Lesson 7: Authentication & Authorization

Wannes Fransen & Tom Eversdijk

UC Leuven

2021

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
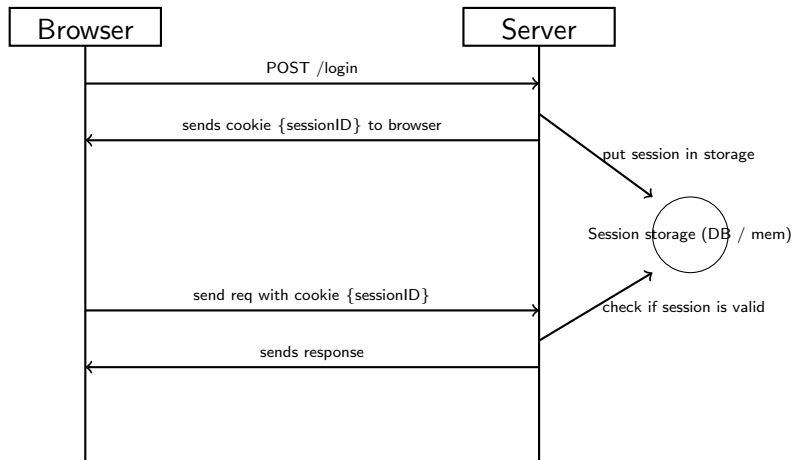- ▶ *Decides if you have permission to access a resource*

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

Recap: Session based

Browser → Server: POST /login

Server → Browser: sends cookie {sessionID} to browser

Server: put session in storage → Session storage (DB / mem)

Browser → Server: send req with cookie {sessionID}

Server: check if session is valid

Server → Browser: sends response
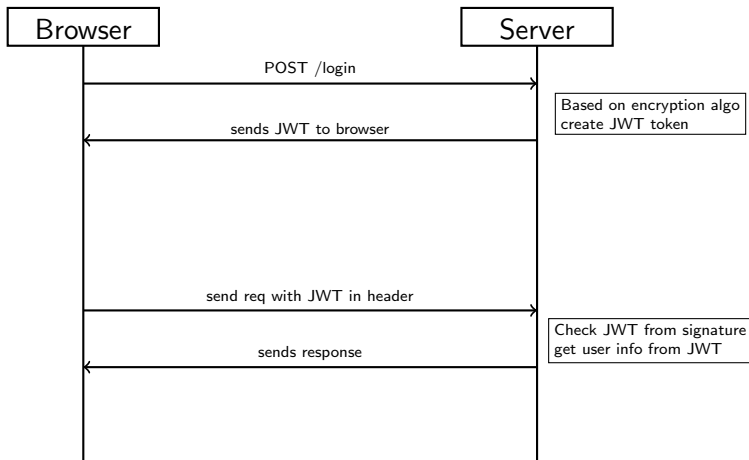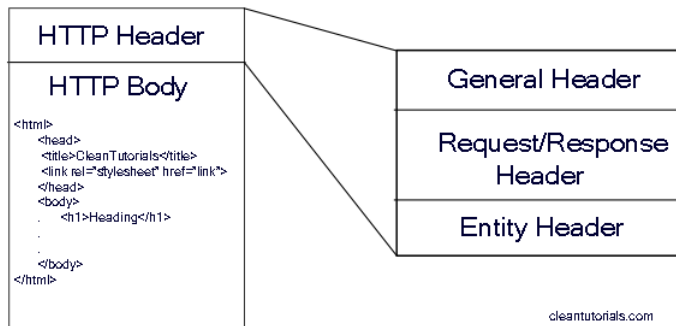
Session based characteristics

- 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

Browser   Server

POST /login

Based on encryption algo
create JWT token

sends JWT to browser

send req with JWT in header

Check JWT from signature
get user info from JWT

sends response

JWT: Where is it exactly?

HTTP Request/response

HTTP Header

HTTP Body

```
<html>
    <head>
    <title>CleanTutorials</title>
    <link rel="stylesheet" href="link">
    </head>
    <body>
    .    <h1>Heading</h1>
    .
    .
    </body>
</html>
```

General Header

Request/Response Header

Entity Header

cleantutorials.com

JWT: What is it exactly?

JWT: What is it exactly?

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1
c2VyX25hbWUiOiJzdW5pbHkiLCJ1c2VyX2VtYWlsI
joic3VuWxAbm90c29zZWN1cmUuY29tTiwiaXNzIj
oiaHR0cDovL2xvY2FsaG9zdCIsImF1ZCI6Im5nYXV
0aCIsImV4cCI6MTQ2MTc2NDM3MSwibmJmIjoxNDYx
NzYyNTcxfQ.YLDzU6FCPxJ3_c8RnTseXsDwO0UfeU
xYaQ9ItTN93DE

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

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