Password Credentials Flow



4.Client Credentials Grant Type

Si utilizza quando un'applicazione server side richiede l'accesso ad una risorsa HTTP senza un *Resource Owner*, quindi senza che una persona reale stia usando l'applicazione Client
Le credenziali *client_id* e *client_secret* (che sono lo username e la password dell'applicazione) devono essere nascoste e non disponibili sul browser =¿ l'applicazione deve essere client side e non server side

4.Client Credentials Grant Type Flow



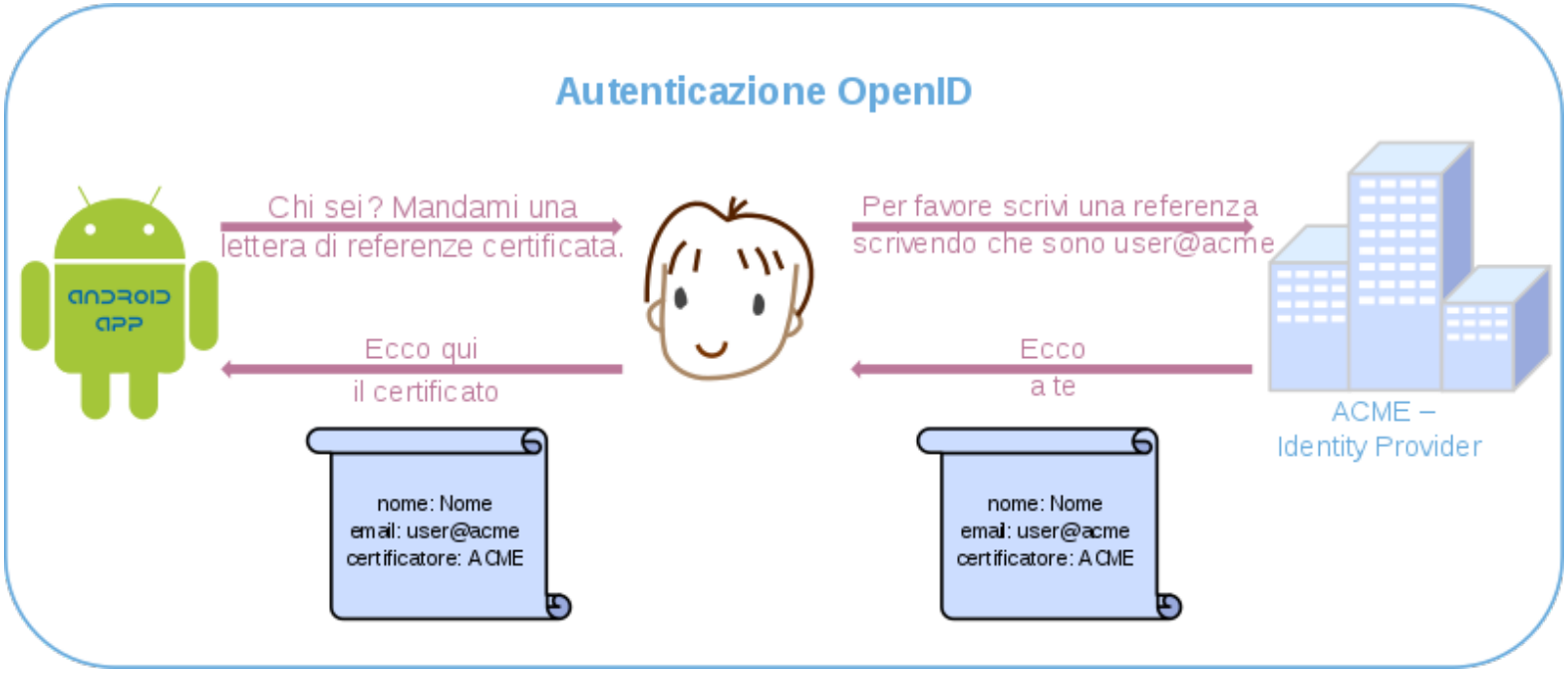
4. OpenID Connect

- È un layer aggiuntivo sopra il protocollo OAuth2
- Consente ai client di verificare l'identità di un utente finale nonché di ottenere informazioni di base tramite un'API HTTP RESTful, utilizzando JSON come formato dati
- è un protocollo di autenticazione decentralizzato ideato per autenticarsi in un servizio (service provider) riutilizzando account già esistenti da altri siti web (identity provider). Lo scopo è di evitare che ogni utente necessiti di registrare un account in ogni fornitore di servizi

4. OpenID Connect

1. Un'applicazione richiede all'utente di effettuare l'accesso
2. L'utente sceglie uno fra gli identity provider a disposizione o comunque ne specifica uno (che da qui in poi chiameremo "ACME")
3. L'utente è reindirizzato al sito di ACME
    1. L'utente effettua il login nel sito di ACME
    2. ACME eventualmente chiede all'utente quali informazioni serve trasmettere al servizio (solo l'e-mail ad esempio)
    3. ACME rilascia all'utente un certificato che può essere utilizzato per verificare che egli si tratti effettivamente di un'utenza ACME , L'utente ritorna automaticamente al sito del servizio di origine consegnandogli il certificato e le informazioni personali
4. L'utente è ora autenticato

4. OpenID Connect



ITS
TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

GRAZIE!

POR Piemonte
FSE 2014-2020