Arquitetura de Segurança - API RESTful: SAW - Desafio Técnico

1. Introdução

Este documento apresenta uma análise de arquitetura de possíveis vulnerabilidades e brechas em uma API RESTful conforme os requisitos estabelecidos no projeto SAW - Desafio Técnico - Backend. O objetivo é fornecer uma visão crítica sobre segurança e propor mecanismos de mitigação.

2. Superfícies de Ataque da API

A seguir estão as principais superfícies de ataque da API:

- Endpoints expostos (`/api/products`, `/api/users`, etc);
- Autenticação e geração de tokens JWT;
- Controle de acesso baseado em roles;
- Validações de entrada;
- Armazenamento de senhas e dados sensíveis;
- Acesso ao banco de dados (MongoDB).

3. Possíveis Vulnerabilidades e Brechas

3.1. Autenticação e Autorização:

- Token JWT com algoritmo `none` ou sem assinatura válida;
- Tokens sem expiração (`exp`);
- Falta de verificação de role antes de ações críticas (ex.: delete produto).

3.2. Validações Insuficientes:

- Falta de validação no lado do servidor;
- Campos como `price`, `stock` e `email` sem validação robusta permitem ataques de fuzzing ou injection.

3.3. Enumerações e Exposição de Dados:

- Mensagens de erro detalhadas revelam estrutura interna (ex.: UUID inválido);
- `GET /api/products` sem paginação ou filtros pode levar a vazamento massivo de dados (Data Leakage).

3.4. Armazenamento de Senhas:

- Senhas devem ser armazenadas com algoritmos fortes (BCrypt, Argon2);
- Nunca devem ser retornadas nem mesmo com `isActive: false`.

3.5. Injeções e Ataques ao MongoDB:
- MongoDB Injection se entradas forem usadas diretamente nas queries;
- Validação dos dados de entrada deve ser rigorosa (ex.: evitar `{"$ne": null}`).

3.6. Excesso de Permissões:
- Usuários `user` conseguindo acessar rotas restritas a `admin`;
- Falta de segmentação entre escopos de leitura e escrita.

3.7. Rate Limiting e Brute Force:
- Falta de controle de tentativas em `/auth/login` permite ataques de força bruta.

4. Recomendações de Segurança

- Usar HTTPS sempre (TLS 1.2+);
- Adotar JWT com expiração curta e refresh token seguro;
- Validar roles e permissões em cada endpoint;
- Sanitizar e validar todas as entradas (server-side);
- Implementar CORS restritivo e cabeçalhos de segurança (Helmet);
- Adotar ferramentas de análise estática e dinâmica no CI/CD;
- Registrar auditoria de ações sensíveis;
- Utilizar políticas de segurança para dados sensíveis no MongoDB (ex.: criptografia at-rest);
- Implementar rate limiting (ex.: 5 tentativas/minuto).

5. Considerações Finais

A segurança de APIs REST exige uma abordagem holística, integrando validações, boas práticas de autenticação/autorização, e mecanismos de proteção contra abuso. Este documento serve como base para revisão contínua da segurança no projeto SAW.