JWT in depth

...

/dʒɒt/

What for ?

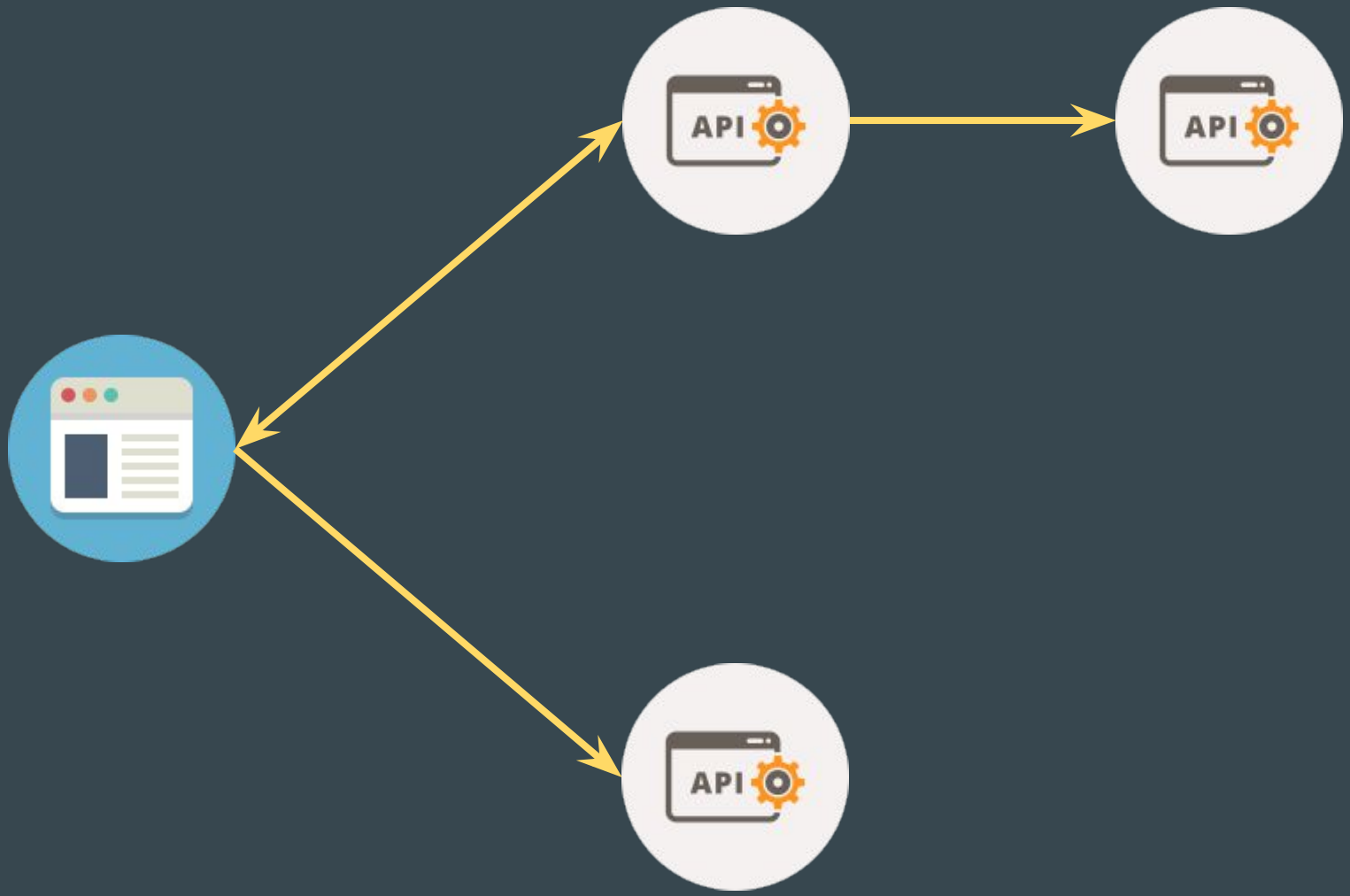
...

But...

...

Who ?

...


API
API
API

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

```
{
  "sub": "Mathieu POUSSE",
  "position": "Works at Zenika",
  "twitter": "@m_pousse",
   "role": ["speaker"]
}
```

•••

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJNYXRoaWV1IFBPVVNTRSIsInBvc2l0aW9uIjoiV29ya3MgYXQgWmVuaWthIiwidHdpdHRlciI6IkBtX3BvdXNzZSJ9.9-DLLyNebtUVCTlgTpn2GUIYZeSbLw44KwdtGdy2Z-8

In depth

...

Enveloppe

JWS

{ Header }

{ Payload }

{ Integrity }

JWE

{ Header }

{ Key }

{ IV }

{ Payload }

{ Integrity }

JOSE - Javascript Object Signing and Encryption

alg: algorithm

crit: critical claims
jku: JWT Key URL
x5u: X509 URl
jwk: Encoded Key
kid: Key Id
x5t: x509 thumbprint

Payload / Claims

sub: subject
exp: expiration date
iat: issue at
iss: issuer
jti: jwt token id
aud: audience

private claims

Signature

HS*** = HMAC using SHA-*** hash algorithm

RS*** = RSA using SHA-*** hash algorithm

ES*** = ECDSA using P-*** curve and SHA-*** hash algorithm

Base 64

f(x) = JWS

BASE64URL(UTF8(JWS Header)) ‘.’
BASE64URL(JWS Payload) ‘.’
BASE64URL(JWS Signature)

f(x) = JWE

BASE64URL-ENCODE(UTF8(JWE Header)) ‘.’
BASE64URL-ENCODE(JWE Encrypted Key) ‘.’
BASE64URL-ENCODE(JWE IV) ‘.’
BASE64URL-ENCODE(JWE Ciphertext) ‘.’
BASE64URL-ENCODE(JWE Authentication Tag)

Transport

Authorization: Bearer <jwt goes here>

Don’t reinvent the wheel

…

https://jwt.io

Storage

Single Page Application

Cross Domain

Javascript = XSS

Storage

Multiple Page Application

Only for issued domain

Sent on every request

Cookie = CSRF

Questions

...

Mathieu POUSSE
@m_pousse