Why JSON Web Tokens Are Awesome

Ado Kukic

About Me

Ado Kukic

Technical Writer, Dev Advocate

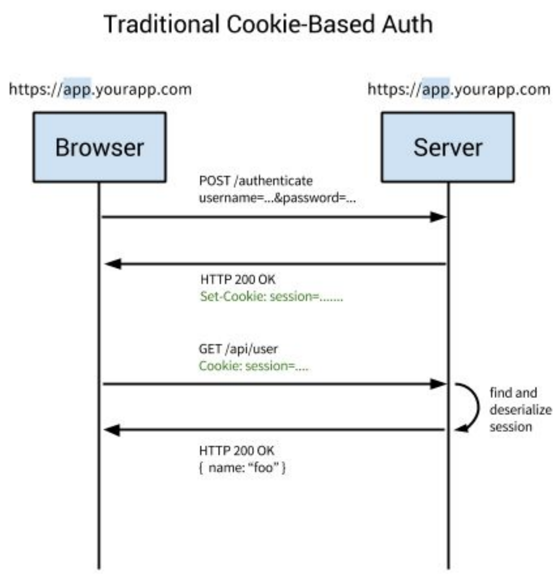
@kukicado
ado@auth0.com

The Need for Token Based Authentication

Stateful

Cross-Domain

Performance



Traditional Cookie-Based Auth

https://app.yourapp.com

https://app.yourapp.com

Browser

Server

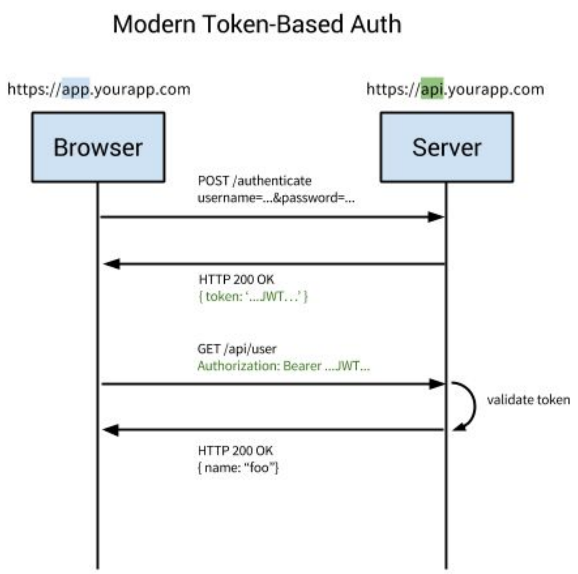
POST /authenticate
username=...&password=...

HTTP 200 OK
Set-Cookie: session=.......

GET /api/user
Cookie: session=....

find and
deserialize
session

HTTP 200 OK
{ name: "foo" }



Modern Token-Based Auth

https://app.yourapp.com

https://api.yourapp.com

Browser

Server

POST /authenticate
username=...&password=...

HTTP 200 OK
{ token: '....JWT....' }

GET /api/user
Authorization: Bearer ....JWT...

validate token

HTTP 200 OK
{ name: "foo"}

Mobile

XSRF

Decoupled

Auth0

Abridged Introduction to JWT’s

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkFkbyBLdWtpYyIsImlhdCI6MTQ2NDI5Nzg4NX0.T6LGj6FdhhliIR4_eOm5flV7xi4xzd83sfKqJ4YfYFQ
```

- **Header** - type of token and hashing algorithm
  > HMAC
  > ECDSA
  > RSA

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

- **Payload** - claims the token makes
  > Reserved - part of the standard (e.g. “iss”, “sub”)
  > Public - common claims registered with IANA (e.g. “name”)
  > Private - custom claims you create (e.g. “role”)

```
{
  "sub": "1234567890",
  "name": "Ado Kukic",
  "iat": 1464297885
}
```

- **Signature** - hash containing header, payload, and secret

  > HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), secret)

Keeping Tokens Secure - Cryptographic Signing

A signature allows a JWT to be **validated** against modifications.

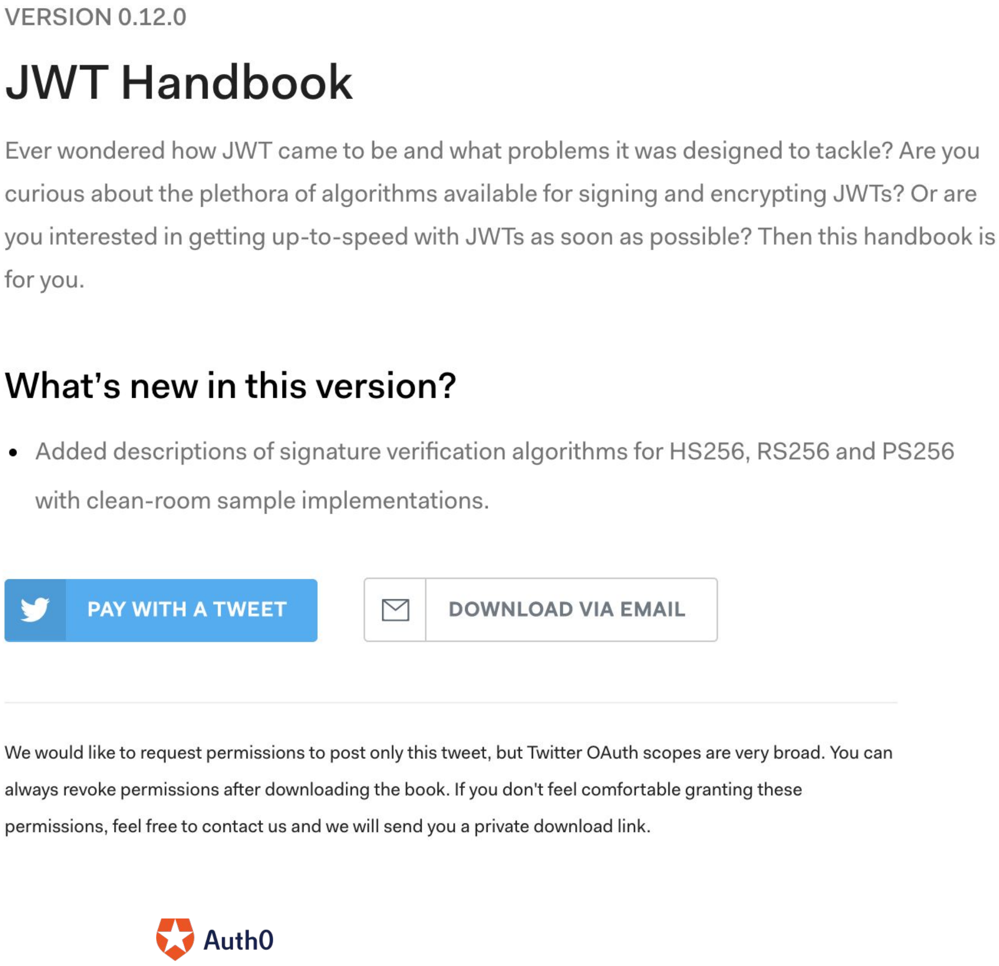
- **HMAC** - most common, symmetric, shared key (length of key is important)
- **RSA** - asymmetric, private/public key pair
- **ECDSA** - asymmetric, private/public key pair, shorter key length than RSA

Concerns with Token Based Authentication

Tokens are Signed, Not Encrypted

- Anyone can view the contents of a JSON Web Token
- This does not mean they can modify it though, even the slightest change will invalidate the token
- Do not store sensitive information within a JWT
- Solution: JSON Web Encryption allows you to safely encrypt the claims of a token

https://goo.gl/Hb7nh5



VERSION 0.12.0

JWT Handbook

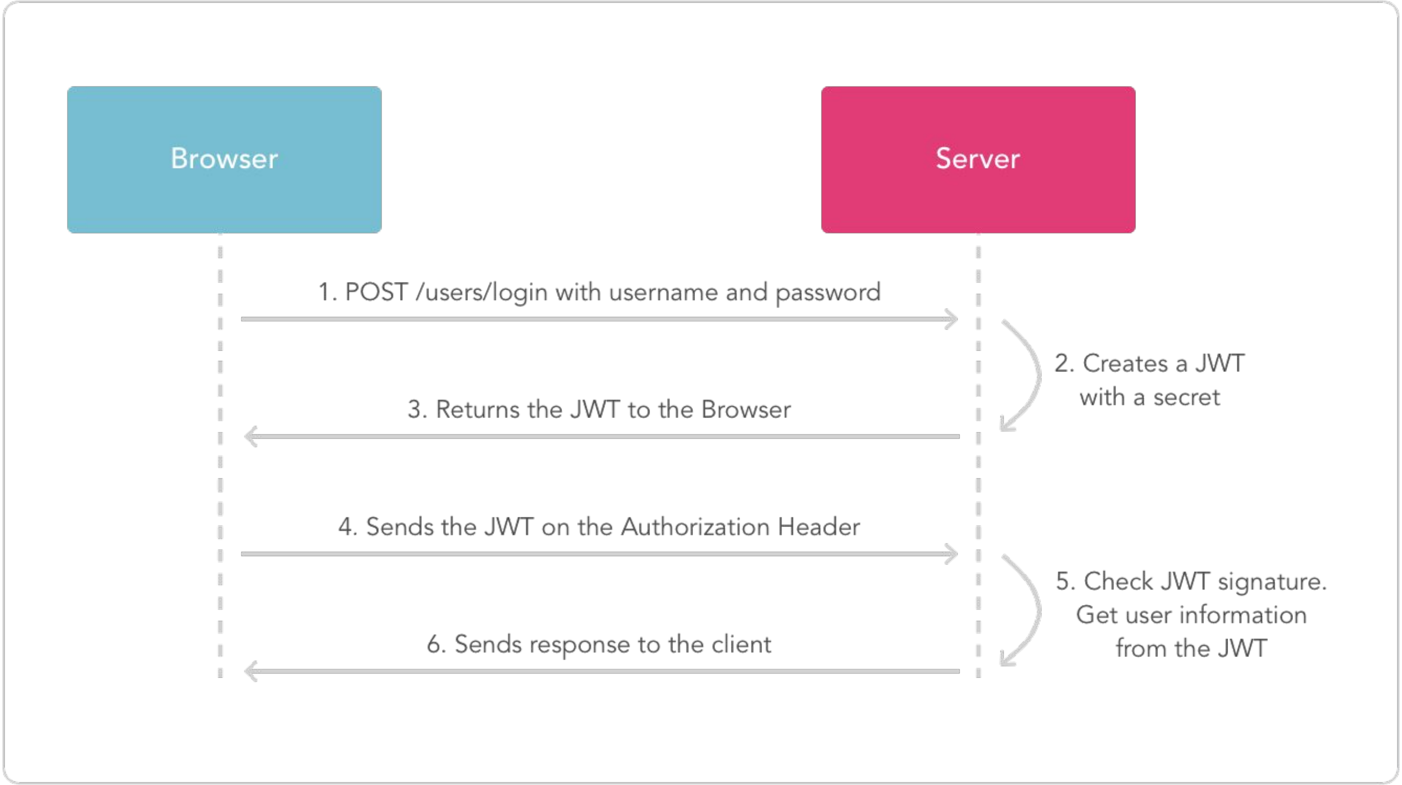
Ever wondered how JWT came to be and what problems it was designed to tackle? Are you curious about the plethora of algorithms available for signing and encrypting JWTs? Or are you interested in getting up-to-speed with JWTs as soon as possible? Then this handbook is for you.

What's new in this version?

- Added descriptions of signature verification algorithms for HS256, RS256 and PS256 with clean-room sample implementations.

We would like to request permissions to post only this tweet, but Twitter OAuth scopes are very broad. You can always revoke permissions after downloading the book. If you don't feel comfortable granting these permissions, feel free to contact us and we will send you a private download link.

Using JSON Web Tokens in Your Applications



Browser
Server
1. POST /users/login with username and password
2. Creates a JWT with a secret
3. Returns the JWT to the Browser
4. Sends the JWT on the Authorization Header
5. Check JWT signature. Get user information from the JWT
6. Sends response to the client

Creating JSON Web Tokens

```
// sign with default (HMAC SHA256)
var jwt = require('jsonwebtoken');
var token = jwt.sign({ name: 'Ado Kukic' }, 'secret');
```

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJuYW1lIjoiQWRvIEt1a2ljIiwiaWF0IjoxNTAwNDk3NTcwfQ.AMe-ntiiiFeeaIbkdxRTvWyWRwu1QfhLhwPjGou7rtk

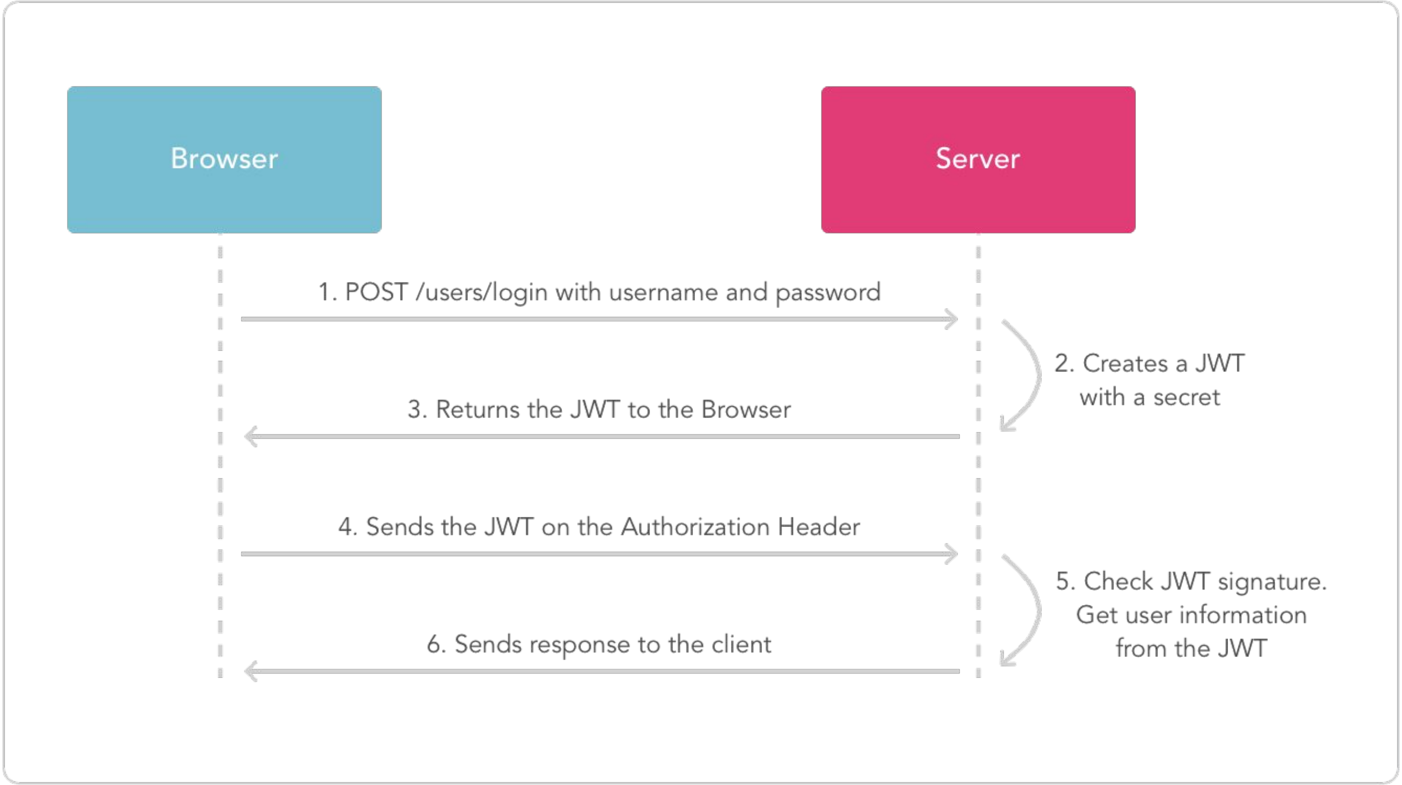
```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

```
{
  "name": "Ado Kukic",
  "iat": 1500497570
}
```

HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), secret)

Auth0

Using JSON Web Tokens in Your Applications



Browser
Server
1. POST /users/login with username and password
2. Creates a JWT with a secret
3. Returns the JWT to the Browser
4. Sends the JWT on the Authorization Header
5. Check JWT signature. Get user information from the JWT
6. Sends response to the client

Validating JSON Web Tokens

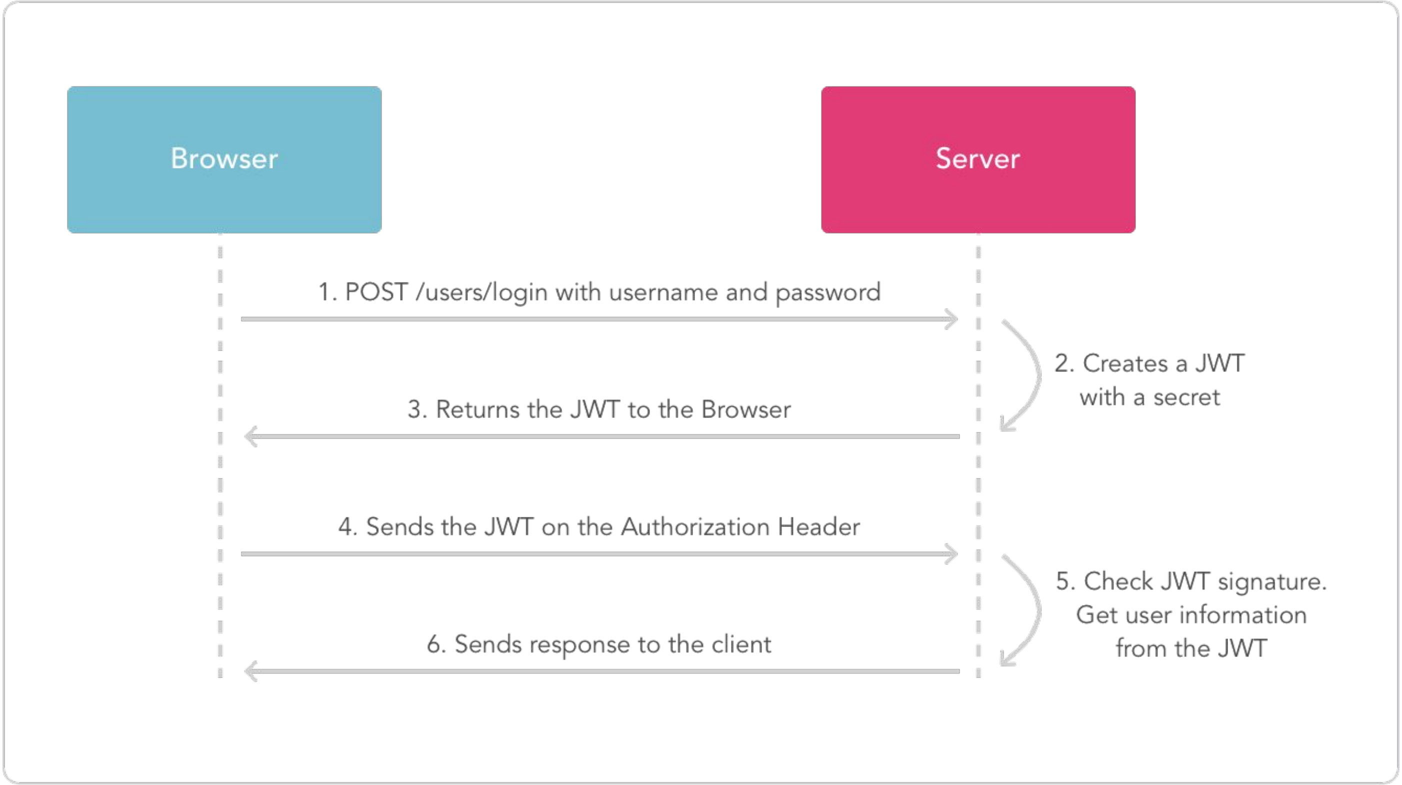
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJuYW1lIjoiQWRvIEt1a2ljIiwiaWF0IjoxNTAwNDk3NTcwfQ.AMe-ntiiiFeeaIbkdxRTvWyWRwu1QfhLhwPjGou7rtk

```
var jwt = require('jsonwebtoken');

// verify a token - synchronous
var decoded = jwt.verify(token, 'secret');
console.log(decoded.name) // Ado Kukic

// verify a token - async
jwt.verify(token, 'secret', function(err, decoded) {
 console.log(decoded.name) // Ado Kukic
});
```

Using JSON Web Tokens in Your Applications



Browser

Server

1. POST /users/login with username and password

2. Creates a JWT with a secret

3. Returns the JWT to the Browser

4. Sends the JWT on the Authorization Header

5. Check JWT signature. Get user information from the JWT

6. Sends response to the client

Advantages of Token Based Authentication

Stateless, **Scalable**, and **Decoupled**

- **Stateless** - the server does not have to keep a record of tokens issued
- **Scalable** - each JWT is self-contained, server only needs to validate the token
- **Decoupled** - your app server does not need to be the one that issues tokens

Advantages of Token Based Authentication

Cross Domain and **CORS**

- Working with cookies across multiple domains is a challenge
- JSON Web Tokens do not have any limitations due to their stateless nature

Advantages of Token Based Authentication

JWT Claims and Performance

- With traditional session based authentication, you simply store the session id which is then used to look up user details
- With token based authentication, you can store any type of data in the JWT (as long as it’s valid JSON)
- Storing relevant data in JWT at time of creation can greatly reduce future data lookups

Use Cases for Token Based Authentication

Application Modernization

- Single Page Applications like Angular, React, Vue, etc. that consume backend API’s
- Serverless Applications
- API’s
- Mobile Applications (both Native and Hybrid)
- Internet of Things (IoT) Devices

Conclusion

As always, use the right tool for the job

- If you are building a simple website with minimal security needs, JWT may be overkill
- Mobile apps, IoT, and Single Page Applications are perfect candidates for token based authentication
- Securing APIs with JWT is an industry standard
- **Thank You!**