Logotipo, nome da empresa

O conteúdo gerado por IA pode estar incorreto.

Segurança de dados

SÃO PAULO – SP

2025

Nome: Marcio Guilherme Jacobino de Freitas Santos

RA: 3752556802

**PORTIFÓLIO**

# SUMÁRIO

[SUMÁRIO 2](#_Toc195781584)

[1.INTRODUÇÃO 3](#_Toc195781585)

[2.VULNERABILIDADES 4](#_Toc195781586)

[2.1. Vulnerable JS Library 4](#_Toc195781587)

[2.2. Ausência de tokens Anti-CSRF 4](#_Toc195781588)

[2.3. CORS Misconfiguration 5](#_Toc195781589)

[2.4. Content Security Policy (CSP) Header Not Set 5](#_Toc195781590)

[2.5. Vulnerable JS Library (jQuery 2.1.3) 5](#_Toc195781591)

[2.6. Falhas em Cookies (HttpOnly, Secure, SameSite) 6](#_Toc195781592)

[2.7. Inclusão de Scripts de Terceiros sem Controle 6](#_Toc195781593)

[3.ANEXOS 7](#_Toc195781594)

[4.CONCLUSÃO 8](#_Toc195781595)

# 1.INTRODUÇÃO

Nesta atividade, você irá identificar e mitigar vulnerabilidades em uma aplicação fictícia. Para isso, você utilizará as diretrizes e ferramentas fornecidas pelo OWASP (Open Web Application Security Project), uma comunidade global dedicada a melhorar a segurança de software. Utilizando o OWASP ZAP, realize uma varredura na aplicação em busca de vulnerabilidades conhecidas. Documente as vulnerabilidades encontradas, incluindo uma descrição, classificação do risco (alta, proposta de mitigação para cada uma delas. Relatório de Recomendações média, baixa) e possíveis impactos. Com base nas vulnerabilidades identificadas, elabore uma a) Compile todas as informações coletadas nos passos anteriores em um relatório. b) Apresente as vulnerabilidades identificadas, suas classificações de risco e as recomendações de mitigação em uma estrutura clara e organizada.

# 2.VULNERABILIDADES

## 2.1. Vulnerable JS Library

* **Risco:** Alto
* **Descrição:** Uso da biblioteca Chart.js versão 2.7.2, que apresenta vulnerabilidades conhecidas (CVE-2020-7746).
* **Impacto:** Possibilidade de exploração por ataques XSS e comprometimento de integridade de dados gráficos.
* **Recomendação:** Atualizar imediatamente para a versão mais recente da biblioteca.
* **Opinião:** A presença de bibliotecas desatualizadas é uma porta de entrada comum e perigosa, frequentemente negligenciada. Automatizar a verificação e atualização de dependências é essencial.

## 2.2. Ausência de tokens Anti-CSRF

* **Risco:** Médio
* **Descrição:** Diversos formulários da aplicação não utilizam tokens CSRF.
* **Impacto:** Permite que um invasor envie requisições maliciosas em nome do usuário, podendo alterar dados ou comprometer a conta.
* **Recomendação:** Implementar tokens CSRF em todos os formulários críticos e utilizar bibliotecas confiáveis como o OWASP CSRFGuard.
* **Opinião:** Considero está uma falha grave em termos de boas práticas. Embora classificada como médio risco, o potencial de exploração em cenários reais é considerável.

## 2.3. CORS Misconfiguration

* **Risco:** Médio
* **Descrição:** Uso do cabeçalho Access-Control-Allow-Origin: \*, permitindo acesso irrestrito a APIs.
* **Impacto:** Pode expor dados não autenticados a domínios maliciosos.
* **Recomendação:** Restringir o CORS apenas a domínios confiáveis.
* **Opinião:** Embora o impacto direto seja mitigado por restrições dos navegadores, este tipo de configuração deve ser evitado em produção.

## 2.4. Content Security Policy (CSP) Header Not Set

* **Risco:** Médio
* **Descrição:** Ausência do cabeçalho CSP em diversas páginas.
* **Impacto:** Aumenta a superfície de ataque para XSS e injeção de conteúdo.
* **Recomendação:** Implementar uma política CSP rigorosa com whitelists específicas para scripts e estilos.
* **Opinião:** É uma camada importante de defesa. Aplicações modernas devem considerar a CSP como parte do padrão de segurança.

## 2.5. Vulnerable JS Library (jQuery 2.1.3)

* **Risco:** Médio
* **Descrição:** A versão de jQuery utilizada possui múltiplas vulnerabilidades críticas (diversos CVEs).
* **Impacto:** Possibilidade de execução remota de scripts, manipulação de DOM e injeções de código.
* **Recomendação:** Atualizar para a versão 3.6 ou superior.
* **Opinião:** Tal como a falha anterior, reforça a importância da gestão de dependências como processo contínuo.

## 2.6. Falhas em Cookies (HttpOnly, Secure, SameSite)

* **Risco:** Baixo
* **Descrição:** Cookies de sessão foram identificados sem os atributos de segurança adequados.
* **Impacto:** Exposição a ataques como XSS, session hijacking e CSRF.
* **Recomendação:**
  + Usar HttpOnly para impedir acesso via JavaScript.
  + Habilitar Secure para transmissão apenas via HTTPS.
  + Definir SameSite=Strict para mitigar CSRF.
* **Opinião:** Pequenas configurações com grande impacto. São ajustes simples que aumentam significativamente a segurança de sessão.

## 2.7. Inclusão de Scripts de Terceiros sem Controle

* **Risco:** Baixo
* **Descrição:** Scripts de diversos domínios externos são carregados sem controle.
* **Impacto:** A aplicação fica dependente de código de terceiros que pode ser alterado ou comprometido.
* **Recomendação:** Implementar Subresource Integrity (SRI) e revisar fornecedores externos periodicamente.
* **Opinião:** Ainda que comum, esse padrão representa um vetor de ataque crescente com o aumento da cadeia de suprimento de software.

# 3.ANEXOS

Tabela

O conteúdo gerado por IA pode estar incorreto.

Relatório gerado pelo app Zap, baseado nas informações contidas nele cheguei as conclusões.

# 4.CONCLUSÃO