Tecnólogo em Desenvolvimento Full-Stack

Relatório da missão prática Software Sem Segurança Não Serve

Aluno: Ricardo Alves dos Santos Junior

15/06/2024

Jaraguá do Sul/SC

Tecnólogo em Desenvolvimento Full-Stack

Relatório da missão prática Software Sem Segurança Não Serve

Aluno: Ricardo Alves dos Santos Junior

15/06/2024

Jaraguá do Sul/SC

Documentação da API - Sistema de Login e Recuperação de Dados

Visão Geral

Este projeto é uma API desenvolvida utilizando **Node.js**, que permite a realização de login, autenticação de usuários, recuperação de dados e geração de tokens JWT para autenticação de acesso aos endpoints. A aplicação também refatora práticas de segurança, como criptografia e controle de acesso.

Tecnologias Usadas

- **Node.js**: Ambiente de execução JavaScript do lado do servidor.
- **Express.js**: Framework minimalista para Node.js, utilizado para construir a API.
- **JWT (JSON Web Token)**: Tecnologia para gerar e validar tokens de autenticação.
- **Crypto**: Módulo nativo do Node.js para realizar criptografia e descriptografia de dados.
- **Body-parser**: Middleware para processar corpos de requisições JSON.

Estrutura do Projeto

1. Endpoints da API

A API disponibiliza os seguintes endpoints principais:

POST /api/auth/login

- **Descrição**: Endpoint responsável pelo login do usuário. Recebe as credenciais (nome de usuário e senha), válida e gera um token JWT para autenticação.
- **Parâmetros (body)**:
    - `username`: Nome de usuário.
    - `password`: Senha do usuário.
- **Resposta**:

    - **200 OK**: Retorna o token JWT no campo *sessionid*.
    - **400 Bad Request**: Caso as credenciais sejam inválidas.

Exemplo de Requisição (POST)

```
{
  "username": "user",
  "password": "123456"
}
```

Exemplo de Resposta

```
{
"sessionid":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTIzLCJwZX
JmaWwiOiJ1c2VyIiwiaWF0IjoxNzMxOTc3MjQyLCJleHAiOjE3MzE5OD
A4NDJ9.fnF5qxTsBl8LmyWxEGaWucIZi4p77vQia1-QLwxa_1I"
}
```

GET /api/me

- **Descrição**: Endpoint para recuperar os dados do usuário logado. Requer um token JWT para autenticação no cabeçalho.
- **Parâmetros**: Nenhum.
- **Headers**:
    - *Authorization*: O token JWT é passado como valor no cabeçalho (formato *Bearer <token>*).
- **Resposta**:
    - **200 OK**: Retorna os dados do usuário logado.
    - **401 Unauthorized**: Caso o token seja inválido ou ausente.

GET /api/users/

- **Descrição**: Recupera os dados de todos os usuários cadastrados na aplicação. Este endpoint é acessível apenas para administradores.
- **Parâmetros**:
    - *sessionid*: Token de autenticação no formato JWT.
- **Resposta**:
    - **200 OK**: Retorna a lista de todos os usuários.

- **403 Forbidden**: Caso o usuário não tenha o perfil *admin*.

GET /api/contracts/

- **Descrição**: Recupera os contratos cadastrados na aplicação. Este endpoint recebe parâmetros específicos para realizar a busca.
- **Parâmetros**:
  - *empresa*: Nome da empresa.
  - *inicio*: Data de início dos contratos.
  - *sessionid*: Token de autenticação.
- **Resposta**:
  - **200 OK**: Retorna os contratos encontrados.
  - **404 Not Found**: Caso não sejam encontrados contratos.

2. Componentes da Aplicação

Autenticação (JWT)

A autenticação é feita utilizando JWT (JSON Web Token), que é gerado durante o login do usuário. O token contém informações codificadas sobre o usuário e seu perfil, permitindo a validação do acesso a diferentes recursos da API.

Criptografia

No exemplo original, a criptografia foi realizada com a chave da empresa, porém foi refatorada para usar JWT, o que melhora a segurança, pois o JWT não depende de um padrão simples de encriptação.

Controle de Acesso

A aplicação agora possui controle de acesso baseado no perfil do usuário. Somente usuários com o perfil *admin* podem acessar dados sensíveis como informações de todos os usuários ou contratos. Os endpoints de recuperação de dados verificam o perfil do usuário antes de liberar as informações.

Sanitização de Parâmetros

Foi implementada uma proteção contra ataques de injeção de código (como SQL Injection) nos parâmetros recebidos. Ao utilizar consultas ao banco de dados, os parâmetros são devidamente validados e limpos.

3. Como Utilizar a Aplicação

Passo 1: Instalação

Clonar o Repositório Clone o repositório do projeto:
git clone https://github.com/RicardoASJunior/software_sem_seguranca_nao_serve.git

1. **Instalar Dependências** Instale as dependências do projeto:
   npm install express body-parser jsonwebtoken dotenv
2. **Iniciar o Servidor** Para rodar o servidor localmente, execute:
   `node server,js`
3. O servidor estará rodando na porta `3000` por padrão.

Passo 2: Realizar o Login

Envie uma requisição POST para o endpoint `/api/auth/login` com as credenciais do usuário:

```
{
  "username": "user",
  "password": "123456"
}
```

1. A resposta retornará o token JWT:

```
{

"sessionid":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTIzLCJwZX
JmaWwiOiJ1c2VyIiwiaWF0IjoxNzMxOTc3MjQyLCJleHAiOjE3MzE5OD
A4NDJ9.fnF5qxTsBl8LmyWxEGaWucIZi4p77vQia1-QLwxa_1I"
}
```

Passo 3: Acessar Dados Protegidos

Para acessar os dados do usuário logado, envie uma requisição GET para `/api/me`, incluindo o token no cabeçalho:

```
GET /api/me
```

`Authorization: Bearer <seu-token-jwt>`

1. Para acessar outros endpoints protegidos (como `/api/users` ou `/api/contracts`), insira o token JWT no cabeçalho da requisição.

Passo 4: Testes

Utilize ferramentas como **Postman** ou **Insomnia** para enviar as requisições e testar a API. Lembre-se de incluir o token JWT no cabeçalho **Authorization** de todas as requisições após o login.

Conclusão

Com as melhorias implementadas (uso de JWT, controle de acesso e sanitização de parâmetros), a aplicação agora é mais segura contra ataques de injeção, além de permitir uma autenticação mais robusta. O fluxo básico de autenticação e autorização está bem definido, tornando o uso da API simples e eficiente.