RPG0035 - SOFTWARE SEM SEGURANÇA NÃO SERVE!

Alessandro Sendi Shigematsu Matrícula 202208809812

Polo Parque Andorinhas

2024.3 Desenvolvimento Full Stack – 2o Semestre Letivo

Objetivo da Prática

- Descrever o controle básico de acesso a uma API Rest;
- Descrever o tratamento de dados sensíveis e log de erros com foco em segurança;
- Descrever a prevenção de ataques de acesso não autorizado com base em tokens desprotegidos/desatualizados;
- Descrever o tratamento de SQL Injection em códigos-fonte; Descrever o tratamento de CRLF Injection em códigos-fonte;
- Descrever a prevenção a ataques do tipo CSRF em sistemas web;

Missão pratica.

Instalação das Dependências:
- As dependências `express`, `body-parser` e `jsonwebtoken` foram instaladas para gerenciar o servidor e os tokens JWT.

Middleware `authenticateToken`:
- Verifica a presença e validade do token JWT nos headers das requisições.

Endpoint de Login:
- Gera um token JWT para o usuário autenticado.

Endpoint para Dados do Usuário Logado:
- Retorna os dados do usuário logado, sem restrições de perfil.

Endpoints Protegidos:
- Endpoints `/api/users` e `/api/contracts/:empresa/:inicio` são protegidos pelo middleware de autenticação e verificação de perfil.

Proteção Contra SQL Injection:
- Sanitização básica dos parâmetros na função `getContracts` para prevenir SQL Injection.

Código Refatorado

```
const express = require('express');
const bodyParser = require('body-parser');
const jwt = require('jsonwebtoken');

const app = express();
app.use(bodyParser.json());

const port = process.env.PORT || 3000;
const JWT_SECRET = 'your_jwt_secret_key'; // Chave secreta para assinar os tokens JWT
const JWT_EXPIRATION = '1h'; // Tempo de expiração dos tokens

// Mock de dados
const users = [
  { "username": "user", "password": "123456", "id": 123, "email": "user@dominio.com", "perfil": "user" },
  { "username": "admin", "password": "123456789", "id": 124, "email": "admin@dominio.com", "perfil": "admin" },
```

```
  { "username": "colab", "password": "123", "id": 125, "email": "colab@dominio.com",
"perfil": "user" },
];

// Middleware para verificar o token JWT
function authenticateToken(req, res, next) {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];

  if (!token) {
    return res.status(401).json({ message: 'Não autorizado' });
  }

  jwt.verify(token, JWT_SECRET, (err, user) => {
    if (err) {
      return res.status(403).json({ message: 'Token inválido ou expirado' });
    }
    req.user = user;
    next();
  });
}

// Endpoint para login do usuário
app.post('/api/auth/login', (req, res) => {
  const { username, password } = req.body;
  const user = users.find(u => u.username === username && u.password === password);

  if (!user) {
    return res.status(401).json({ message: 'Credenciais inválidas' });
  }

  const token = jwt.sign({ id: user.id, perfil: user.perfil }, JWT_SECRET, { expiresIn:
JWT_EXPIRATION });
  res.json({ token });
});

// Endpoint para dados do usuário logado
app.get('/api/auth/me', authenticateToken, (req, res) => {
  const user = users.find(u => u.id === req.user.id);
  res.json({ user });
});

// Endpoint para recuperação dos dados de todos os usuários cadastrados
app.get('/api/users', authenticateToken, (req, res) => {
  if (req.user.perfil !== 'admin') {
    return res.status(403).json({ message: 'Forbidden' });
```

```
  }
  res.json({ data: users });
});

// Endpoint para recuperação dos contratos existentes
app.get('/api/contracts/:empresa/:inicio', authenticateToken, (req, res) => {
  const { empresa, inicio } = req.params;
  const result = getContracts(empresa, inicio);
  if (result.length > 0) {
    res.status(200).json({ data: result });
  } else {
    res.status(404).json({ message: 'Dados Não encontrados' });
  }
});

// Mock da função getContracts com proteção contra SQL Injection
function getContracts(empresa, inicio) {
  const repository = new Repository();
  // Sanitização básica para prevenir SQL Injection
  const sanitizedEmpresa = empresa.replace(/[^a-z0-9]/gi, '');
  const sanitizedInicio = inicio.replace(/[^a-z0-9-]/gi, '');
  const query = `SELECT * FROM contracts WHERE empresa = ? AND data_inicio =
?`;
  const result = repository.execute(query, [sanitizedEmpresa, sanitizedInicio]);
  return result;
}

// Mock da classe Repository
class Repository {
  execute(query, params) {
    // Implementação fictícia
    console.log('Executing query:', query, 'with params:', params);
    return []; // Retorna resultados fictícios
  }
}

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