Rapport de Sécurité - SR10

Léopold Chappuis - Alexandre Eberhardt

1 Injection SQL

1.1 Démonstration de l'Attaque

L'injection SQL permet à un attaquant d'insérer du code SQL malveillant dans une requête via des entrées utilisateur non sécurisées. Voici un exemple en Express.js :

```
const express = require('express');
const app = express();
const mysql = require('mysql');

const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: 'password',
  database: 'test'
});

app.get('/user', (req, res) => {
  const userId = req.query.id;
  connection.query(`SELECT * FROM users WHERE id = ${userId}`, (err
    , results) => {
    if (err) throw err;
    res.send(results);
  });
});

app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

Listing 1: Code vulnérable à l'injection SQL

1.2 Solution: Échappement des Entrées

Pour se protéger contre l'injection SQL, il est essentiel d'utiliser des requêtes préparées ou d'échapper les entrées utilisateur. Voici comment corriger le code ci-dessus :

```
app.get('/user', (req, res) => {
  const userId = req.query.id;
  connection.query('SELECT * FROM users WHERE id = ?', [userId], (
    err, results) => {
```

```
    if (err) throw err;
    res.send(results);
  });
});
```

Listing 2: Code protégé contre l'injection SQL

L'utilisation des requêtes préparées empêche l'injection SQL en séparant les données des instructions SQL.

2 Hash des Mots de Passe

2.1 Démonstration de l'Attaque

Le stockage des mots de passe en clair est une faille de sécurité majeure. Si un attaquant obtient l'accès à la base de données, il peut voir tous les mots de passe. Voici un exemple de code vulnérable :

```
app.post('/register', (req, res) => {
  const { username, password } = req.body;
  connection.query('INSERT INTO users (username, password) VALUES
    (?, ?)', [username, password], (err, results) => {
    if (err) throw err;
    res.send('User registered');
  });
});
```

Listing 3: Code vulnérable avec stockage en clair des mots de passe

2.2 Solution: Hachage des Mots de Passe

Pour protéger les mots de passe, il est recommandé d'utiliser un algorithme de hachage. Voici un exemple utilisant bcrypt :

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

app.post('/register', (req, res) => {
  const { username, password } = req.body;
  bcrypt.hash(password, saltRounds, (err, hash) => {
    if (err) throw err;
    connection.query('INSERT INTO users (username, password) VALUES
     (?, ?)', [username, hash], (err, results) => {
      if (err) throw err;
      res.send('User registered');
    });
  });
});
```

Listing 4: Code sécurisé avec hachage des mots de passe

Le hachage des mots de passe empêche les attaquants de lire les mots de passe en clair en cas de fuite de la base de données.

3 Protection des Routes

3.1 Démonstration de l'Attaque

Les routes non protégées permettent à un utilisateur non autorisé d'accéder à des ressources sensibles. Voici un exemple de route vulnérable :

```
app.get('/admin', (req, res) => {
  res.send('Welcome to the admin area');
});
```

Listing 5: Route non protégée

3.2 Solution: Vérification des Accès

Pour protéger les routes, il est essentiel de vérifier les autorisations de l'utilisateur. Voici un exemple avec un middleware de vérification :

```
function isAuthenticated(req, res, next) {
  if (req.isAuthenticated()) {
    return next();
  }
  res.redirect('/login');
}

app.get('/admin', isAuthenticated, (req, res) => {
  res.send('Welcome to the admin area');
});
```

Listing 6: Route protégée avec vérification des accès

L'utilisation de middleware permet de s'assurer que seules les personnes autorisées peuvent accéder aux routes sensibles.

4 Conclusion

En implémentant des pratiques de sécurité comme l'échappement des entrées pour les requêtes SQL, le hachage des mots de passe et la vérification des accès, nous pouvons considérablement améliorer la sécurité de notre application Express.js. Ces mesures protègent contre les attaques courantes et assurent la confidentialité et l'intégrité des données utilisateur.