Working with OAuth

Filip Ekberg
SENIOR SOFTWARE ENGINEER

@fekberg fekberg.com

Overview

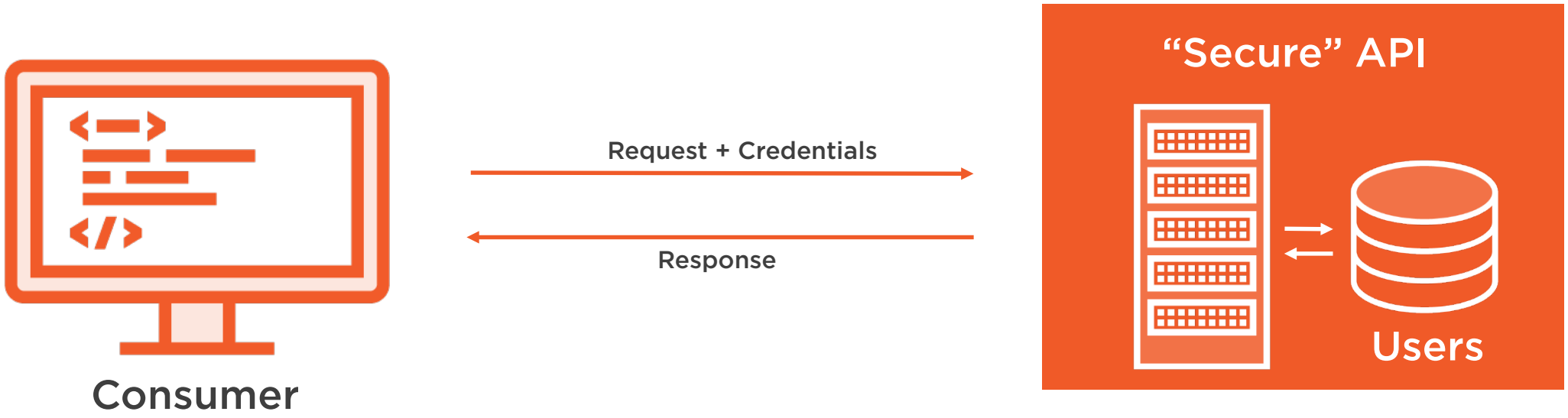
How does OAuth (Authorization Server) compare to our bad API security?

How do we inspect and validate an Access Token?

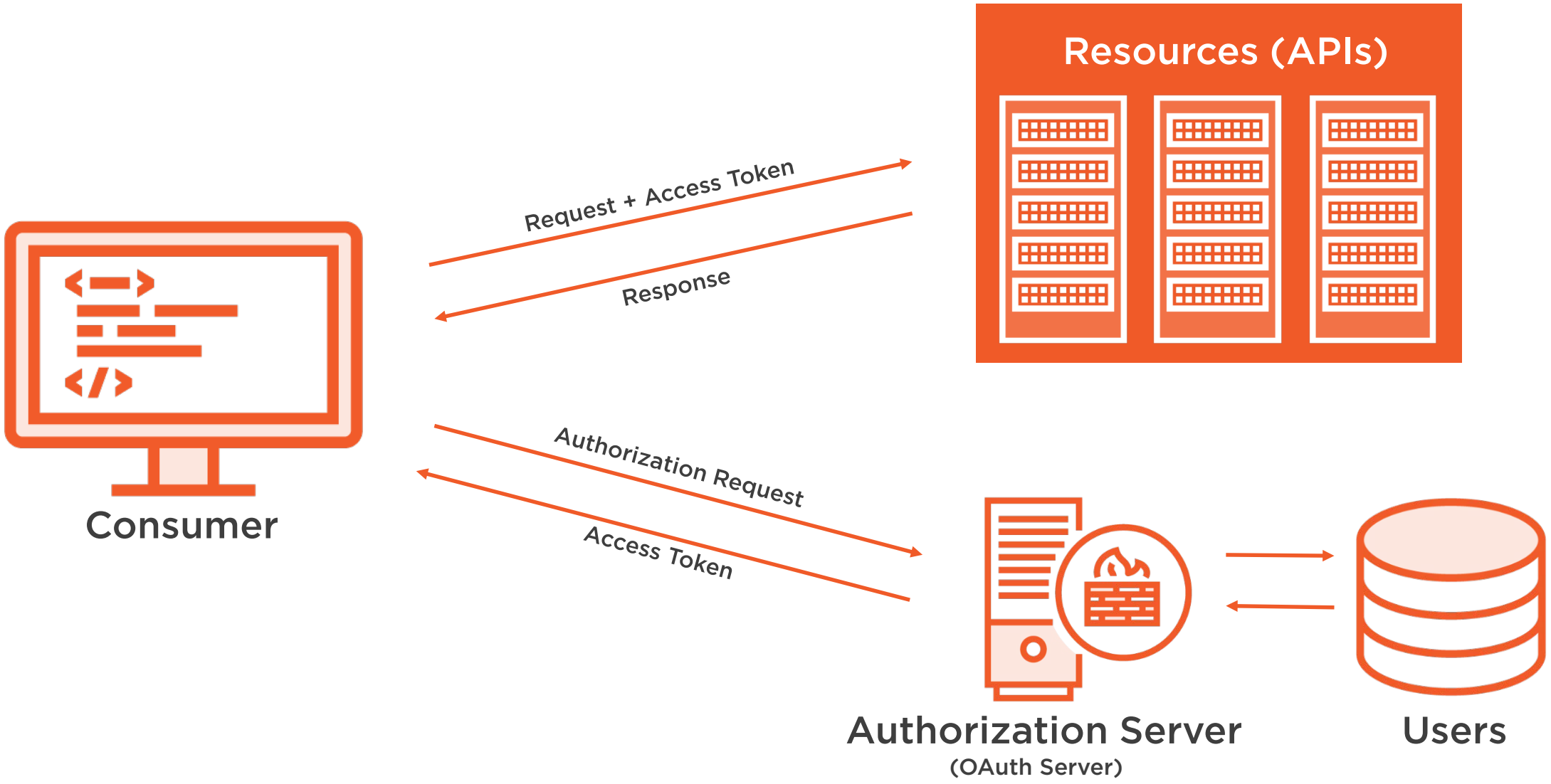
Where do Google, Twitter, Facebook and others fit into the picture?

What does this mean for third parties?

What Did Our **Bad** API Do?



Where Does OAuth Come In?



Why Not **Only** Twitter, Google, Facebook et. al?

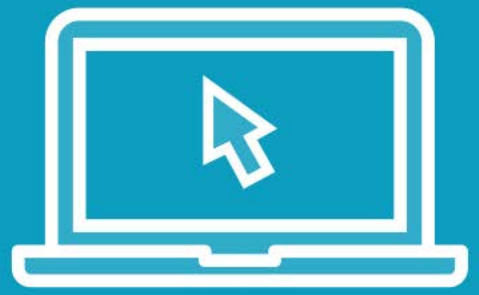
Don’t build your business around third party identity services

Supply social login as an option

Headache for third parties using your API

Don’t Force External Logins

Demo



Using OAuth: What does it look like?

How can the Resource Servers (APIs) validate an Access Token?

What does the content of an Access Token look like?

Can anyone use my Authorization Server?

The OAuth 2.0 Authorization Framework

RFC 6749

OAuth Endpoints

/authorize

- New Access Token request (Certain Flows)

/token

- New Access Token request (Certain Flows)
- Refresh Access Token
- Trade Authorization Code for an Access Token

/revocation

- Revoke an Access or Refresh Token (RFC 7009)

OpenID Connect Endpoints

/userinfo

/checksession

/endsession

/.well-known/openid-configuration

- Lists endpoints and configurations

/.well-known/jwks

- Lists information about JWT signing keys. Used for token validation

Access Token

JSON Web Token

Information about the authorized user

Signed by the Authorization Server

Resource Server (API) should validate using a public key

Base64 encoded information

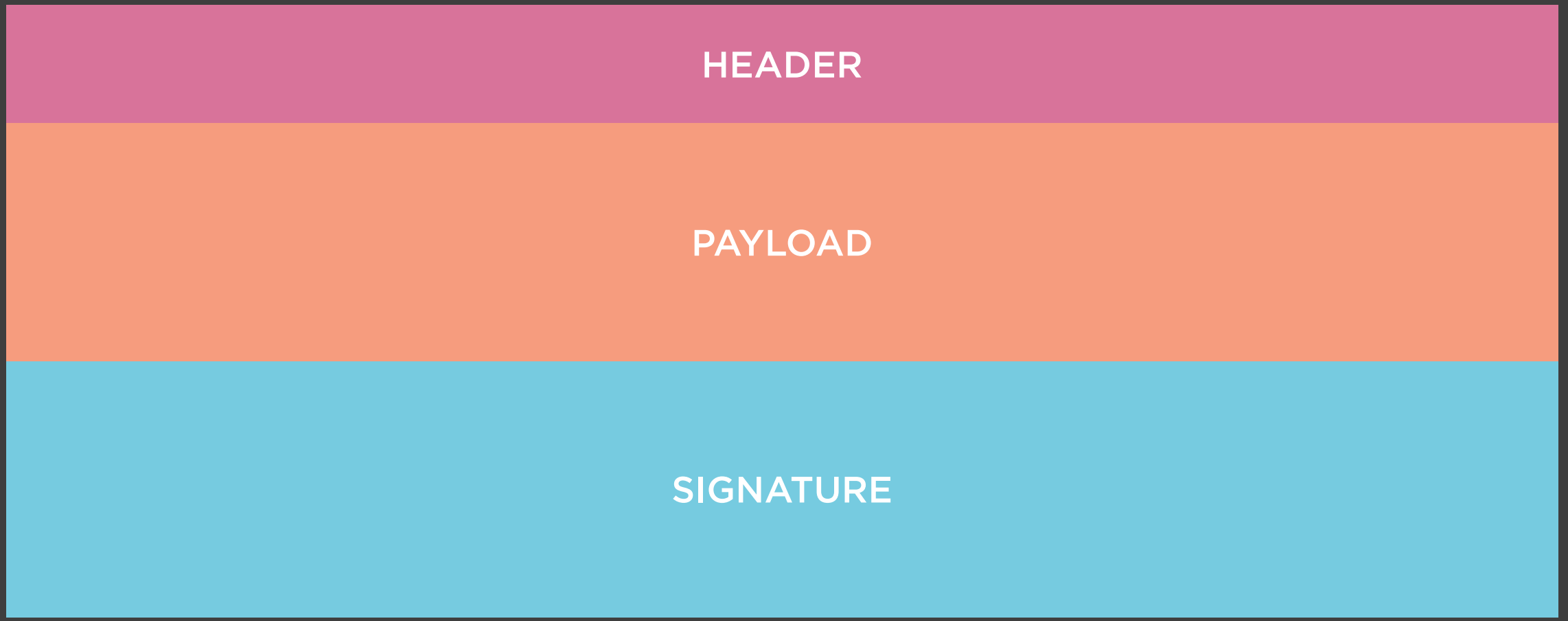
```
{
    "access_token": "eyJ0eXAiOiJK…",
    "expires_in": 3600,
    "token_type": "Bearer“
}
```

Getting an Access Token

POST /token HTTP/1.1
Host: pluralsight-oauth.azurewebsites.net
Content-Type: application/x-www-form-urlencoded

client_id=1&secret=secret&grant_type=password&username=**mail@filipekberg.se**&password=**password**

Access Token (JWT)



Claims (Payload)

- Issuer
- Audience
- Expiry
- Not Before
- Client Id
- Scopes
- Custom data

Scopes

Limit access to functionality based on Scopes

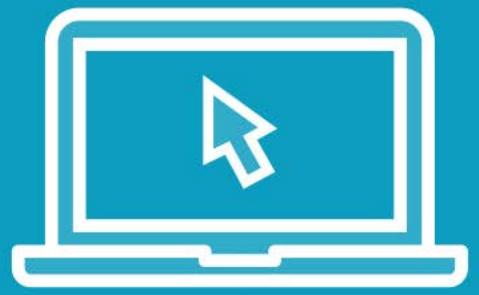
OpenID Connect Scope Examples

- openid
- profile
- email
- address
- offline_access

Custom Scope Examples

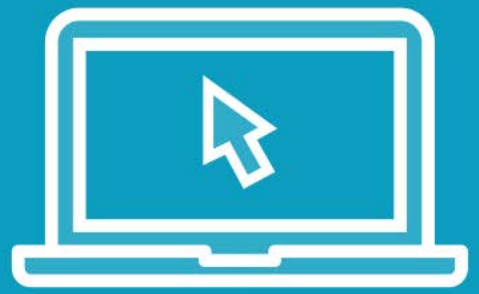
- read
- write

Demo



Getting and Inspecting Access Token

Demo



Using an Access Token

Access Token Validation

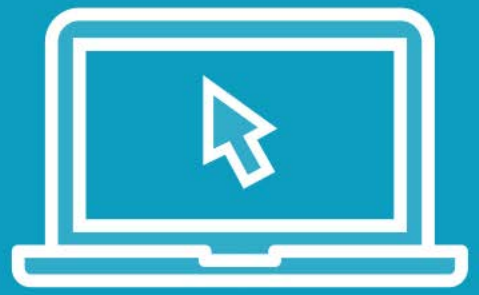
Authorization Server exposes a public key

Resource Server and Consumer should validate on their end

Could use Token Introspection for validation (RFC 7662)

Always **validate** the Access Token

Demo



Manipulating an Access Token

Refresh Token

Access Tokens could expire anytime

Refresh Tokens allow you to get a new Access Token

Delivered together with your first Authorization

Obtain a Refresh Token by requesting the Scope **offline_access**

Flows & Grants

Redirect Flows

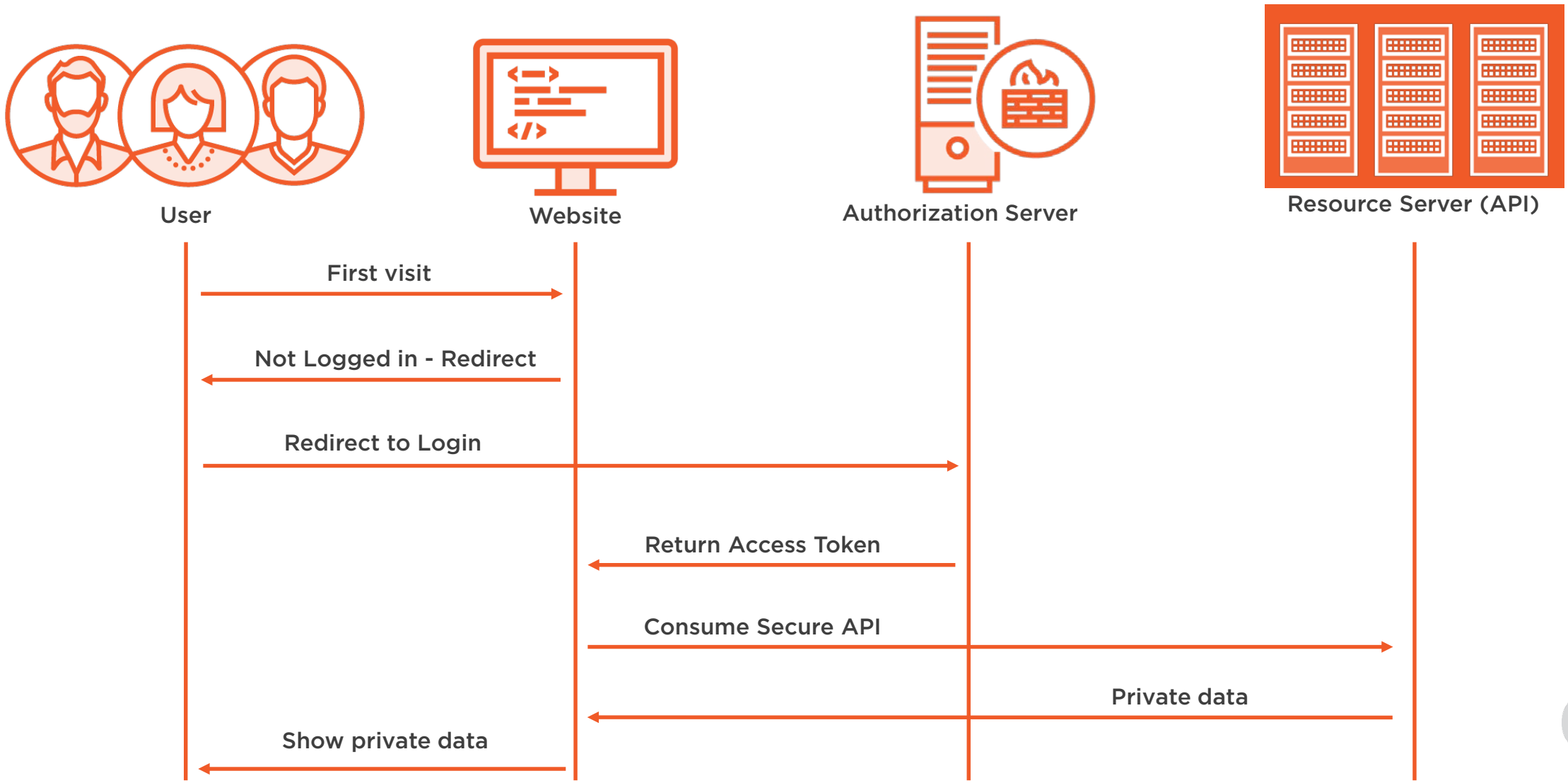
- Implicit Grant
- Authorization Code

Credential Flows

- Resource Owner Password Credentials
- Client Credentials

More in Module 4 and 5!

Redirect Flow (Implicit Grant)



Implements both OAuth 2.0 and OpenID Connect

Easy to Setup & Get Started

Built by Industry Experts

Well supported in older as well as newer ASP.NET versions

Summary

- Where OAuth Fits into the Picture
- How to Retrieve and Analyze an Access Token
- Importance of Access Token Validation
- How to Refresh an Access Token
- Different OAuth and OpenID Connect Endpoints
- Scopes, Claims as well as other OAuth and OpenID Connect Fundamentals
- External Login can be added as an option