Universidade: Estácio de Sá

Campus: Estácio Interlagos

Curso: Desenvolvimento Full Stack

Disciplina: Software sem segurança não serve!

Número da Turma: 2024.3

Semestre Letivo: 5° Semestre

Integrante da Prática: Lucas Henrique Silva Santos

Introdução

Durante a realização da Missão Prática, dediquei-me à refatoração de uma aplicação Node.js com o objetivo de corrigir falhas críticas de segurança, melhorar sua organização e implementar melhores práticas de desenvolvimento. As vulnerabilidades identificadas comprometiam seriamente a confiabilidade do sistema, expondo-o a riscos como quebra de sessão, injeção de SQL e acessos indevidos. Neste relatório, apresento as alterações realizadas, justificativas e os resultados alcançados.

Etapas da Refatoração

1. **Substituição do Gerenciamento de Sessões** A aplicação utilizava um esquema de `session-id` baseado em uma criptografia frágil e previsível, que poderia ser quebrada com facilidade. Para corrigir isso, implementei o uso de **JWT (JSON Web Tokens)**. Com o JWT:
    a. O token inclui informações do usuário, como ID e perfil, assinadas digitalmente para garantir autenticidade.
    b. Tokens possuem tempo de expiração, limitando a janela de ataque em caso de comprometimento.
    c. A chave secreta para geração e validação foi movida para uma variável de ambiente, reduzindo a exposição a invasores.

Justificativa: Essa abordagem garante sessões seguras e facilita a verificação de permissões.

2. **Sanitização e Validação de Dados** Identifiquei que os endpoints não realizavam validação ou sanitização dos parâmetros recebidos, o que os tornava suscetíveis a ataques de **SQL Injection**. Para mitigar esse risco:
    a. Adotei parâmetros preparados nas consultas ao banco de dados.
    b. Integrei validações com a biblioteca `express-validator` para garantir que os dados recebidos estivessem no formato esperado.

Exemplo de validação: No endpoint de contratos, os parâmetros `empresa` e `inicio` agora são validados antes da execução da query.

Justificativa: Isso evita ataques que poderiam comprometer dados sensíveis no banco.

3. **Controle de Acesso Rigoroso** O sistema permitia acessos não autorizados a dados sensíveis devido à ausência de verificações adequadas. Para resolver isso:
    a. Implementei um middleware de autenticação e autorização que verifica o perfil do usuário diretamente no JWT.
    b. Apliquei esse middleware nos endpoints que requerem acesso restrito, como `/api/users` e `/api/contracts`.

Justificativa: Essa medida impede que usuários comuns acessem informações restritas e mantém a integridade dos dados.

4. **Organização e Modularização do Código** O código original era monolítico e violava o princípio de responsabilidade única, dificultando sua manutenção. Reorganizei a aplicação utilizando o padrão **Model-View-Controller (MVC)**:
    a. **Controllers:** Para gerenciar as requisições e respostas.
    b. **Services:** Para encapsular a lógica de negócios.
    c. **Routes:** Para definir as rotas e associá-las aos controllers.

Justificativa: A modularização melhora a legibilidade, facilita a manutenção e torna o sistema escalável.

5. **Melhoria dos Endpoints**
    a. **Login (`/api/auth/login`):** Substituí a geração de `session-id` por JWT. Agora, o login retorna um token seguro com expiração.
    b. **Usuários (`/api/users`):** Adicionei o middleware de autorização, permitindo apenas que administradores acessem os dados.
    c. **Contratos (`/api/contracts`):** Apliquei controle de acesso e sanitização de parâmetros, além de proteger a consulta ao banco com parâmetros preparados.

Justificativa: Essas melhorias garantem que cada endpoint atenda aos critérios de segurança e boas práticas.

Resultados Alcançados

Com a refatoração, obtive os seguintes resultados:

1. **Segurança aprimorada:** A aplicação está protegida contra ataques como força bruta, SQL Injection e acessos não autorizados.
2. **Código organizado:** A nova estrutura modular facilita a manutenção e adição de novas funcionalidades.
3. **Melhor experiência do usuário:** Processos como login e autenticação tornaram-se mais eficientes e seguros.
4. **Aderência a boas práticas:** O uso de JWT, middleware e modularização alinha-se aos padrões recomendados para aplicações Node.js.

Conclusão

A refatoração realizada transformou a aplicação, tornando-a mais robusta, segura e escalável. Essa experiência foi enriquecedora, permitindo-me aplicar conceitos fundamentais de segurança e arquitetura de sistemas. Estou confiante de que, com essas alterações, a aplicação está pronta para atender aos requisitos da Missão Prática e oferecer um serviço confiável aos usuários.