Nome: Gabriel Brito Bitencourt

Nome: Mateus Brito Bitencourt

Nome: Cecília Weselowsky da Cunha

Nome: Gabriel Marques Costa

Proposta Técnica: DockShield - Sistema de Análise de Vulnerabilidades em Ambientes Containerizados

Disciplina: Ferramentas de Desenvolvimento Web (FDW)

Semestre: 2025/2

1. Resumo Executivo

O presente documento formaliza a proposta técnica para o desenvolvimento do **DockShield**, uma plataforma web integrada voltada para a gestão de segurança em infraestruturas baseadas em containers Docker. O sistema visa preencher a lacuna entre a identificação técnica de vulnerabilidades (CVEs) e a tomada de decisão gerencial, oferecendo uma interface centralizada e segura para a visualização de relatórios de auditoria de segurança.

2. Contextualização e Problematização

2.1. O Cenário Atual

A adoção massiva de arquiteturas de microsserviços e containerização (Docker/Kubernetes) trouxe agilidade ao ciclo de desenvolvimento de software (DevOps). No entanto, essa velocidade frequentemente introduz riscos de segurança:

- Imagens base desatualizadas contendo vulnerabilidades conhecidas.
- Dificuldade em rastrear quais containers em produção estão vulneráveis.
- Relatórios de segurança dispersos em logs brutos (JSON/Texto) de difícil interpretação por gestores.

2.2. O Problema

A ausência de uma ferramenta centralizada e acessível via web que permita a autenticação segura de usuários e a visualização amigável dos relatórios de varredura de vulnerabilidades torna o processo de mitigação de riscos lento e propenso a falhas humanas.

3. Objetivos do Projeto

3.1. Objetivo Geral

Desenvolver e implantar um Produto Mínimo Viável (MVP) de uma aplicação web *full-stack* capaz de autenticar usuários e apresentar, de forma estruturada e gráfica, as vulnerabilidades detectadas em imagens Docker.

3.2. Objetivos Específicos

1. **Segurança de Acesso:** Implementar um módulo de autenticação robusto que garanta que apenas pessoal autorizado tenha acesso aos dados sensíveis de vulnerabilidade.
2. **Visualização de Dados:** Converter relatórios técnicos complexos (JSON/Markdown gerados por ferramentas de varredura ou IA) em dashboards web intuitivos.
3. **Arquitetura Resiliente:** Construir o sistema sobre uma arquitetura de microsserviços containerizados, garantindo facilidade de *deploy* e isolamento de falhas.
4. **Integração:** Conectar interfaces web modernas (HTML5/CSS3) com lógicas de backend performáticas (Node.js e Python).

4. Escopo do MVP (Entregas da Avaliação N2)

O escopo deste MVP foi delimitado para atender aos requisitos da disciplina de FDW, focando nas funcionalidades críticas de segurança e visualização.

4.1. Módulo de Gestão de Identidade (Frontend/Auth)

Este módulo é o ponto de entrada do sistema.

- **Tecnologia:** Node.js + Express.
- **Funcionalidades:**
    - Cadastro de novos usuários.
    - Login seguro com validação de credenciais.
    - Edição de senha.
    - Exclusão de conta (Direito ao Esquecimento).
- **Mecanismos de Segurança:**
    - Armazenamento de senhas via Hash (Bcrypt).
    - Gestão de sessão via JSON Web Tokens (JWT).
    - Proteção contra XSS via Cookies HttpOnly.

4.2. Módulo de Core Business (Backend/App)

Este módulo processa e exibe as informações de segurança.

- **Tecnologia:** Python + Flask (Servido via Apache/WSGI).
- **Funcionalidades:**
    - Listagem de imagens Docker analisadas (Coleções).
    - Visualização detalhada do relatório de análise das CVEs (Renderização Markdown -> HTML).
    - Listagem paginada de CVEs (*Common Vulnerabilities and Exposures*).
- **Mecanismos de Segurança:**

- Middleware de verificação de Token JWT para proteção de rotas.

4.3. Infraestrutura de Dados

- **Tecnologia:** MongoDB (NoSQL).
- **Justificativa:** A escolha de um banco NoSQL deve-se à natureza hierárquica e variável dos relatórios de vulnerabilidade de containers, que se adaptam melhor a documentos JSON do que a tabelas rígidas relacionais.

5. Arquitetura da Solução

A solução adota o padrão de **Microsserviços**, onde cada componente executa em seu próprio container Docker, orquestrados via Docker Compose.

- **Frontend:** Porta 3000 (Acesso Público para Login).
- **Backend:** Porta 5000 (Acesso Restrito/Redirecionado).
- **Banco de Dados:** Porta 27017 (Rede Interna Docker).

A comunicação entre os serviços ocorre através de uma rede *bridge* privada do Docker, garantindo que o banco de dados não fique exposto diretamente à internet pública.

6. Resultados Esperados

Ao final do desenvolvimento, espera-se entregar uma plataforma funcional que permita:

1. O registro e autenticação segura de um administrador.
2. A navegação fluida entre o serviço de autenticação e o serviço de aplicação.
3. A visualização clara das vulnerabilidades de uma imagem Docker (ex: ubuntu:latest), permitindo a rápida identificação de riscos críticos (CVSS Score).