Introduction to OAuth 2.0

Justin Richer
Bespoke Engineering
APIs are meant to be used

- Much of my data and the functionality of my life is available through APIs today
- I want to have applications access my APIs
- I don’t want the applications to have to impersonate me
- I don’t want to share my keys with everyone
A valet key for APIs

• A **valet key** gives someone else limited access to a car

• What if we could do that for web APIs?
OAuth 2.0
From the spec (RFC6749)

The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf.
The good bits

The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf.
In other words

OAuth 2.0 is a **delegation protocol** that lets people **allow applications to access things** (like APIs) on their behalf.
Who is involved?

- Resource Owner
- Authorization Server
- Client
- Protected Resource

©2016 Bespoke Engineering LLC
The resource owner

- Has access to some resource or API
- Can delegate access to that resource or API
- Usually has access to a web browser
- Usually is a person
The protected resource

- Web service (API) with security controls
- Protects things for the resource owner
- Shares things on the resource owner’s request
The client application

• Wants to access the protected resource
• Does things on the resource owner’s behalf
• Could be a web server
  – But it’s still a “client” in OAuth parlance
  – Could also be a native app or JS app
What are we trying to solve?

The Goal:
Give the client access to the protected resource on behalf of the resource owner.
Introducing the Authorization Server (AS)

The Authorization Server gives us a mechanism to bridge the gap between the client and the protected resource.
The Authorization Server

- Generates tokens for the client
- Authenticates resource owners (users)
- Authenticates clients
- Manages authorizations
OAuth Tokens

• Represent granted delegated authorities
  – From the resource owner to the client for the protected resource

• Issued by authorization server

• Used by client
  – Format is opaque to clients

• Consumed by protected resource
Example OAuth Tokens

- 92d42038006dba95d0c501951ac5b5eb
- 2df029c6-b38d-4083-b8d9-db67c774d13f
- eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ
- waterbuffalo-elephant-helicopter-argument
The OAuth approach at the AS

• Client authenticates for itself
• User authorizes client to act on user’s behalf
• Server generates a token to represent that authorization
• Client presents that token to gain access
You’ve used OAuth

Approve this client?

client_id: oauth-client-1

The client is requesting access to the following:

- read
- write
- delete

[Approve] [Deny]
The pieces of OAuth

- Resource Owner
- Authorization Server
- Access Token
- Protected Resource
- Client
THE AUTHORIZATION CODE FLOW

A deep dive into the canonical OAuth 2.0 transaction
The authorization code flow
TWO FORMS OF COMMUNICATION
The front channel

Front channel uses HTTP redirects through the web browser, no direct connections
The back channel

Back channel uses direct HTTP connections between components, the browser is not involved.
THE AUTHORIZATION CODE FLOW

Step by step
Authorization Code: Step 1

Client redirects the resource owner to the authorization server.
Authorization Code: Step 2

Resource owner authenticates to the authorization server
Authorization Code: Step 3

Resource owner authorizes the client
A layered trust model

**Whitelist**
- Internal parties
- Known business partners
- Customer organizations
- Trust frameworks
- Centralized control
- Traditional policy management

**Greylist**
- Unknown entities
- Trust On First Use
- End user decisions
- Extensive auditing and logging
- Rules on when to move to the white or black lists

**Blacklist**
- Known bad parties
- Attack sites
- Centralized control
- Traditional policy management
Authorization Code: Step 4

Authorization server redirects resource owner back to the client with an authorization code.
Authorization Code: Step 5

Client sends the authorization code back to the authorization server along with its own credentials.
Authorization Code: Step 6

Authorization server issues OAuth token to the client.
Authorization Code: Step 7

Client accesses the protected resource using the access token.
Interpreting the token

• The client never knows or cares what’s in the token itself

• The resource server needs to understand what’s in the token
  – Who it’s issued for
  – What it’s good for
Thank You
B A C K U P  S L I D E S

Here there be dragons
OTHER WAYS TO DO OAuth 2.0
The implicit flow

Implicit grant type
uses only the front channel since the client is inside the browser
The client credentials flow

**Client credentials grant type:** Client trades its own credentials for a token, uses only the back channel since the client is acting on its own behalf.
The resource owner password flow

Resource owner credentials grant type:
Client trades username and password for an OAuth token over the back channel
The assertions flows

Client trades a cryptographically protected element (assertion) for a token
Different use cases

- Authorization code flow: web applications, some native applications
- Implicit flow: in-browser applications
- Client credentials flow: non-interactive
- Password flow: trusted legacy clients
- Assertion flows: trust frameworks
How to choose a flow

Is the client acting on behalf of a resource owner?

Yes

Can the resource owner interact with a web browser while using the client?

Yes

Is the client running completely inside of a web browser?

Yes

Implicit

No

Does the user have a simple set of credentials like a password?

Yes

Resource Owner Credentials

No

Client Credentials

Is the client acting on its own behalf?

Yes

Authorization Code

No

Assertion

Is the client acting on behalf of a third party authority?

Yes

Choose the appropriate OAuth grant type for the type of application you’re building

No
Can we build authentication on OAuth?
How can we split the network?

The security domain boundary is crossed.

The diagram illustrates the roles and components involved in a network split, including Resource Owner, Client, User, Relying Party, Authorization Server, Identity Provider, and Protected Resource.
A better way to split the network
That works!

• We’re using OAuth to **protect the identity**
• The client **consumes the identity**
Authorization is Chocolate

• Good on its own
• Great as part of a larger recipe
• Many different recipes can use it
Authentication is Fudge

- Confection with several ingredients
- Tends to have one flavor as the most obvious
- Could be made using chocolate
  - But not required
Agreeing on a recipe

• Let’s make a recipe for chocolate fudge:
  – Standard authentication protocol
  – Built on top of standard authorization protocol
  – Interoperable cross domain
OpenID Connect

• IdP offers interactive OAuth flows
• ID Token carries authentication information
  – Formatted as a JWT
  – Audience is the client, not the resource
• UserInfo Endpoint
  – Standard set of claims and scopes
User Managed Access
Person to person delegation

- OAuth lets Alice share with herself
- UMA lets Alice share with Bob
  - Bob is the “Requesting Party (RqP)” to Alice’s “Resource Owner (RO)”
  - Alice can set policies ahead of time
User Managed Access

Protection API

Authorization API

Resource Owner

Authorization Server

Protected Resource

Requesting Party

Client

©2016 Bespoke Engineering LLC
Wide ecosystem benefits

- Alice can introduce a new resource to her AS
- The resource server can manage its access using this AS and its tokens
Reference book for OAuth 2

- *OAuth 2 In Action*
- First 9 chapters available today, more coming soon
- Out this spring/summer

https://manning.com/books/oauth-2-in-action