Programmation Web – Avancé

JavaScript & Node.js

Sécurisation de RESTful APIs : Hachage d'information et protection contre les attaques XSS

Sécurisation des passwords

Comment assurer que les passwords enregistrés dans la DB ne puissent pas être récupérés ?

Hacher les passwords côté serveur ?

```
[
  {
    "username": "admin",
    "password": "$2b$10$RqcgWQT/Irt9MQC8
UfHmjuGCrQkQNeNcU6UtZURdSB/fyt6bMWARa"
  }
]
```

- Recommandé
- Défense contre les « hash attacks » :
  - Salt
  - Salt round : nombre de fois que la hashage est fait
- Hachage sous Node.js : **bcrypt** [89.]
- Installation & initialisation : **npm install bcrypt**

```
const bcrypt = require("bcrypt");
const saltRounds = 10;
```

Hacher un password via bcrypt

- Existence de méthodes asynchrones ou synchrones : **hash()**, **hashSync()**, **compare()**, **compareSync()**
- Hasher le passsword : **hash()**

```
async addOne(body) {
    const items = parse(this.jsonDbPath, this.defaultItems);
    const hashedPassword = await bcrypt.hash(body.password, saltRounds);
    const newitem = {username: body.username, password: hashedPassword, };
    items.push(newitem);
    serialize(this.jsonDbPath, items);
    return newitem;  }
```

Comparer un password haché via bcrypt

- Comparer le password : **compare()**

```
async login(username, password) {
    const userFound = this.getOneByUsername(username);
    if (!userFound) return;
    const match = await bcrypt.compare(password, userFound.password);
    if (!match) return;
    const authenticatedUser = {username: username, token: "Future signed token",};
    const token = jwt.sign(
      { username: authenticatedUser.username }, jwtSecret,
      { expiresIn: LIFETIME_JWT });
    authenticatedUser.token = token;
    return authenticatedUser;  }
```

Garder son password crypté ou hashé côté client ?

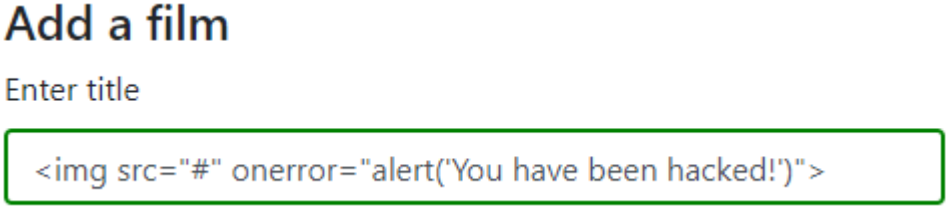
- Non recommandé

```
  {
    "username": "teacher@vinci.be",
    "password": "$2b$10$yS7G6Ruhw9o.d47E
ZVM4v.UMzf2HDLFSbFM1kMS9mA/h4oq/.LiSW"
  }
```

Protection contre les attaques XSS

Comment faire pour ne pas autoriser de JavaScript malicieux qui pourrait être lu via nos RESTful APIs ?

Echapper les caractères dangereux



- Caractères dangereux ?
  ", ', &, <, et >
- Librairie sous Node.js échappant les String pour utilisation en HTML : **escape-html** [112.]

```
var escape = require("escape-html");
const newPizza = {
      id: this.getNextId(),
      title: escape(body.title),
      content: escape(body.content), };
```

Attaque XSS


https://forum.my-bank.com
GET or POST request
Session=ejwz5536x…
https://api.my-bank.com
https://forum.my-bank.com
GET or POST request
Session=ejwz5536x…
https://api.malicious.com

Hacher l'information côté serveur

- DEMO : Création d'une RESTfull API pour une pizzeria : Part 6 – Sécurisation : hachage des passwords et protection contre attaques XSS