Json Web Token

1. Despre JWT

Un JSON Web Token este un standard de transmitere a informației, semnată digital, sub forma de obiect JSON, . Structura acestuia este formată din:

1. **Header**
   Este un JSON cu 2 proprietăți: tipul algoritmului folosit la criptare și tipul tokenului. De exemplu acest obiect identifică tipul obiectului ca fiind un JWT, iar algoritmul de criptare este un HMAC cu SHA-256

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

2. **Payload**
   Este un JSON cu mai multe proprietăți, care pot fi puse de programator, în funcție de necesități, precum și altele **standard** precum:
   - iss (Issuer): entitatea care a semnat JWT-ul
   - sub (Subject): subiectul JWT-ului
   - aud (Audience): entitatea căreia îi este destinat JWT-ul
   - exp (Expiration Time): momentul de timp la care expiră JWT-ul (număr de secunde de la 1 ianuarie 1970)
   - nbf (Not Before): momentul de timp de la care poate fi acceptat JWT-ul
   - iat (Issued at): momentul de timp în care a fost creat JWT-ul
   - jti (JWT ID): identificator unic al JWT-ului dintre mai mulți issueri

   Avem următorul exemplu:

```
{
  "userId": 2,
  "sub" : "authorization",
  "aud" : "simple_user",
  "iat": 1619362177,
  "exp": 1619362537
}
```

3. **Semnătura**
   Se calculează prin concatenarea header-ului si a payload-ului (codificate fiecare în Base64UrlEncoding, despărțite prin ‘.’), iar asupra textului obținut se aplică algoritmul specificat în **header** (în cazul de față HMAC-SHA256) și se folosește cheia secretă.

Cele 3 părți se codifică fiecare în formatul Base64UrlEncoding, iar rezultatele se concatenează, fiind despărțite de ‘.’ De exemplu, tokenul final arată astfel:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
J1c2VySWQiOjIsInN1YiI6ImF1dGhvcml6YXRpb
24iLCJhdWQiOiJzaW1wbGVfdXNlciIsImlhdCI6
MTYxOTM2MjE3NywiZXhwIjoxNjE5MzYyNTM3fQ.
eKd61LsgdQuvgWbVUaBZAGzSr5r1QmUx9Tv_3XK
KBSo
```

Pentru partea de **validare** a tokenului, se obține algoritmul de criptare (prin decodificarea header-ului), se concatenează primele 2 blocuri (header-ul si payload-ul), despărțite prin '.', după care rezultatul se trece prin algoritmul de criptare cu cheia secretă. Dacă semnătura coincide cu rezultatul final, atunci tokenul este valid.

2.Utilizarea

Am ales să utilizez librăria **jsonwebtoken** pe care am folosit-o în proiectul de Licență pentru partea de autentificare și autorizare. Mai precis, în momentul în care un user se autentifică, este trimis un request la server cu datele acestuia în vederea stabilirii identității sale.

În program, este importată librăria:

```
1   const jwt = require('jsonwebtoken')
```

iar stabilirea identității userului are loc in funcția de **login**:

```
19      static login = async (data) => {
20          const user = await models.User.findOne({
21              where: {
22                  username: Encrypt.encryptData(data.username)
23              }
24          });
25
26          if (user) {
27              if (Encrypt.hashData(data.password) == user.password) {
28                  const token = jwt.sign({ userId: user.id }, process.env.JWT_SECRET, {
29                      expiresIn: '360s'
30                  });
31                  return token;
32              }
33          }
34          throw new Error(errors.WRONG_CREDENTIALS, httpCodes.UNAUTHORIZED);
35      }
```

Se caută in baza de date userul după username, după care se compară parola introdusă cu cea memorată în baza de date (parolele fiind hash-uite în baza de date, este necesară trecerea parolei prin aceeași functie hash). În cazul în care cele 2 hash-uri corespund, atunci este generat cu metoda **sign** un **JSON Web Token** (cu ajutorul unei chei secrete, păstrată ca variablă de environment sub denumirea de JWT_SECRET) care conține în payload id-ul userului curent, momentul de timp la

care a fost generat tokenul si momentul de timp la care expiră acesta (dupa 360 de secunde de la generare în cazul de față).

După, pentru fiecare request făcut către server, userul atașează în header-ul de autorizare tokenul obținut pentru verificarea identității. Se folosește metoda **verify** care verifică integritatea tokenului (a fost semnat cu aceeași cheie secretă, nu este expirat și nu i-au fost aduse modificări), iar rezultatul este variabila **data** care reprezintă json-ul inițial. În cazul de față, verificarea tokenului se face pe un middleware, iar în caz pozitiv, se atașează pe conținutul requestului primit de server id-ul user-ului și aceasta este forma finală a requestului care ajunge in backend-ul serverului.

```
5  const authenticationMiddleware = (req, res, next) => {
6    const token = req.headers.authorization ? req.headers.authorization.replace('Bearer ', '') : null;
7    /*  Check if the token is valid. If so, append the userId to the request*/
8    jwt.verify(token, process.env.JWT_SECRET, async (err, data) => {
9      if (err) {
10       res
11         .status(httpCodes.UNAUTHORIZED)
12         .send(errors.SESSION_TIMED_OUT);
13     } else {
14       req.userId = data.userId;
15       next();
16     }
```

3.Motivarea

Folosirea acestei librării este dată de necesitatea obținerii unui cod de autorizare (access token) prin autentificarea utilizatorului și necesitatea restricției accesului la resursele din baza de date doar către persoanele autorizate (autorizarea). Astfel, atunci când un user face un anumit request către server (de exemplu, obținerea de detalii despre un anumit cont bancar), este necesar un access token (în cazul de față, sub forma unui JWT) prin care serverul poate identifica persoana care inițiază cererea și validarea accesului persoanei respective la contul respectiv. Puterea conceptului de JWT este dată de **puterea algoritmilor** prin care are loc semnarea conținutului**.** Având un JWT, conținutul payload-ului este vizibil pentru toată lumea care-l deține, deci **NU** este recomandat ca acesta să conțină date senzitive (parole, numărul cardului de credit etc), deci partea de validare și stabilire a integrității sale se realizează prin validarea semnăturii (al 3-lea bloc din JWT).

Bibliografie:

- https://en.wikipedia.org/wiki/JSON_Web_Token
- https://jwt.io/
- https://tools.ietf.org/html/rfc7519
- https://auth0.com/docs/tokens/json-web-tokens