

Secure Web Development

Authentication & Authorization

ESILV

Roch Moreau - 2022

Authentication & Authorization

- Definitions
- Authentication
- Authorization
- Implementation
- More: 2FA, OAuth2...



Definitions

Definitions

Authentication & Authorization

- “**Authentication** is the act of proving an assertion, such as the identity of a computer system user” (Wikipedia)
- “**Authorization** [...] is the function of specifying access rights/privileges to resources” (Wikipedia)

Authentication

Authentication

How to prove the identity of a user ?

User proves their identity by:

- **Password authentication:** proving their knowledge of a secret password
- **Token authentication:** giving a token signed by a trusted authentication system
- **Biometric authentication:** showing unique biometric characteristic
- **Certificate-based authentication:** using a digital certificate delivered by a trusted certificate authority
- **MFA/Multi-Factor authentication:** using multiple authentication method above

Authentication

Risks and mitigations

- **Password authentication:**

- Stealing: Hard to mitigate, user's responsibility to use unique passwords. Never store plain-text password. Avoid needing to transfer the password (only once when possible)
- Brute-force: Rate-limiting API requests. No mitigation possible on stolen database data
- Rainbow-table attacks (pre-bruteforced password list): Use salt when hashing passwords

- **Token authentication:**

- Stealing: Use short-lived token with refresh token
- Token-key leak: Change key ASAP

Authorization

Authorization

How to define authorizations of a user ?

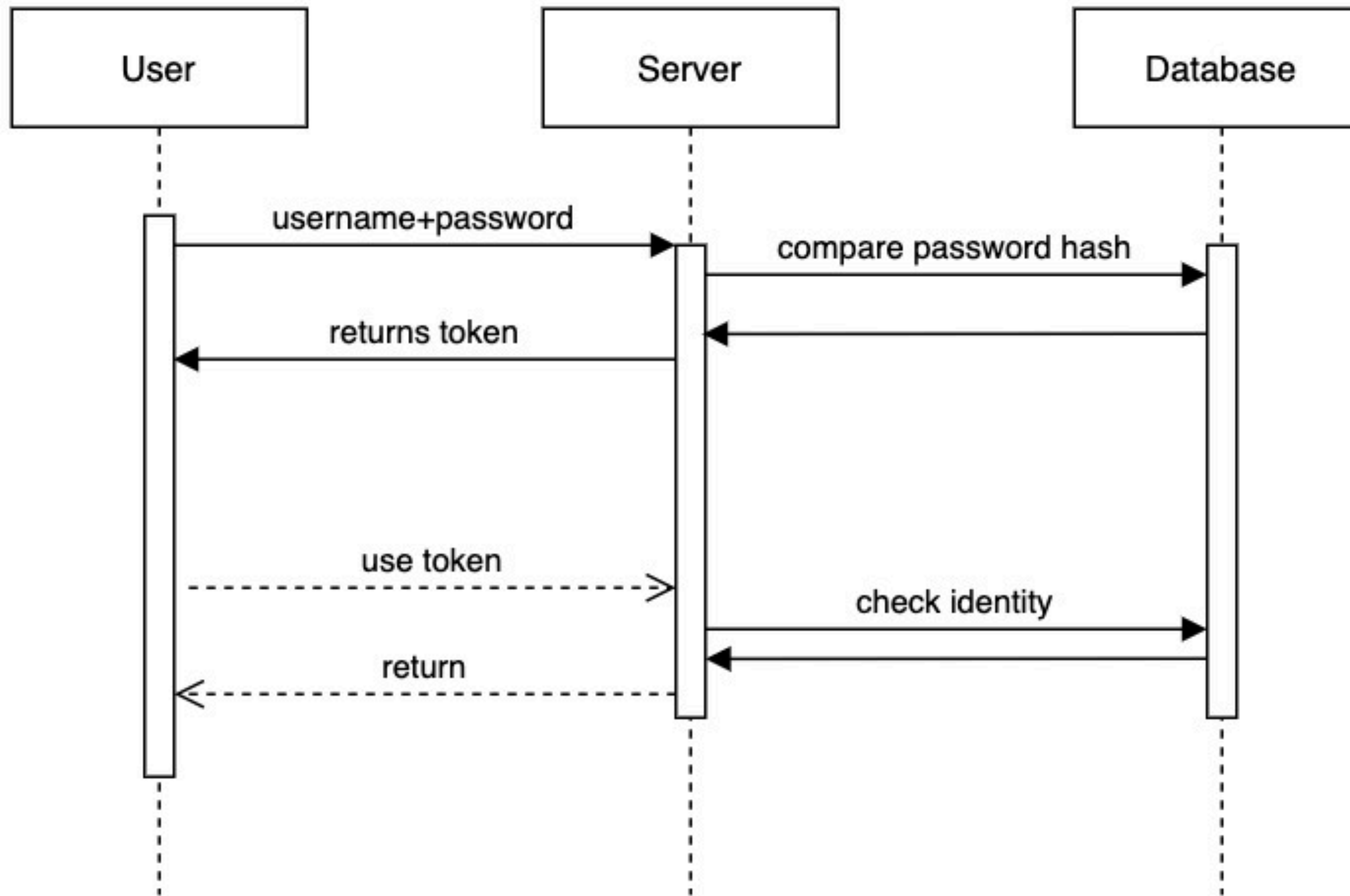
Allow user access to ressources based on:

- **Role-based access control (RBAC):** user's group (admin, moderator etc...)
- **Policy-based access control (PBAC):** user privileges on resources policies (each resource have a policy, users can have multiple policies enabled)

Implementation

Implementation

Authentication Flow



Implementation

Password Auth Using Passport Middleware

Login URL



```
app.post( path: '/login/password',  
  passport.authenticate('local', { failureRedirect: '/login', failureMessage: true } ),  
  async function (req :..., res :...) {  
    const jwt = await userService.signJwt(req.user);  
    res.status( code: 200 ).send( body: { token: jwt } );  
  } );
```

Authentication
(Passport Middleware)

JWT Generation

Implementation

Token (JWT) Auth Using Passport Middleware

Resource URL



```
app.post( path: '/locations',  
  passport.authenticate('jwt', { failureRedirect: '/login', failureMessage: true } ),  
  async function (req: ..., res: ...) {  
    const newLocation = await locationsService.create(req.body);  
    res.status( code: 200 ).send(newLocation);  
  } );
```

Authentication
(Passport Middleware)



Implementation

Token (JWT) Auth + RBAC

Resource URL



```
app.post( path: '/locations',  
  passport.authenticate('jwt', { failureRedirect: '/login', failureMessage: true } ),  
  authService.canAccess( role: ['admin'] ),  
  async function (req :... , res :... ) {  
    const newLocation = await locationsService.create(req.body);  
    res.status( code: 200 ).send(newLocation);  
  }  
);
```

← Authentication

← RBAC Authorization

Implementation

Token (JWT) Auth + PBAC

Resource URL



```
app.post(path: '/locations',
  passport.authenticate('jwt', { failureRedirect: '/login', failureMessage: true }),
  authService.canAccess(policy: 'locations.create'),
  async function (req:..., res:...) {
    const newLocation = await locationsService.create(req.body);
    res.status(code: 200).send(newLocation);
  });
```

Authentication

PBAC Authorization