Securing Your API

Kevin Dockx
Architect

@KevinDockx | www.kevindockx.com

Coming Up

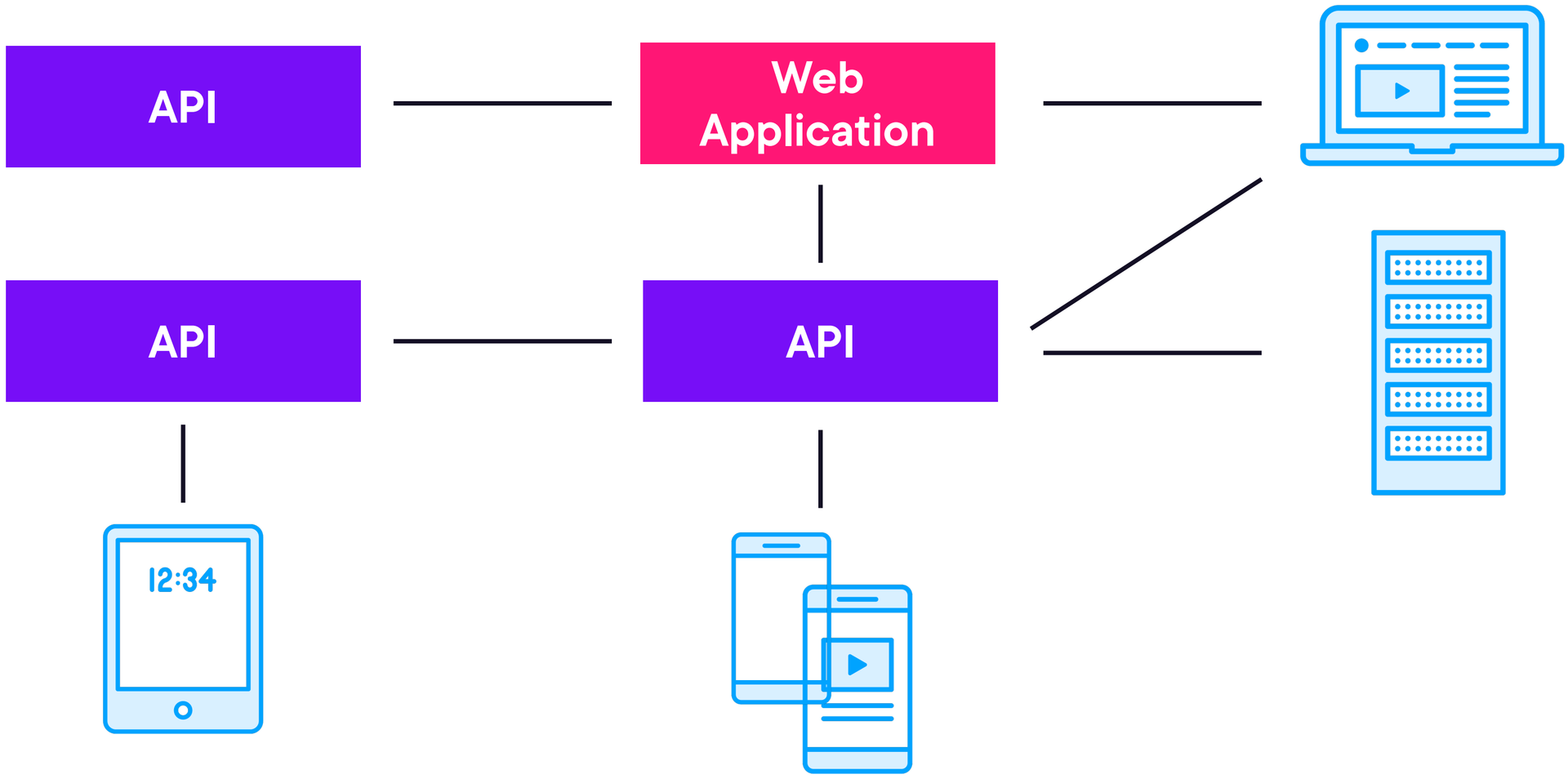
A few words on securing APIs

Supporting and implementing token-based security

Working with authorization policies

OAuth2 and OpenID Connect

A Few Words on Securing APIs



A Few Words on Securing APIs

Which entity (user/app) is trying to access the API?

- How can we verify this?

Once we know who/what the entity is, how do we check whether access should be granted?

Sending username/password on each request proved to be a bad idea...

- Huge attack vector

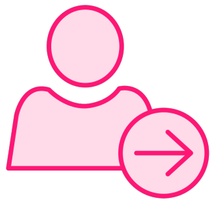
A Few Words on Securing APIs

Token-based security

- Send a token on each request
- A token represents consent
- Validate the token at level of the API

Approach works for almost all modern application types

Implementing Token-based Security



API “login” endpoint accepting a username/password

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c


```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```


Payload

E.g.: some JSON that contains generic token info, like when the token was created, and some info about the user

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

Signature

A hash of the payload, used to ensure the data wasn’t tampered with

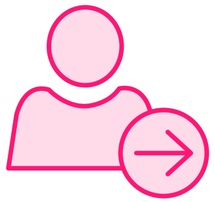
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c


```
{
  "alg": "HS256",
  "typ": "JWT"
}
```


Header

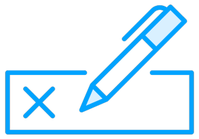
Essential token information like the key algorithm used for signing

Implementing Token-based Security

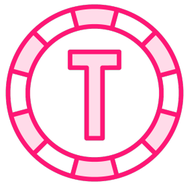


API “login” endpoint accepting a username/password

```
POST api/login
```



Ensure the API can only be accessed with a valid token



Pass the token from the client to the API as a Bearer token on each request

```
Authorization: Bearer mytoken123
```

Demo

Creating a token

Demo

Requiring and validating a token

Demo

Using information from the token in your controller

Working with Authorization Policies

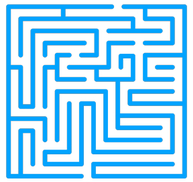
Authorization policies help with building a full-fledged authorization layer

- Avoids having to enter the actual controller action

ABAC/CBAC/PBAC

Access rights granted through policies

A policy combines a set of attributes (claims) together



Allows much more complex rules than RBAC
(Role-based Access Control)

Policy example

“If users are from country A and live in a city with more than half a million people, and were born between 1980 and 1985, then they are allowed action X“

Demo

Using information from the token in an authorization policy

Generating a Token with dotnet user-jwts

More often than not, you won't write your own token generation service

- Handled at a central location
- Proven identity provider (Entra ID, Ping, OKTA, IdentityServer, ...)

Can be challenging during development

Demo

Generating a token with
`dotnet user-jwts`

Improving Token-based Security with OAuth2 and OpenID Connect

Security is a large, fast-evolving topic

- We implemented the basics/a rudimentary form of token-based security
- Standards exist that improve on this

OAuth2

OAuth2 is an open protocol to allow secure authorization in a simple and standard method from web, mobile and desktop applications

OpenID Connect

OpenID Connect is a simple identity layer on top of the OAuth2 protocol

Summary

Multiple ways of securing APIs exist

- Token-based security is the advised approach

Summary

Token-based security

- Create a login endpoint that accepts credentials and returns a token
- Send the token to the API as a Bearer token on each request
- Validate the token at level of the API

Summary

Command-line tool for generating tokens, typically used during local development:

- `dotnet user-jwts`

Summary

Use authorization policies to create an authorization layer

Vastly improve token-based security by relying on standards like OAuth2 and OpenID Connect

Up Next:

Versioning and Documenting Your API