# Securing Web APIs in ASP.NET Core

Brock Allen
@BrockLAllen
Solliance, Inc

## Overview

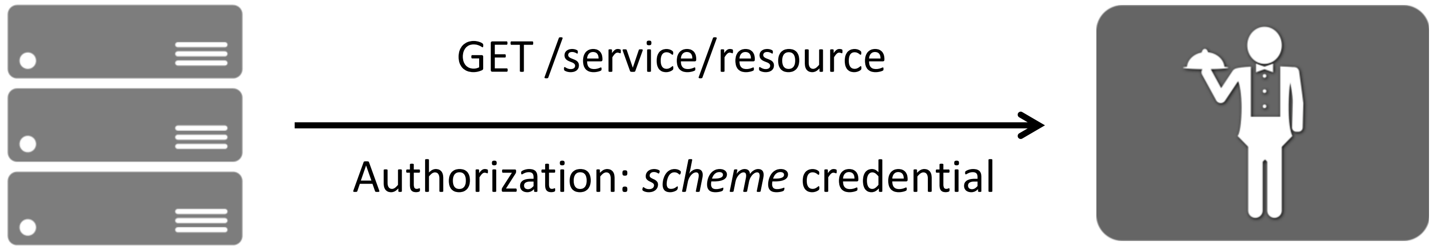
- Web API Security Design
- OAuth2
- ASP.NET Core Security Architecture
- Using APIs server-to-server
- Using APIs on behalf of users

## Securing Web APIs

- No Cookies for securing Web APIs
  - Not all clients are browsers
  - Not all clients have a user
  - Cookies require anti-forgery protection

# HTTP Authentication

- Authorization header used to send credentials
  - e.g. shared secrets, signatures, access tokens…

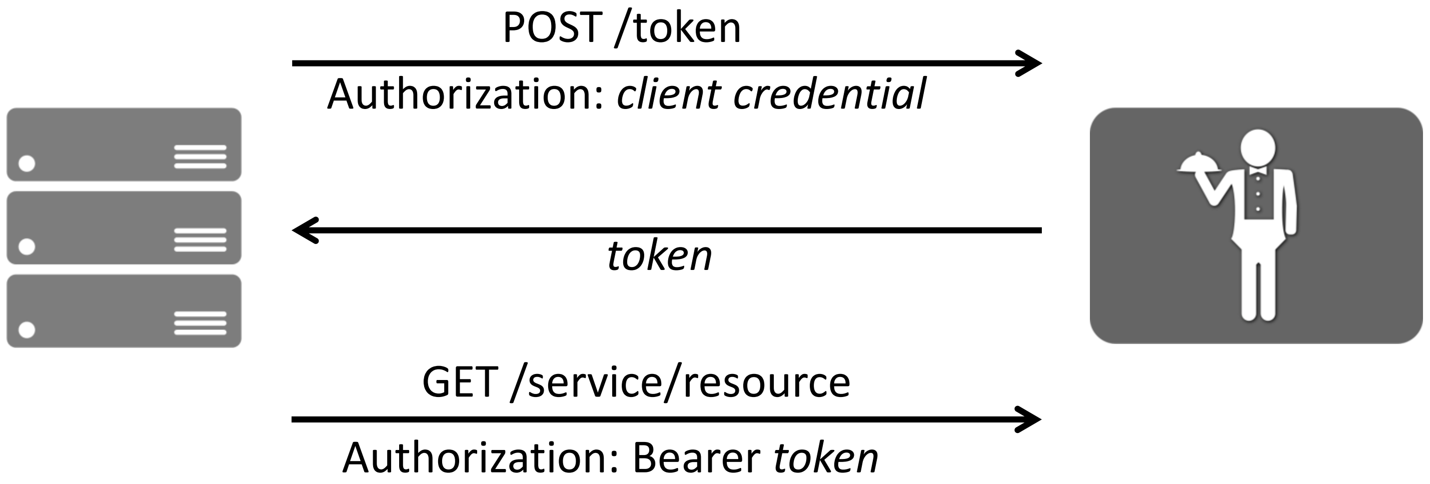


GET /service/resource

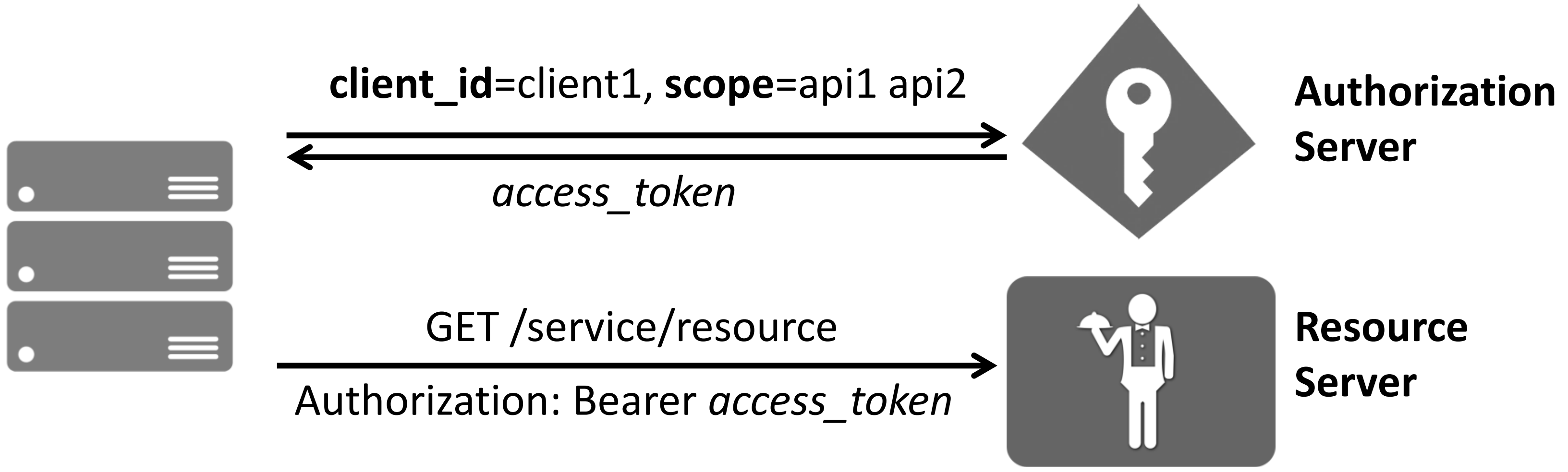
Authorization: *scheme* credential

# Token-based Authentication

- Similar to cookies, but for APIs


POST /token
Authorization: *client credential*
*token*
GET /service/resource
Authorization: Bearer *token*

# OAuth 2.0


**client_id**=client1, **scope**=api1 api2
*access_token*
**Authorization Server**
GET /service/resource
Authorization: Bearer *access_token*
**Resource Server**

# Issuing Tokens: IdentityServer

- ASP.NET Core does not have a token server
  - Microsoft recommends IdentityServer
- OpenID Connect and OAuth 2.0 token service
  - FOSS, Apache 2.0, ASP.NET Core
  - https://identityserver.io

# JWT Access Tokens

**Header**

```
{
  "typ": "JWT",
  "alg": "RS256"
  "kid": "1"
}
```

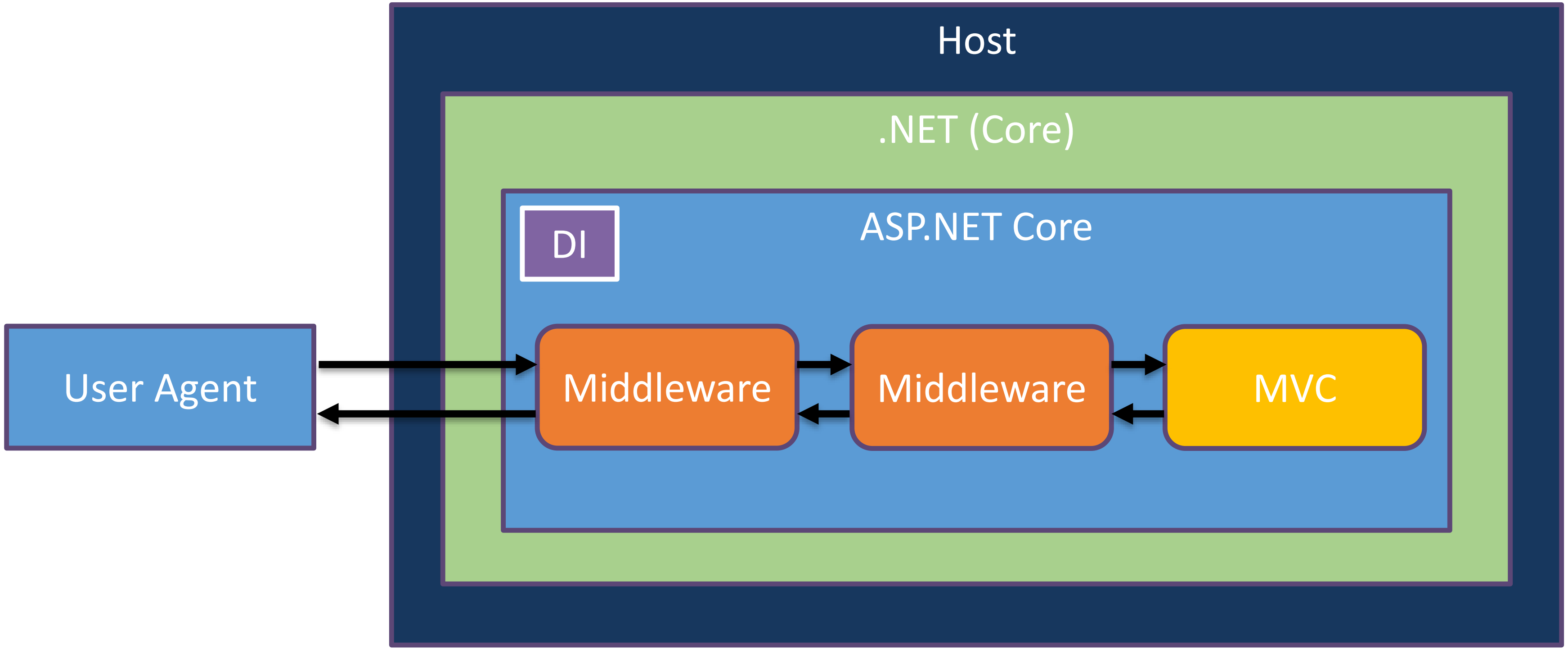
**Payload**

```
{
  "iss": "http://myIssuer",
  "exp": "1340819380",
  "aud": "http://myResource",

  "client_id": "client1",
  "scope": ["api1", "api2"]
}
```

$$\underbrace{\texttt{eyJhbGciOiJub25lIn0}}_{\textbf{Header}}\texttt{.}\underbrace{\texttt{eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMD}}_{\textbf{Payload}}\texttt{.}\underbrace{\texttt{4MTkzODAsDQogImh0dHA6Ly9leGFt}}_{\textbf{Signature}}$$

# ASP.NET Core Architecture



# Authentication in ASP.NET Core

- Combination of middleware and services and handlers in DI
  - Middleware invokes handlers for request related processing
  - Handlers can be also invoked manually
- Handlers implement specific authentication methods
  - Cookies for browser based authentication
  - Google, Facebook, and other social authentication
  - OpenId Connect for external authentication
  - JSON web token (JWT) for token-based authentication

## Access Token Validation

- JWT bearer token authentication handler

```
services.AddAuthentication(defaultScheme: "Bearer")
    .AddJwtBearer(options =>
    {
         options.Authority = "https://url_of_your_token_service";
         options.Audience = "api1";
    });
```

# User-Centric Clients

- Typical Pattern
  - authenticate user
  - make API calls **on behalf** of the user

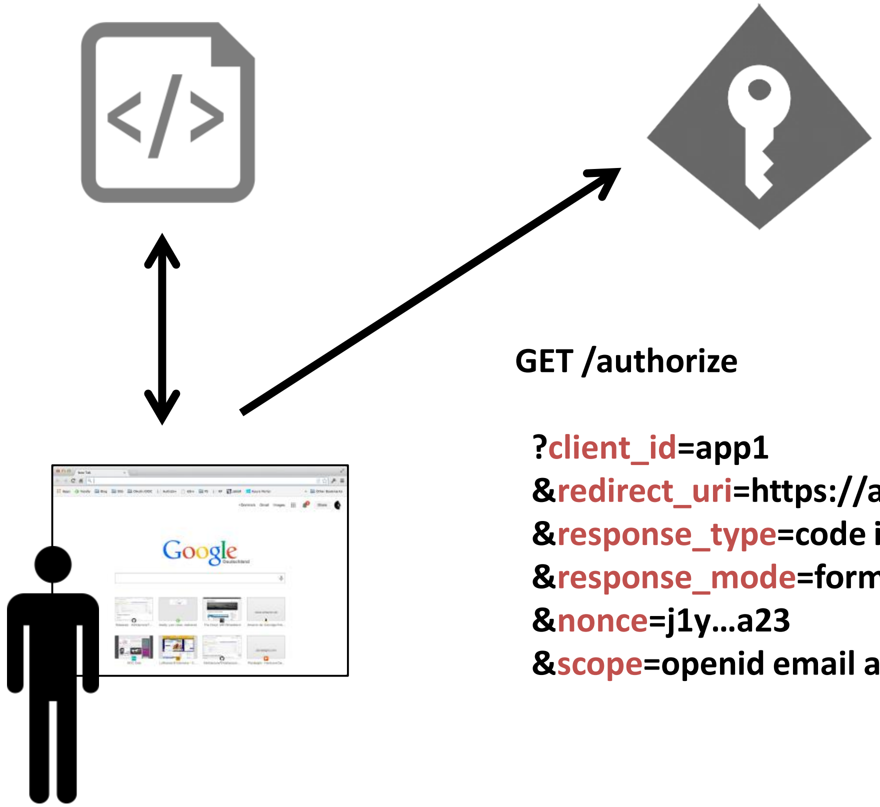
- Server-side Web Applications
- Client-side Web Apps/SPAs
- Native/Mobile Applications

# Web Applications

- OpenID Connect Hybrid Flow combines
  - user authentication (identity token)
  - access to APIs (access token)

- Additional Security Features
  - access tokens not exposed to the browser
  - (optional) long-lived API access

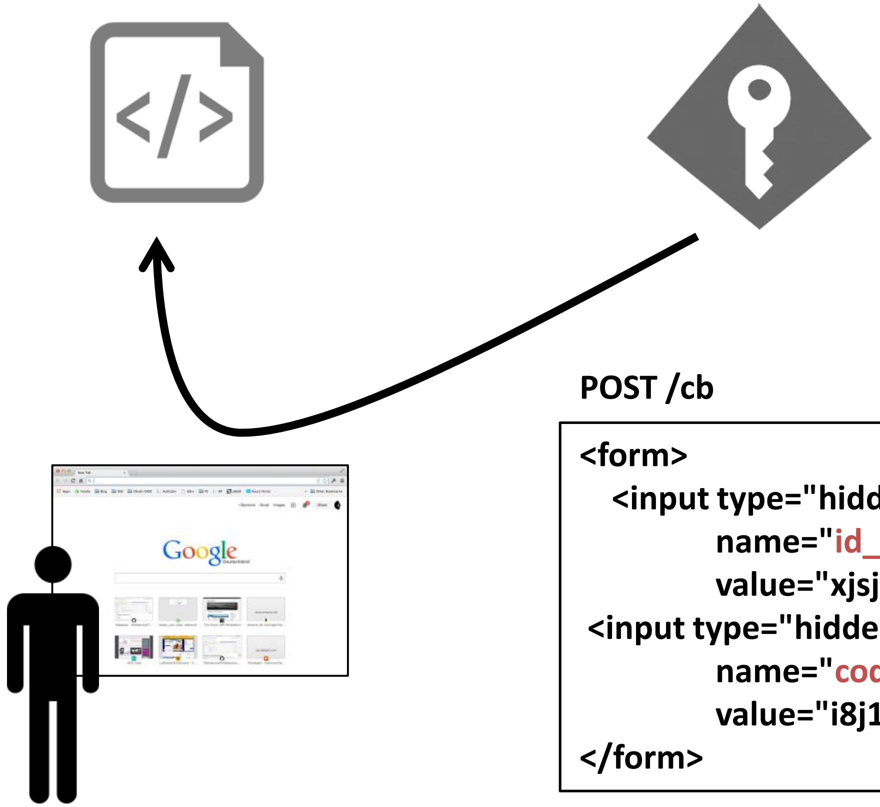
# Hybrid Flow Request



GET /authorize

?client_id=app1
&redirect_uri=https://app.com/cb
&response_type=code id_token
&response_mode=form_post
&nonce=j1y…a23
&scope=openid email api1 api2

# Hybrid Flow Response

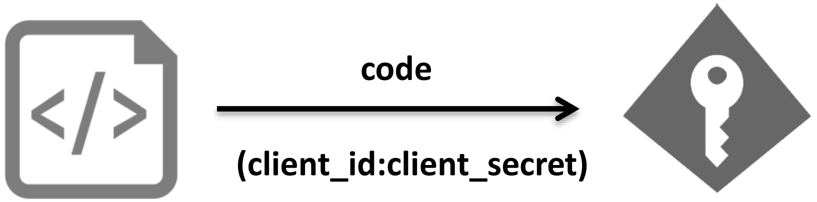


POST /cb

```
<form>
   <input type="hidden"
            name="id_token"
            value="xjsj…aas" />
 <input type="hidden"
            name="code"
            value="i8j1…jj19" />
</form>
```

## Retrieving the Access Token

- Exchange code for access token
  - using client id and secret

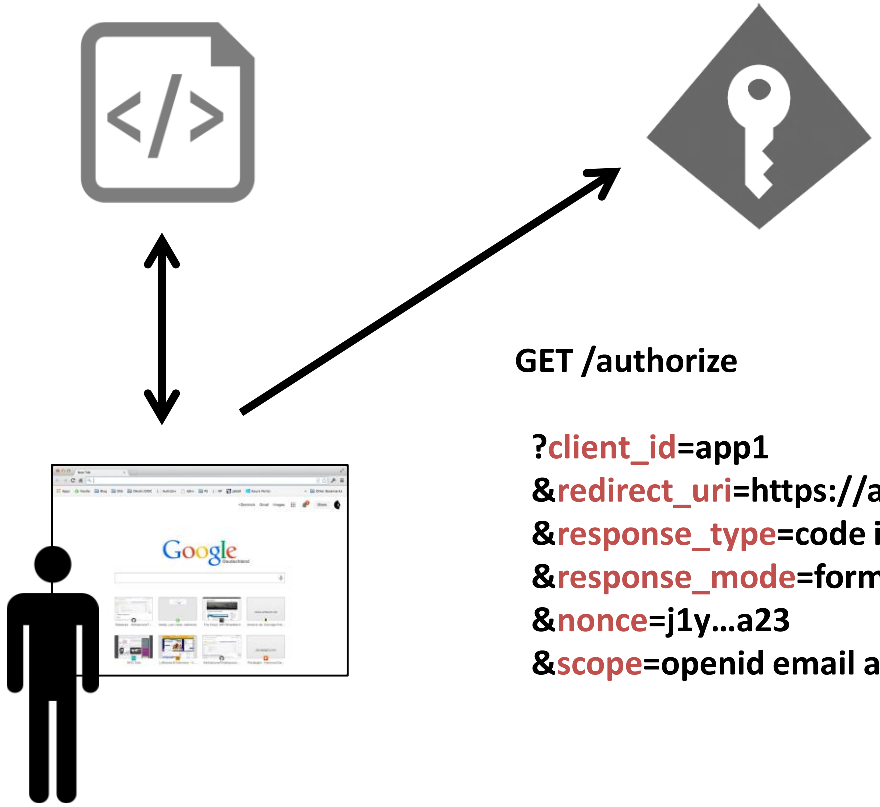


```
{
  access_token: "xyz…123",
  expires_in: 3600,
  token_type: "Bearer"
}
```

## Access Token Lifetime Management

- Access tokens have finite lifetimes
  - requesting a new token requires browser round trip to authorization server
  - should be as short lived as possible
- Refresh tokens allow renewal semantics
  - no user interaction required
  - typically combined with a revocation feature

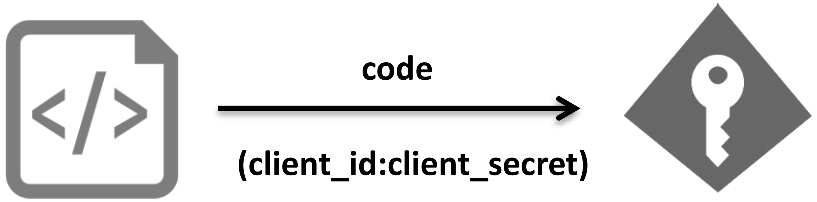
# Requesting a Refresh Token



GET /authorize

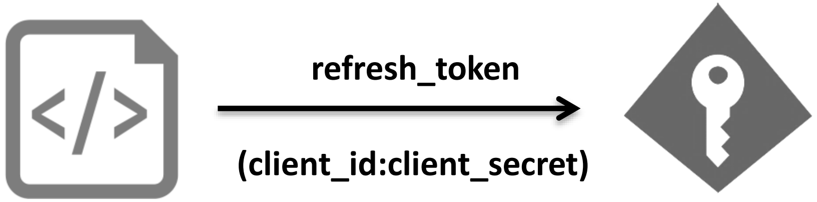
?client_id=app1
&redirect_uri=https://app.com/cb
&response_type=code id_token
&response_mode=form_post
&nonce=j1y…a23
&scope=openid email api1 offline_access

## Retrieving the Access Token (w/ Refresh Token)



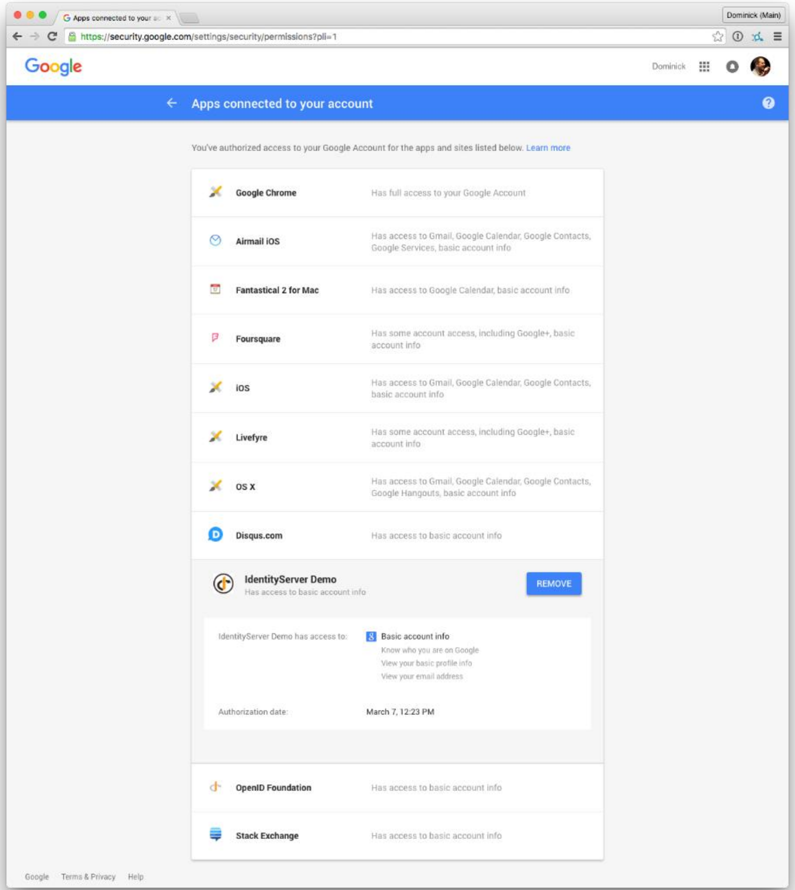
```
{
  access_token: "xyz…123",
  refresh_token: "jdj9…192j",
  expires_in: 3600,
  token_type: "Bearer"
}
```

# Refreshing an Access Token



```
{
  access_token: "xyz…123",
  refresh_token: "jdj9…192j",
  expires_in: 3600,
  token_type: "Bearer"
}
```

# Revocation



Apps connected to your account

You've authorized access to your Google Account for the apps and sites listed below. Learn more

| | |
|---|---|
| Google Chrome | Has full access to your Google Account |
| Airmail iOS | Has access to Gmail, Google Calendar, Google Contacts, Google Services, basic account info |
| Fantastical 2 for Mac | Has access to Google Calendar, basic account info |
| Foursquare | Has some account access, including Google+, basic account info |
| iOS | Has access to Gmail, Google Calendar, Google Contacts, basic account info |
| Livefyre | Has some account access, including Google+, basic account info |
| OS X | Has access to Gmail, Google Calendar, Google Contacts, Google Hangouts, basic account info |
| Disqus.com | Has access to basic account info |

**IdentityServer Demo**
Has access to basic account info

IdentityServer Demo has access to: Basic account info
Know who you are on Google
View your basic profile info
View your email address

Authorization date: March 7, 12:23 PM

| | |
|---|---|
| OpenID Foundation | Has access to basic account info |
| Stack Exchange | Has access to basic account info |

Apps connected to your ac
Dominick (Main)
https://security.google.com/settings/security/permissions?pli=1
Google
Dominick
REMOVE
Google
Terms & Privacy
Help

# Summary

- Tokens-based authentication for securing APIs
- OAuth2 protocol for obtaining tokens
- Client credentials flow for server-to-server APIs
- Hybrid flow for user-based web apps calling APIs
- IdentityServer FOSS token server