Avaliação de vulnerabilidades em aplicações utilizando OWASP Top 10

Leonardo Osvald de Souza[1], **Vitalino Pitt**[1]

[1]Ciência da Computação – Atitus Educação
Passo Fundo – RS – Brasil

aluno@atitus.edu.br, orientador@atitus.edu.br

Abstract. *teste*

Resumo. *Resumo do trabalho (teste)...*

1. Introdução

A segurança de aplicações web tornou-se um pilar fundamental no desenvolvimento de sistemas modernos, dada a crescente dependência do mundo digital para operações críticas e interações diárias. Com a grande crescente de serviços online, a superfície de ataque para cibercriminosos expandiu-se significativamente, tornando as aplicações web alvos constantes de exploração. A falha em proteger adequadamente essas aplicações pode resultar em perdas financeiras substanciais, comprometimento de dados sensíveis, danos à reputação e interrupção de serviços.

Nesse cenário, a compreensão das vulnerabilidades e a implementação de práticas de desenvolvimento seguro são imperativas. Segundo Almeida (2021) Uma vulnerabilidade pode ser definida como uma fraqueza ou falha em um sistema de informação que pode ser explorada por uma ameaça para comprometer a segurança do sistema. A identificação e mitigação dessas falhas são essenciais para garantir a integridade, confidencialidade e disponibilidade das informações processadas pelas aplicações web .

2. Referencial Teórico

Na seguinte seção serão apresentados os conceitos base para a evolução do trabalho.

2.1. OWASP Top 10

A Open Web Application Security Project (OWASP) é uma comunidade global sem fins lucrativos dedicada a melhorar a segurança de software. Uma de suas iniciativas mais reconhecidas é o OWASP Top 10, um documento que lista e descreve os dez riscos de segurança mais críticos para aplicações web. Este ranking é atualizado a cada 4 anos com base em dados de vulnerabilidades de milhares de aplicações, servindo como um guia essencial para desenvolvedores e profissionais de segurança .

O OWASP Top 10 não é uma lista de todas as vulnerabilidades existentes, mas sim um consenso sobre os riscos mais notaveis e impactantes que as empresas devem priorizar em seus esforços de segurança. Ele visa aumentar a conscientização sobre os riscos de segurança de aplicações e dar orientações para a uma redução de possíveis falhas em sistemas . O entendimento e aplicação dos conceitos do OWASP Top 10 são cruciais para o desenvolvimento de aplicações robustas e seguras.

2.2. Segurança de APIs e aplicações Web

Com a arquitetura de microsserviços e a crescente adoção de APIs (Application Programming Interfaces) para comunicação entre sistemas, a segurança dessas interfaces tornou-se uma preocupação crítica. As APIs são frequentemente a porta de entrada para dados e funcionalidades de backend, tornando-as alvos atraentes para atacantes . O OWASP API Security Top 10 é uma iniciativa específica que foca nos riscos de segurança mais críticos para APIs, complementando o OWASP Top 10 tradicional .

Segundo Leite (2022) "Uma aplicação web é um tipo de software projetado para operar mediante o acesso e a interação por meio de um navegador web"as quais, caso desenvolvidas sonegando boas práticas de segurança, tornam-se alvos em potencial para invasores, tendo em vista que toda grande corporação possui algum tipo de sistema que está conectado a rede mundial de computadores. As vulnerabilidades em APIs e sistemas web podem incluir quebras de autenticação e autorização, exposição excessiva de dados, injeção, e configurações de segurança inadequadas. A análise de maneiras seguras no desenvolvimento de APIs, de acordo com as diretrizes do OWASP API Security Top 10, é fundamental para proteger os sistemas modernos . Isso envolve a implementação de autenticação e autorização robustas, validação de entrada e saída, gerenciamento de erros seguro e monitoramento contínuo.

2.3. Vulnerabilidades mais Comuns em Aplicações Web

As vulnerabilidades listadas no OWASP Top 10 abrangem uma infinidade de falhas de segurança que podem ser exploradas por atacantes. Dentre as mais persistentes e perigosas,como de acordo com o OWASP TOP 10 2021 destacam-se:

2.2.1 Quebra de Controle de Acesso (Broken Access Control)

As falhas de controle de acesso ocorrem quando as restrições sobre o que usuários autenticados podem fazer não são aplicadas corretamente. Atacantes podem explorar essas falhas para acessar funcionalidades não autorizadas, visualizar ou modificar dados de outros usuários, ou alterar privilégios. A implementação de um controle de acesso robusto e a verificação de permissões em cada requisição são cruciais para prevenir essas vulnerabilidades .

2.2.2 Exposição de dados sensíveis (Cryptographic Failures)

As exposições de dados sensiveis (Cryptographic Failures) acontecem no momento em que, segundo Joseph (2024), dados são enviados/recebidos sem serem criptografados assim levando dados sensíveis a serem expostos. A utilização do protocolo HTTPS, por exemplo é inprescindível para que dados possam transitar na web de forma segura, permitindo que dados sejam criptografados e que apenas quem de fato deveria ter acesso aos dados, possa acessá-los.

2.2.3 Injeção (Injection)

As falhas de injeção, como SQL Injection, ocorrem quando dados não confiáveis são enviados a um interpretador como parte de um comando ou consulta. Os dados maliciosos do atacante podem enganar o interpretador para executar comandos não intencionais ou acessar dados sem autorização . Este tipo de ataque pode levar à divulgação completa de dados, modificação ou exclusão de informações, e até mesmo ao controle to-

tal do servidor . A prevenção envolve o uso de consultas parametrizadas, procedimentos armazenados e validação rigorosa de entrada .

2.2.2 Quebra de Autenticação e Gerenciamento de Sessão (Broken Authentication)

Essas vulnerabilidades permitem que atacantes comprometam senhas, chaves de sessão ou tokens de autenticação, ou explorem falhas na implementação de funções de autenticação ou gerenciamento de sessão para assumir a identidade de outros usuários. Isso pode incluir ataques de força bruta, credenciais fracas, ou falhas na expiração de sessão .

2.2.3 Cross-Site Scripting (XSS)

O XSS, que é um outro tipo de Injeção de código malicioso, ocorre quando uma aplicação inclui dados não confiáveis em uma página web sem a validação ou escape adequado. Isso permite que atacantes injetem scripts maliciosos no navegador da vítima, que podem roubar cookies de sessão, redirecionar o usuário para sites maliciosos ou realizar outras ações maliciosas em nome do usuário . A mitigação de XSS requer a sanitização de todas as entradas e a codificação de saída para evitar a execução de scripts .

3. Trabalhos Relacionados

Neste item serão apresentados os principais trabalhos que possuem uma relação com o tema definido neste estudo. Os 5 trabalhos selecionados abrangem a maioria dos assuntos chave para a evolução da pesquisa. Os estudos abaixo citam boas práticas, preocupações, e métodos adequados para previnir possíveis vulnerabilidades em aplicações Web, assim como explicações sobre as vulnerabilidades mais comuns.

- **Vulnerabilidade em Aplicações Web (Almeida, 2021)**
  Com o avanço da internet e o aumento das transações online, a segurança da informação tornou-se um tema essencial. As aplicações web, por estarem sempre conectadas, tornam-se alvos frequentes de invasões que exploram falhas como autenticação quebrada, injeções de código e erros de configuração.
  O estudo de Heitor Almeida teve como objetivo compreender o funcionamento dessas vulnerabilidades e propor formas práticas de mitigá-las. O autor buscou base teórica no OWASP Top 10, analisando as principais falhas que afetam servidores web e os riscos que representam tanto para usuários quanto para empresas.
  No desenvolvimento, o trabalho apresentou exemplos práticos e didáticos de códigos que exploram e corrigem vulnerabilidades, como SQL Injection, Broken Access Control e Falhas Criptográficas. Foram demonstradas técnicas de ataque e, em seguida, formas de defesa, como o uso de consultas parametrizadas e criptografia adequada para senhas.
  Como conclusão, o estudo reforçou a importância da adoção de boas práticas de desenvolvimento seguro e o uso contínuo das diretrizes do OWASP para prevenir ataques. O autor destaca que entender o funcionamento das falhas é essencial para desenvolver sistemas realmente protegidos e garantir a segurança dos dados de usuários e organizações.
- **Análise de Vulnerabilidades em Aplicações Web: Um Estudo de Caso Utilizando o OWASP ZAP (Sampaio, 2021)**

A segurança de aplicações web é um dos maiores desafios da atualidade, especialmente com o crescimento de ataques cibernéticos que exploram falhas em sistemas mal configurados. Diante desse cenário, o trabalho de Fernando Sampaio propõe analisar vulnerabilidades em aplicações web reais utilizando ferramentas de código aberto e metodologias consagradas.
O estudo teve como objetivo identificar e classificar vulnerabilidades presentes em um sistema web por meio da ferramenta OWASP ZAP, adotando como referência o OWASP Top 10. A pesquisa buscou demonstrar como falhas simples de configuração ou validação de entrada podem expor dados sensíveis e comprometer todo o sistema.
Durante o desenvolvimento, o autor realizou testes práticos em uma aplicação web, aplicando técnicas de varredura automatizada e inspeção manual. As vulnerabilidades encontradas foram analisadas quanto à gravidade e ao impacto, e foram sugeridas medidas corretivas como sanitização de entradas, uso de HTTPS, e gerenciamento adequado de sessões.
O trabalho conclui que o uso de ferramentas como o OWASP ZAP é fundamental para a detecção precoce de falhas, permitindo que desenvolvedores corrijam problemas antes que sejam explorados por atacantes. Sampaio reforça a necessidade de incorporar práticas de teste de segurança contínuo no ciclo de desenvolvimento de software.

- **Análise de Vulnerabilidades sobre a Aplicação Web do CINTE-RN: Um Estudo de Caso (Duarte, 2020)**
  Com o aumento constante de ataques cibernéticos, a segurança de aplicações web se tornou uma preocupação crítica, especialmente porque muitas falhas surgem quando a segurança não é uma prioridade durante o desenvolvimento do software. Este estudo se propôs a investigar, na prática, as vulnerabilidades presentes no site do Centro de Inovações Tecnológicas do Rio Grande do Norte (CINTE-RN). O objetivo era não apenas apontar falhas, mas também refletir sobre a importância de se adotar uma cultura de segurança desde a codificação até a hospedagem de uma aplicação.
  Para isso, os pesquisadores simularam a abordagem de um invasor externo, utilizando ferramentas especializadas como o NMAP e o Nessus para analisar o site e seu servidor sem ter nenhum conhecimento prévio da infraestrutura—uma metodologia conhecida como teste de "caixa preta".
  Os resultados revelaram 216 possíveis vulnerabilidades. A boa notícia é que a grande maioria das falhas mais sérias poderia ter sido evitada com medidas relativamente simples, como manter os sistemas atualizados, aplicar correções de segurança e configurar o servidor para não divulgar informações que facilitem um ataque. O trabalho conclui que a segurança não é uma etapa complexa e inacessível, mas sim uma soma de boas práticas e atenção contínua que podem prevenir a maioria dos problemas.
- **Understanding The Top 10 OWASP Vulnerabilities (Bach-Nutman, 2020)**
  A segurança de aplicações web é um tema urgente, já que falhas durante o desenvolvimento podem expor dados sensíveis, interromper serviços e causar sérios prejuízos financeiros e reputacionais. Com o aumento de ataques cibernéticos, é fundamental que empresas e desenvolvedores compreendam as vulnerabilidades mais comuns para se protegerem de forma eficaz.

Este trabalho buscou explicar e detalhar as 10 principais vulnerabilidades listadas pelo OWASP (Open Web Application Security Project), mostrando como elas podem ser exploradas e, principalmente, como podem ser mitigadas. O objetivo era oferecer uma visão clara e prática para ajudar desenvolvedores e organizações a protegerem suas aplicações.

O estudo foi baseado em uma revisão aprofundada de cada uma das vulnerabilidades do Top 10 do OWASP, como injeção de SQL, quebra de autenticação, exposição de dados sensíveis e configurações inseguras. Para cada item, foram apresentados exemplos de como os ataques ocorrem e medidas práticas de prevenção, como o uso de prepared statements, validação de entradas e a implementação de logging e monitoramento.

A pesquisa concluiu que a maioria das vulnerabilidades pode ser evitada com práticas de desenvolvimento seguro, atualizações regulares e configurações adequadas. Além disso, destacou que ataques bem-sucedidos não só causam perdas financeiras diretas, mas também abalam a confiança dos clientes. Investir em segurança desde o início do desenvolvimento é essencial para proteger tanto os negócios quanto os usuários finais.

- **Segurança Em Aplicações WEB: Uma abordagem prática para identificar vulnerabilidades (Santos, 2023)**

  A segurança em aplicações web é um tema de extrema importância na atualidade, mas ainda é negligenciada em muitos ambientes acadêmicos e profissionais, especialmente em empresas de pequeno e médio porte. A falta de conhecimento técnico e a priorização de funcionalidades em detrimento da segurança tornam os sistemas vulneráveis a ataques, o que pode resultar em prejuízos financeiros, danos à reputação e problemas legais.

  Este trabalho teve como objetivo principal demonstrar, de forma prática e acessível, o processo de análise de segurança em uma aplicação web. Para isso, foi desenvolvido e analisado o sistema RentX – uma plataforma de aluguel de carros – com o intuito de identificar vulnerabilidades e propor melhorias que garantam um nível mínimo de segurança, servindo como exemplo para desenvolvedores e empresas.

  Foi desenvolvida uma aplicação web completa, o RentX, utilizando tecnologias modernas como React, Node.js, Express e PostgreSQL. Em seguida, realizou-se um teste de penetração utilizando a ferramenta Zed Attack Proxy (ZAP), que permitiu uma análise dinâmica da aplicação em busca de vulnerabilidades com base no OWASP Top 10.

  A análise revelou a existência de vulnerabilidades de baixo e médio risco, como a ausência de cabeçalhos de segurança (CSP, X-Content-Type-Options), bibliotecas desatualizadas e configurações inadequadas. Concluiu-se que a maioria das falhas poderia ser corrigida com ajustes simples de configuração e atualizações, reforçando que a segurança deve ser integrada desde as fases iniciais do desenvolvimento. O trabalho mostrou que mesmo aplicações bem estruturadas podem conter brechas significativas, destacando a necessidade de testes de segurança contínuos.

A escolha dos 5 trabalhos acima se dá pela a correlação entre as pequisas e o tema abordado neste artigo, onde cada um dos tópicos cita pontos importantes para a presente

pesquisa, como segurança, pentest, autenticação, tipos de vulnerabilidades e como as previnir, etc...

4. Materiais e Métodos

Nesta seção serão estabelecidos os métodos adotados para a validação científica desta pesquisa, bem como as ferramentas utilizadas para identificação, exploração e mitigação das vulnerabilidades encontradas, tendo como principal referencial o *OWASP Top 10*.

4.1. Máquinas virtuais disponíveis na WEB

Para a validação e análise das vulnerabilidades estudadas, serão utilizadas máquinas virtuais e aplicações deliberadamente vulneráveis, obtidas em repositórios públicos (GitHub) para fins educacionais.

Visando simular um ambiente próximo ao encontrado em cenários reais, será empregada uma rede local composta por duas aplicações vulneráveis executadas em um único equipamento (servidor/notebook) e atacadas a partir de outro dispositivo na mesma rede. A abordagem do pentest adotada combina dois modos: *gray-box* para uma das aplicações e *black-box* para a outra, permitindo avaliar diferenças na detecção e exploração de falhas sob distintos níveis de informação.

4.2. Cenário Black Box: Metasploitable 3

Para simular um teste de penetração do tipo **Black Box**, onde o analista não possui conhecimento prévio da infraestrutura interna, foi utilizado o **(Metasploitable3, 2025)**. Este ambiente é uma máquina virtual intencionalmente vulnerável, baseada em Windows Server, que expõe uma variedade de serviços e aplicações, incluindo tecnologias do ecossistema Microsoft.

- **Justificativa da Escolha:** A escolha do Metasploitable 3 se deu por sua natureza como um "alvo desconhecido". Ele simula um servidor corporativo real, permitindo que a análise siga as fases de um pentest ético (reconhecimento, varredura, exploração) sem qualquer informação privilegiada.
- **Abordagem (Black Box):** O teste foi iniciado conhecendo apenas o endereço IP da máquina. A partir daí, todas as informações sobre serviços, tecnologias e vulnerabilidades foram obtidas através das ferramentas de análise, replicando a perspectiva de um atacante externo.

4.3. Cenário Grey Box: OWASP Juice Shop

Para a simulação de um teste do tipo **Grey Box**, foi escolhido o **OWASP Juice Shop (Shop, 2025)**. Trata-se de uma aplicação web moderna e complexa (escrita em Node.js e Angular), projetada pela OWASP para ser um campo de treinamento para segurança de aplicações.

- **Justificativa da Escolha:** O Juice Shop é ideal para uma análise Grey Box, pois seu código-fonte é aberto e acessível. Isso permite que o analista tenha acesso a informações parciais, como a lógica de negócio e a estrutura do código, para guiar os testes de segurança, um cenário comum quando se audita uma aplicação da própria empresa.

- **Abordagem (Grey Box):** A análise foi conduzida com acesso total ao repositório do código-fonte no GitHub. O código foi consultado para entender como certas funcionalidades foram implementadas, identificar potenciais falhas lógicas e construir ataques mais direcionados. Por exemplo, ao analisar o código de autenticação, foi possível formular hipóteses sobre como explorar a vulnerabilidade de "Quebra de Autenticação".

4.4. Ferramentas e Instrumentos de Análise

A seleção de ferramentas buscou cobrir as diferentes fases de um teste de invasão, sendo aplicadas em ambos os cenários conforme a necessidade, assim como ferramentas para o setup de infraestrutura requerida para a exposição das aplicações:

- **Nmap (Nmap, 2025):** Utilizado na fase de reconhecimento mapeamento de portas e serviços.
- **OWASP ZAP (Zap, 2025):** Empregado como proxy de interceptação para analisar o tráfego HTTP/S em ambos os ambientes, permitindo a manipulação de requisições para testar falhas de injeção, controle de acesso e outras.
- **Ferramentas Específicas:** Conforme a vulnerabilidade, ferramentas como **SQL Map (Map, 2025)** (para SQL Injection) e **Hydra (Hydra, 2025)** (para ataques de força bruta) foram utilizadas para automatizar e validar a exploração.
- **Análise de Código-Fonte:** O Visual Studio **(VSCode, 2025)** Code foi utilizado para navegar e inspecionar o código-fonte do Juice Shop, sendo uma ferramenta essencial para a abordagem Grey Box.
- **Docker (Docker, 2025):** Para a configuração das VMs e exposição das aplicações, serão utilizados containeres Docker para facilitar o processo de *deploy*, onde todoa a aplicação será *conteinerizada*.

Com base nos Materiais descritos previamente nesta seção, a analise será executada visando ter duas perspectivas de análise das vulnerabilidades, uma onde o pentester não tem informações prévias sobre o alvo (Black box) e outra onde as informações serão limitadas (Grey box). As escolhas de infraestrutura (um servidor em uma rede local) se dão para que seja possível simular cenários semelhantes aos encontrados em ambientes produtivos e as ferramentas selecionadas auxiliarão no processo.

5. Resultados e Discussão

Essa seção deverá ser escrita na segunda parte do trabalho, conhecida como TCC2, e deverá conter os resultados dos experimentos realizados, discussão comparando os resultados obtidos com outros encontrados em trabalhos similares, além de um parágrafo apontando as limitações da metodologia adotada.

Tabela 1. Minha tabela

cabeçalho 1	**cabeçalho 2**
texto à esquerda	Existem muitas variações das passagens do Lorem Ipsum disponíveis, mas a maior parte sofreu alterações de alguma forma.

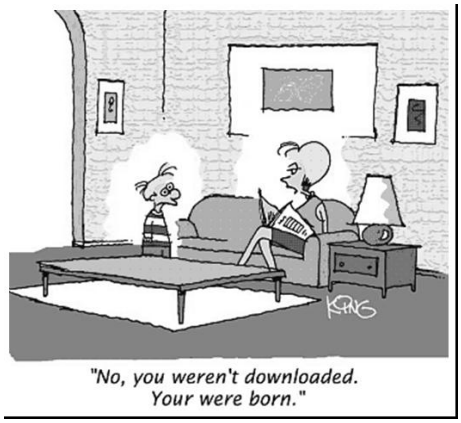
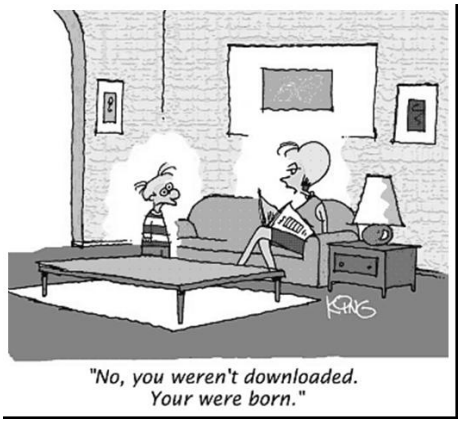
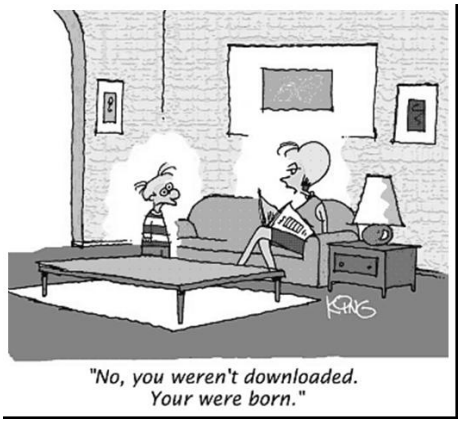


Figura 1. Exemplo de uso de figura

Figura 2. Exemplo com sub-figuras



(a) Sub-figura 1



(b) Sub-figura 2

Tabela 2. Tabela com sub-tabelas

cabeçalho 1	**cabeçalho 2**
Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old.	The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those interested.

(a) Sub-tabela da esquerda

cabeçalho 1	**cabeçalho 2**
Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old.	The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those interested.

(b) Sub-tabela da direita

6. Considerações Finais

Essa seção deverá ser escrita na segunda parte do trabalho, conhecida como TCC2.

Referências

ALMEIDA, H. S. de. **Vulnerabilidade em Aplicações Web**. 2021. Faculdade Pitágoras de Sobral. Disponível em: https://repositorio.pgsscogna.com.br/bitstream/123456789/40152/1/HEITOR%20SOUSA%20DE%20ALMEIDA.PDF.

BACH-NUTMAN, M. Understanding The Top 10 OWASP Vulnerabilities. **arXiv preprint arXiv:2012.09960**, 2020. Disponível em: https://arxiv.org/abs/2012.09960.

DOCKER, 2025. Disponível em: https://www.docker.com/.

DUARTE, G. C. **Análise de Vulnerabilidades sobre a Aplicação Web do CINTE-RN: Um Estudo de Caso**. 2020. Instituto Federal do Rio Grande do Norte. Disponível em: https://memoria.ifrn.edu.br/bitstream/handle/1044/2291/TCC%20Final-%20GILENO%20CORDEIRO%20DUARTE.pdf.

HYDRA, 2025. Disponível em: https://www.kali.org/tools/hydra/.

JOSEPH, J. **OWASP Top 10 Vulnerabilities in .NET Core 7+ (and How to Beat Them)**. 2024. Disponível em: https://james-joseph.medium.com/owasp-top-10-vulnerabilities-in-net-core-7-and-how-to-beat-them-2f27c38fb60b.

LEITE, M. V. C. **Segurança em Aplicações Web: Análise de Vulnerabilidades e Testes de Penetração**. 2022. Universidade Federal do Maranhão. Disponível em: https://rosario.ufma.br/jspui/bitstream/123456789/7552/1/Marcos_Vinicius_Correia_Leite.pdf.

MAP, S., 2025. Disponível em: https://sqlmap.org/.

METASPLOITABLE3. Análise de Vulnerabilidades sobre a Aplicação Web do CINTE-RN: Um Estudo de Caso, 2025. Disponível em: https://github.com/rapid7/metasploitable3.

NMAP, 2025. Disponível em: https://nmap.org/.

SAMPAIO, F. F. **Uma Análise Prática das Principais Vulnerabilidades em Aplicações Web Baseado no Top 10**. 2021. Universidade Federal do Ceará. Disponível em: https://repositorio.ufc.br/bitstream/riufc/62466/1/2021_tcc_ffsampaio.pdf.

SANTOS, M. G. dos. Segurança em Aplicações Web: Uma Abordagem Prática para Identificar Vulnerabilidades. João Pessoa, PB, 2023. Trabalho de Conclusão de Curso. Disponível em: https://repositorio.ufpb.br/jspui/handle/123456789/31664.

SHOP, O. J., 2025. Disponível em: https://owasp.org/www-project-juice-shop/.

VSCODE, 2025. Disponível em: https://code.visualstudio.com/.

ZAP, O., 2025. Disponível em: https://owasp.org/www-chapter-dorset/assets/presentations/2020-01/20200120-OWASPDorset-ZAP-DanielW.pdf.