

# Lesson: Authentication & Authorization

Wannes Fransen & Tom Eversdijk

UC Leuven

2021

Intro

Authentication

Authorization

# Difference authentication & authorization

## Authentication - AuthN

- ▶ Authentication is establishing the your identity
- ▶ *Verifies you are who you say you are*

## Authorization - AuthZ

- ▶ Authorization is establishing your privilege
- ▶ *Decides if you have permission to access a resource*

## Fun fact

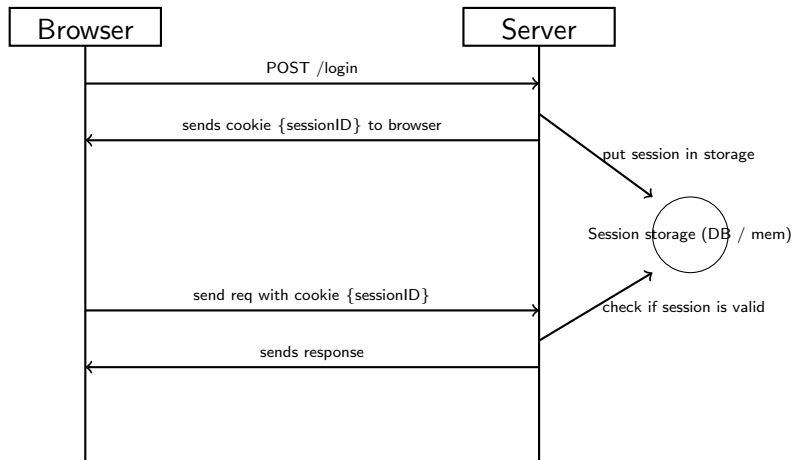
*broken authentication / authorization is nr2 on the most critical security risks to web apps.*

Intro

Authentication

Authorization

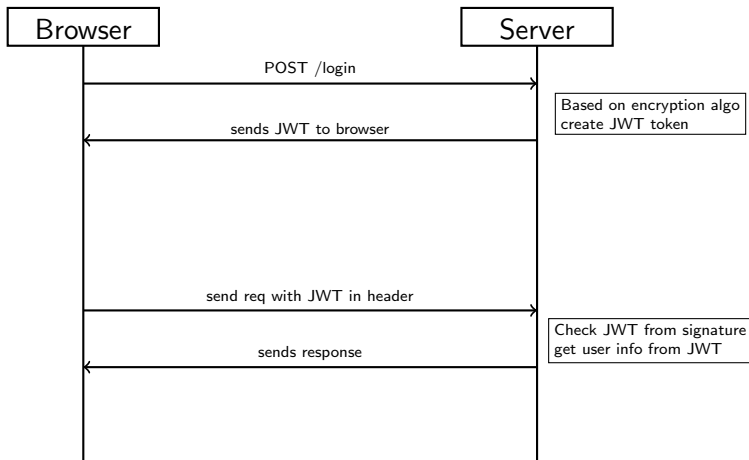
## Recap: Session based



# Session based characteristics

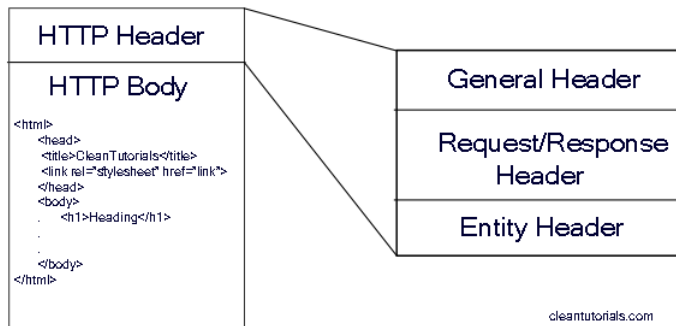
- ▶ Easy to implement stateful user sessions
- ▶ Can revoke at any time
- ▶ Can store any data alongside your session ID
- ▶ Can be a bottleneck in case of large volumes / data / traffic
- ▶ Serialization can be a problem (when storing objects)
- ▶ Session storage is a bottleneck in distributed contexts

# JWT based



# JWT: Where is it exactly?

## HTTP Request/response





# JWT: What is it exactly?

**JWT**  
JSON WEB TOKEN



**HEADER**  
ALGORITHM  
& TOKEN TYPE

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

+

**PAYLOAD**  
DATA

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "admin": true  
}
```

+

**SIGNATURE**  
VERIFICATION

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),secretKey)
```

NORDICAPIS.COM

# JWT: What is it exactly?

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX25hbWUiOiJzdW5pbHkiLCJ1c2VyX2VtYWlsIjoic3VuaWxAbm90c29zZW1cmUuY29tIiwiaXNzIjoiaHR0cDovL2xvY2FsaG9zdCI6ImF1ZCI6Im5nYXV0aCI6ImV4CI6MTQ2MTc2NDM3MSwibmJmIjoxNDYxNzYyNTcxZQ.YLDzU6FCPxJ3_c8RnTseXsDw00UfeUxYaQ9ItTN93DE
```

## HEADER: ALGORITHM & TOKEN TYPE

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

## PAYLOAD: DATA

```
{  
  "user_name": "sunily",  
  "user_email": "sunil@notsosecure.com",  
  "iss": "http://localhost",  
  "aud": "ngauth",  
  "exp": 1461764371,  
  "nbf": 1461762571  
}
```

## VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  secret  
) @secret base64 encoded
```

☑ Signature Verified

# JWT based characteristics

- ▶ no central bottleneck anymore
- ▶ allows horizontal scaling
- ▶ built-in expiration functionality
- ▶ no way to revoke the JWT token
- ▶ self-contained, but keep it short

# What to use?

- ▶ Both are viable, depends on your needs
- ▶ We'll go with JWT (Guardian library) as this can be re-used in Distributed Applications

Intro

Authentication

Authorization

# Usages

- ▶ Roles (Superuser, forum maintainer, normal user)
- ▶ User-specific resources (You manage your own resources)
- ▶ permissions

# Libraries

- ▶ Different libraries have their own implementation
- ▶ Same as a mindset, different ways to achieve the same result

## extra references

- ▶ Bodyguard can be used to implement Authorization in Elixir