



UNIVERSIDADE FEDERAL DE SANTA CATARINA

CENTRO TECNOLÓGICO

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

JOUSON BARRETO JOSÉ

**Metodologia Mista para Auxiliar o Desenvolvimento de Aplicações Web  
Seguras: combinando ferramentas tradicionais e de IA**

Florianópolis 2025

JOUSON BARRETO JOSÉ

***Metodologia Mista para Auxiliar o Desenvolvimento de Aplicações Web  
Seguras: combinando ferramentas tradicionais e de IA***

Dissertação submetida ao  
Programa de Pós-Graduação em Ciência  
da computação da Universidade Federal  
de Santa Catarina como requisito parcial  
para a obtenção do título de Mestre em  
ciências da computação Orientador(a):  
Prof.(a) .Dr.(a) Carla Merkle Westphall.

Florianópolis 2025

José , Jouson Barreto

Metodologia Mista para Auxiliar o Desenvolvimento de Aplicações Web Seguras: combinando ferramentas tradicionais e de IA; Orientadora, Carla Merkle Westphall, 2025.

Dissertação (mestrado) – Universidade Federal de Santa Catarina, centro tecnológico, programa de pós-graduação em ciência da computação , Florianópolis, 2025.

Jouson Barreto José

Metodologia Mista para Auxiliar o Desenvolvimento de Aplicações Web  
Seguras: combinando ferramentas tradicionais e de IA

O presente trabalho em nível de Mestrado foi avaliado e aprovado, em 06 de Agosto de 2025, pela banca examinadora composta pelos seguintes membros:

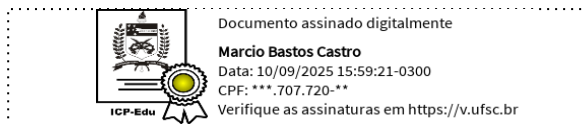
Prof.(a) Carla Merkle Westphall, Dr.(a)  
Universidade federal de Santa Catarina (UFSC)

Prof.(a) Cristina Meinhardt, Dr.(a)  
Universidade federal de Santa Catarina (UFSC)

Prof.(a) Romão Tiago Tiburski, Dr  
Instituto Federal de Santa Catarina(IFSC)

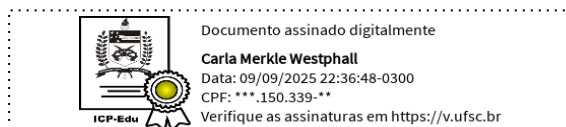
Prof.(a) Márcia Henke, Dr.(a)  
Universidade Federal de Santa Maria (UFSM)

Certificamos que esta é a versão original e final do trabalho de conclusão que foi julgado adequado para obtenção do título de Mestre em ciências da computação.



---

Prof.(a) Márcio Castro, Dr.  
Coordenação do Programa de Pós-Graduação



---

Prof.(a) Carla Merkle Westphall, Dr.(a)  
Orientador(a)

Florianópolis 2025

À todos os profissionais que se dedicam à segurança computacional,  
tornando a web um ambiente mais seguro para todos.

## AGRADECIMENTOS

A jornada deste mestrado foi desafiadora, mas nunca solitária. A cada passo, tive o privilégio de contar com pessoas extraordinárias, cujo apoio, incentivo e ensinamentos foram fundamentais para que este trabalho se tornasse realidade. Aos meus pais, **Albertina Odete Chipindo e Elias de Oliveira Tomé**, minha eterna gratidão. Vocês me ensinaram que o conhecimento é um tesouro e a persistência é a chave para conquistá-lo. O amor e os valores que me transmitiram foram a base que sustentou cada conquista desta caminhada. Aos meus irmãos, pelo apoio inabalável e pela motivação constante, lembrando-me sempre de que o sucesso é ainda mais significativo quando compartilhado em família.

À minha esposa, pelo carinho, paciência e incentivo nos momentos mais desafiadores. Sua presença foi meu porto seguro e sua confiança em mim tornou essa jornada mais leve. Ao **Dr. Hélder Lucas Chipindo**, agradeço imensamente pela oportunidade e pela confiança depositada em mim. Seu incentivo foi essencial para que eu seguisse firme nesse caminho.

À minha orientadora, **Prof.<sup>a</sup> Carla Merkle Westphall**, expresso minha sincera gratidão. Sua dedicação, conhecimento e orientação foram essenciais para que este trabalho atingisse um nível de excelência. Obrigado por cada conselho, cada sugestão e, acima de tudo, por acreditar no meu potencial. A todos que, de alguma forma, contribuíram para esta conquista, direta ou indiretamente, deixo o meu profundo agradecimento. Este trabalho é fruto não apenas do meu esforço, mas também do apoio de cada um de vocês.

*"A segurança de um sistema é tão forte  
quanto seu elo mais fraco."  
(SCHNEIER, 2000, p. 12)*

*Referência: SCHNEIER, Bruce. Secrets and Lies:  
Digital Security in a Networked World. Wiley, 2000.*

## RESUMO

A segurança na implantação de aplicações web tornou-se um desafio fundamental devido ao aumento das ameaças cibernéticas e à sofisticação dos ataques. Vulnerabilidades como Injeção de SQL, XSS e CSRF continuam sendo exploradas por agentes mal-intencionados, comprometendo a confidencialidade, integridade e disponibilidade dos sistemas. A complexidade crescente das aplicações exige soluções mais robustas, que vão além das abordagens tradicionais, incorporando técnicas automatizadas e inteligência artificial (IA) para identificar e mitigar falhas de segurança.

Este estudo propõe a metodologia **M-MADAWS** (Metodologia Mista para Auxiliar o Desenvolvimento de Aplicações Web Seguras), que integra ferramentas de varredura automática, como o Wapiti, e análise baseada em IA, incluindo o treinamento de um modelo com redes bayesianas. A metodologia foi desenvolvida com base na análise das vulnerabilidades mais comuns e na avaliação de desempenho de ferramentas automatizadas, considerando métricas como precisão, taxa de detecção e falsos positivos.

Para validar a eficácia da abordagem, a metodologia foi testada em ambiente controlado, com uso da aplicação DVWA, e comparada a métodos tradicionais. Os resultados demonstraram melhorias significativas na detecção e mitigação de falhas, comprovando a efetividade da abordagem.

Além dos benefícios na identificação de vulnerabilidades, destaca-se a importância do treinamento contínuo de administradores e desenvolvedores, bem como da adoção de boas práticas para reduzir riscos e evitar a exploração de falhas conhecidas. Conclui-se que a M-MADAWS representa uma abordagem promissora para fortalecer a segurança desde a fase de implantação de aplicações web.

**Palavras-chave:** Metodologia, segurança de aplicações web, inteligência artificial, Wapiti, varredura automática, redes bayesianas, métricas de avaliação, vulnerabilidades, mitigação, ataques cibernéticos.

## ABSTRACT

Security in web applications deployment has become a critical challenge due to the rise of cyber threats and increasingly sophisticated attacks. Vulnerabilities such as SQL Injection, XSS, and CSRF continue to be exploited by malicious actors, affecting system confidentiality, integrity, and availability. The growing complexity of web applications demands more robust solutions beyond traditional approaches, integrating automated techniques and artificial intelligence (AI) to detect and mitigate security flaws.

This study proposes the M-MADAWS methodology (Mixed Methodology to Assist the Development of Secure Web Applications), which combines automated scanning tools, such as Wapiti, with AI-based analysis, including the training of a Bayesian network model. The methodology was built upon an in-depth analysis of common vulnerabilities and performance evaluation of automated tools, using metrics such as precision, detection rate, and false positives.

To validate its effectiveness, M-MADAWS was tested in a controlled environment using DVWA and compared with traditional methods. Results confirmed significant improvements in vulnerability detection and mitigation, demonstrating the approach's practical value.

Beyond technical results, the study highlights the importance of ongoing training for developers and administrators, and the adoption of best practices to reduce persistent risks. M-MADAWS proves to be a promising methodology to strengthen web application security from the development phase.

**Keywords:** Methodology, web application security, artificial intelligence, Wapiti, automated scanning, Bayesian networks, evaluation metrics, vulnerabilities, mitigation, cyberattacks.

## LISTA DE FIGURAS

|                                                                                     |    |
|-------------------------------------------------------------------------------------|----|
| Figura 1. Resultado do experimento com a ferramenta Legion (Serviços).....          | 13 |
| Figura 2. Resultado do experimento com a ferramenta Mutillidae .....                | 14 |
| Figura 3. Sucesso no ataque de SQL Injection .....                                  | 15 |
| Figura 4. Ataque de XSS.....                                                        | 16 |
| Figura 5. Processo empregado para localizar e selecionar os artigos analisados .... | 27 |
| Figura 6. Arquitetura da metodologia M-MADAWS .....                                 | 54 |
| Figura 7. Resultado do scanner com wapiti .....                                     | 58 |
| Figura 8. Resultado do scanner com deepCode .....                                   | 58 |
| Figura 9. Resultados da precisão para ferramenta de ia.....                         | 62 |
| Figura 10. Resultados da matriz de confusão para XSS e CSFR .....                   | 64 |
| Figura 12. Código php da página login.php da dvwa .....                             | 68 |
| Figura 13. Padrões de segurança para mitigar ataques CSRF e XSS.....                | 69 |
| Figura 14. Resultado da varredura com a wapiti .....                                | 70 |
| Figura 15. Resultado da varredura com deepcode .....                                | 73 |
| Figura 16. Código php melhorado da pagina login.php da dvwa.....                    | 74 |
| Figura 17. Resultado da varredura com a wapiti .....                                | 76 |
| Figura 18 Integração com CI/CD (GitHub Actions).....                                | 77 |
| Figura 19. Script do Workflow GitHub Actions (main.yml) .....                       | 78 |

## LISTA DE TABELAS

|                                                                             |    |
|-----------------------------------------------------------------------------|----|
| Tabela 1. Scanners de Vulnerabilidades .....                                | 12 |
| Tabela 2. Critérios de classificação de padrões de design de segurança..... | 17 |
| Tabela 3. Bases de dados bibliográficas .....                               | 27 |
| Tabela 4. Artigos obtidos.....                                              | 29 |
| Tabela 5. Análise comparativa dos trabalhos.....                            | 46 |
| Tabela 6. Resultados para tipos de ataques CSRF .....                       | 61 |
| Tabela 7. Resultados para tipos de ataques XSS .....                        | 61 |
| Tabela 8. Plano de Testes e Análise de Vulnerabilidades .....               | 71 |
| Tabela 9. Consulta ao OWASP Top 10 .....                                    | 72 |
| Tabela 10. Comparação com linhas alteradas do Código HTML/PHP .....         | 75 |
| Tabela 11. Resultado da varredura CSRF .....                                | 79 |
| Tabela 12. Resultado da varredura XSS .....                                 | 79 |

## LISTA DE ABREVIATURAS E SIGLAS

**AI** – Artificial Intelligence (Inteligência Artificial)

**CD** – Continuous Delivery (Entrega Contínua)

**CI** – Continuous Integration (Integração Contínua)

**CSRF** – Cross-Site Request Forgery (Falsificação de Requisição Entre Sites)

**DVWA** – Damn Vulnerable Web Application (Aplicação Web Extremamente Vulnerável)

**HTTP** – Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto)

**IA** – Inteligência Artificial

**M-MADAWS** - Metodologia Mista para Auxiliar o Desenvolvimento de Aplicações Web Seguras

**ISO/IEC** – International Organization for Standardization / International Electrotechnical Commission

**OWASP** – Open Web Application Security Project

**SQL** – Structured Query Language (Linguagem de Consulta Estruturada)

**SSL/TLS** – Secure Sockets Layer / Transport Layer Security

**SFL** – Spatial Feature Learning (Aprendizado de Características Espaciais)

**T.I.** – Tecnologia da Informação

**TICs** – Tecnologias da Informação e Comunicação

**TP** – True Positive (Verdadeiro Positivo)

**FP** – False Positive (Falso Positivo)

**FN** – False Negative (Falso Negativo)

**VP** – Verdadeiro Positivo

**VDR** – Vulnerability Detection Rate (Taxa de Detecção de Vulnerabilidades)

**VRR** – Vulnerability Resolution Rate (Taxa de Resolução de Vulnerabilidades)

**XSS** – Cross-Site Scripting

## SUMÁRIO

|                                                                                                         |    |
|---------------------------------------------------------------------------------------------------------|----|
| 1 INTRODUÇÃO .....                                                                                      | 2  |
| 1.1 Motivação e problema científico .....                                                               | 3  |
| 1.2 Objetivos .....                                                                                     | 3  |
| 1.3 Justificativa .....                                                                                 | 4  |
| 1.4 Metodologia de Investigação .....                                                                   | 4  |
| 1.5 Resultados esperados .....                                                                          | 6  |
| 1.6 Organização do trabalho .....                                                                       | 6  |
| 2 FUNDAMENTAÇÃO TEÓRICA .....                                                                           | 8  |
| 2.1 Vulnerabilidades em aplicações Web .....                                                            | 8  |
| 2.1.1 Princípios básicos de segurança em sistemas web .....                                             | 8  |
| 2.1.2 Vulnerabilidades em ambientes Web .....                                                           | 9  |
| 2.1.3 Frameworks para testes de vulnerabilidades .....                                                  | 10 |
| 2.1.4 Redes Bayesianas como Ferramenta de Análise Preditiva de Vulnerabilidades em Aplicações Web ..... | 12 |
| 2.2 Testes práticos com algumas ferramentas .....                                                       | 13 |
| 2.3 Desenvolvimento seguro de aplicações Web .....                                                      | 17 |
| 2.3.1 Padrões de projeto .....                                                                          | 17 |
| 2.3.2 Normas existentes para segurança no desenvolvimento .....                                         | 19 |
| 2.3.3 Soluções para desenvolvimento seguro .....                                                        | 20 |
| 2.3.3.1 Ferramentas de segurança .....                                                                  | 21 |
| 2.3.3.2 Boas práticas de desenvolvimento Seguro .....                                                   | 22 |
| 2.3.4 Implantação segura de aplicações web .....                                                        | 22 |
| 3 TRABALHOS RELACIONADOS .....                                                                          | 25 |
| 3.1 Seleção e execução .....                                                                            | 27 |
| 3.2 Resultados da revisão .....                                                                         | 28 |
| 3.3 Descrição dos trabalhos relacionados .....                                                          | 30 |
| 3.3.1 Análise comparativa dos trabalhos relacionados .....                                              | 45 |
| 3.3.2 Respostas das questões de pesquisa .....                                                          | 47 |
| 3.4 Levantamento de problemas e desafios para a segurança no desenvolvimento de aplicações web .....    | 49 |
| 3.4.1 Sugestões de possíveis soluções para os problemas .....                                           | 50 |
| 4 METODOLOGIA MISTA PARA AUXILIAR O DESENVOLVIMENTO DE APLICAÇÕES WEB .....                             | 52 |
| 4.1 Metodologia Mista para Auxiliar o Desenvolvimento de Aplicações Web Seguras (M-MADAWS) .....        | 53 |
| 4.1.1 Métricas de avaliação .....                                                                       | 55 |

|                                                                                                          |    |
|----------------------------------------------------------------------------------------------------------|----|
| 4.2 Aplicações e ferramentas usadas nos experimentos .....                                               | 56 |
| 4.3 Resultados experimentais .....                                                                       | 59 |
| 4.3.1 Considerações sobre a ferramenta Wapiti .....                                                      | 60 |
| 4.4 Medição da ferramenta de scanner com redes bayesianas e ferramEntas de inteligência artificial ..... | 62 |
| 4.5 Considerações sobre os resultados obtidos .....                                                      | 64 |
| 5 ESTUDO DE CASO E RESULTADOS.....                                                                       | 66 |
| 5.1 Escopo do Estudo de Caso .....                                                                       | 66 |
| 5.1.1 Aplicando a Metodologia Mista M-MADAWS na DVWA.....                                                | 66 |
| 5.1.2 Etapa 1 - Definição do Modelo de Ameaças .....                                                     | 67 |
| 5.1.3 Etapa 2 - Definição de requisitos de segurança do ambiente e implantação no servidor .....         | 69 |
| 5.1.4 Etapa 3 - Varredura inicial com a ferramenta Wapiti .....                                          | 70 |
| 5.1.5 Etapa 4 - Plano de Testes e Análise de Vulnerabilidades .....                                      | 71 |
| 5.1.6 Etapa 5 - Consulta ao OWASP Top 10 .....                                                           | 72 |
| 5.1.7 Etapa 6 - Análise de Código-Fonte para Detecção de Vulnerabilidades com DeepCode .....             | 73 |
| 5.1.8 Etapa 7 - Desenvolvimento ou correção do código da aplicação .....                                 | 74 |
| 5.1.9 Etapa 8 - Implantação CI/CD e Configuração do Servidor .....                                       | 77 |
| 5.1.10 Etapa 9 - Monitoramento Contínuo .....                                                            | 78 |
| 5.2 Resultados dos experimentos .....                                                                    | 79 |
| 6 CONCLUSÕES .....                                                                                       | 80 |
| 6.1 Considerações sobre os objetivos da dissertação .....                                                | 80 |
| 6.2 Trabalhos Futuros .....                                                                              | 81 |
| REFERÊNCIAS .....                                                                                        | 83 |

## 1 INTRODUÇÃO

Com o constante avanço da tecnologia e a crescente dependência de aplicações web em setores críticos como saúde, governo, comércio eletrônico e serviços financeiros, a segurança desses sistemas tornou-se uma exigência estratégica. Vulnerabilidades como Injeção de SQL, Cross-Site Scripting (XSS) e Cross-Site Request Forgery (CSRF) continuam a ser amplamente exploradas por agentes mal-intencionados, comprometendo a confidencialidade, integridade e disponibilidade das informações (OWASP, 2021). As consequências de uma violação de segurança podem incluir perdas financeiras, danos à reputação institucional e vazamentos de dados sensíveis (Santos, 2018).

Nesse contexto, a identificação precoce e a mitigação eficaz de falhas tornaram-se pilares fundamentais para garantir a robustez das aplicações. A análise de segurança deve abranger todas as camadas do sistema, incluindo aplicação, servidor e banco de dados, e incorporar práticas modernas como a codificação segura, o uso de ferramentas de varredura e a aplicação de inteligência artificial. As abordagens tradicionais, embora úteis, têm se mostrado insuficientes diante da complexidade dos ataques atuais e da velocidade com que novas ameaças emergem.

Diante desses desafios, este trabalho propõe a metodologia M-MADAWS (Metodologia Mista para Auxiliar o Desenvolvimento de Aplicações Web Seguras), que combina ferramentas de varredura automática e análise baseada em IA com o objetivo de fortalecer o processo de desenvolvimento e implantação segura de aplicações web. A metodologia integra a ferramenta Wapiti, utilizada para varredura de vulnerabilidades, com o DeepCode (ou ferramentas similares de análise de código com suporte a IA) para inspeção inteligente do código-fonte, além do treinamento de um modelo baseado em redes bayesianas para apoiar a tomada de decisão na priorização de falhas.

A aplicação da M-MADAWS em um ambiente de testes controlado demonstrou ganhos significativos, alcançando taxas elevadas de detecção de vulnerabilidades, com destaque para a identificação de falhas XSS e CSRF, bem como melhorias na taxa de resolução dos problemas identificados, quando comparada a métodos convencionais. Os resultados comprovam que a combinação entre ferramentas automatizadas e inteligência artificial pode aumentar substancialmente a eficácia das

práticas de segurança adotadas, desde as fases iniciais do desenvolvimento até a implantação final.

Espera-se que esta pesquisa contribua significativamente para o fortalecimento da segurança no desenvolvimento de aplicações web, tanto no meio acadêmico quanto no ambiente profissional. Entre os resultados esperados, destacam-se: o desenvolvimento de uma metodologia que integre diferentes técnicas de proteção; a melhoria na precisão e taxa de detecção de vulnerabilidades, com redução de falsos positivos e negativos; a validação da metodologia por meio de estudos de caso e testes experimentais; a formulação de diretrizes práticas para sua adoção em contextos reais; e a contribuição para a literatura especializada, oferecendo um modelo estruturado e replicável por futuras pesquisas na área de segurança da informação.

## 1.1 MOTIVAÇÃO E PROBLEMA CIENTÍFICO

Atualmente, as aplicações web estão cada vez mais presentes em nossas vidas, desde serviços bancários até compras online. No entanto, essa crescente dependência de sistemas web também abre espaço para uma maior exposição a ameaças cibernéticas, como hackers e criminosos virtuais, que buscam explorar vulnerabilidades em tais sistemas. O problema científico que esta pesquisa visa abordar é identificação das principais vulnerabilidades de segurança identificadas em sistemas web e quais técnicas podem ser adotadas para mitigar essas vulnerabilidades? (S. Yadav e M. Brown, 2023, 2022) .

## 1.2 OBJETIVOS

O objetivo geral deste trabalho é desenvolver a metodologia M-MADAWS para auxiliar o desenvolvimento e implantação segura de aplicações web, integrando técnicas de varredura automática e análise baseada em inteligência artificial para aprimorar a identificação e mitigação de vulnerabilidades.

Para alcançar este objetivo, são propostos os seguintes objetivos específicos:

- Analisar vulnerabilidades comuns em aplicações web e seu impacto na segurança dos sistemas;

- Avaliar o desempenho de ferramentas de varredura automática e de análise de código baseada em IA, considerando métricas como Precisão e Taxa de Detecção;
- Implementar e testar a metodologia proposta em um ambiente controlado, comparando sua eficácia com métodos tradicionais; e,
- Validar a metodologia proposta, por meio de estudos de caso, experimentação prática ou simulações em ambientes controlados.

### 1.3 JUSTIFICATIVA

A segurança de aplicações web é essencial no cenário atual, onde essas plataformas são amplamente utilizadas em setores como bancos, comércio eletrônico e governo. Com a crescente digitalização, falhas de segurança podem expor informações sensíveis e causar prejuízos significativos, tornando a identificação e mitigação de vulnerabilidades um aspecto crítico da segurança cibernética (S. Yadav, 2023). Diante do aumento dos ataques cibernéticos, é fundamental adotar abordagens para fortalecer a proteção desses sistemas. Esta dissertação justifica-se pela necessidade de desenvolver a M-MADAWS, uma metodologia mista que combina técnicas tradicionais de varredura com inteligência artificial, proporcionando maior precisão na detecção e mitigação de vulnerabilidades.

O estudo busca contribuir para a segurança digital, alinhando-se às crescentes demandas por mecanismos mais eficazes na proteção de aplicações web, garantindo maior segurança para usuários e organizações.

### 1.4 METODOLOGIA DE INVESTIGAÇÃO

Para atingir os objetivos desta investigação, adota-se uma abordagem qualitativa, fundamentada em revisão bibliográfica, análise de documentos técnicos e estudo de caso. Essa abordagem possibilita uma compreensão aprofundada das vulnerabilidades em aplicações web e das metodologias existentes para mitigá-las, servindo de base para o desenvolvimento da proposta metodológica M-MADAWS.

A metodologia, neste contexto, refere-se ao estudo dos métodos científicos aplicáveis à pesquisa (Vianello, 2015). Esta pesquisa adota métodos teóricos e empíricos. Os métodos teóricos fornecem a base conceitual necessária à formulação da metodologia proposta. Os métodos empíricos são empregados para validar a

metodologia por meio de experimentação prática, aplicando a M-MADAWS em diferentes cenários para avaliar sua eficácia na identificação e mitigação de vulnerabilidades em aplicações web.

A estrutura da investigação foi guiada pelo método hipotético-dedutivo, conforme proposto por Popper (1975) e descrito por Lakatos e Marconi (2008). Esse método organiza-se em quatro etapas: conhecimento, problema, conjectura e falsificação.

Na etapa de conhecimento buscou-se consolidar o conhecimento existente sobre segurança em aplicações web, metodologias de implantação e principais vulnerabilidades. As atividades incluíram:

- Revisão da literatura sobre segurança em aplicações web e metodologias de desenvolvimento seguro;
- Identificação de padrões de segurança estabelecidos por entidades como OWASP, NIST e ISO/IEC 27001;
- Análise de ferramentas para detecção de vulnerabilidades, como Wapiti e DeepCode; e,
- Estudo de práticas ágeis e DevOps voltadas à segurança no ciclo de vida do software.

Na etapa do problema, o objetivo foi identificar os principais desafios relacionados à segurança em aplicações web, definindo o problema central da pesquisa. As atividades abrangeram:

- Análise de casos de falhas de segurança;
- Avaliação de vulnerabilidades descritas no OWASP Top 10, com foco em XSS e CSRF;
- Aplicação do Wapiti em ambientes simulados para detecção de falhas;
- Utilização do DeepCode para análise de vulnerabilidades no código-fonte; e,
- Definição de métricas e critérios para avaliação da metodologia proposta.

Na etapa de conjectura, foi elaborada a proposta metodológica M-MADAWS. As atividades incluíram:

- Definição das etapas e procedimentos da metodologia;

- Integração de práticas de CI/CD (Continuous Integration/Continuous Delivery);
- Monitoramento contínuo de vulnerabilidades; e,
- Planejamento de experimentos em ambientes simulados ou reais.

Na etapa final, a metodologia foi testada para verificação de sua eficácia. As ações desenvolvidas foram:

- Realização de testes de segurança (manuais e automatizados);
- Aplicação da metodologia em diferentes cenários; e,
- Ajustes e aperfeiçoamentos com base nos resultados obtidos.

## 1.5 RESULTADOS ESPERADOS

Espera-se que esta dissertação contribua para a segurança no desenvolvimento e implantação de aplicações web, por meio da proposta e validação da M-MADAWS – uma metodologia mista que combina varredura automatizada e inteligência artificial para identificação e mitigação de vulnerabilidades. Entre os principais resultados esperados, destacam-se:

- Desenvolvimento de uma metodologia para reforçar a segurança das aplicações web, integrando diferentes abordagens e técnicas de proteção;
- Melhoria na precisão e taxa de detecção de vulnerabilidades, reduzindo falsos positivos e falsos negativos em comparação com métodos tradicionais;
- Geração de diretrizes e boas práticas para que empresas e desenvolvedores adotem a metodologia proposta, aprimorando a segurança desde a fase de desenvolvimento até a implantação dos sistemas; e,
- Contribuição para a literatura acadêmica e profissional, fornecendo um modelo estruturado que pode ser utilizado e aprimorado por futuras pesquisas na área de segurança da informação.

## 1.6 ORGANIZAÇÃO DO TRABALHO

O Capítulo 1 corresponde à introdução, na qual são apresentados o contexto da pesquisa, a problemática, os objetivos e a justificativa do estudo. Além disso, oferece uma visão geral sobre os temas abordados ao longo do trabalho.

No Capítulo 2, é apresentada a Fundamentação Teórica, abordando as principais vulnerabilidades em aplicações web, analisando as falhas mais comuns e

os impactos que podem comprometer a segurança dos sistemas. Também são discutidos aspectos relacionados ao desenvolvimento seguro de aplicações web, destacando princípios, diretrizes e boas práticas para a construção de sistemas resilientes a ataques cibernéticos. Além disso, são analisadas abordagens e ferramentas que auxiliam na implementação da segurança desde as primeiras fases do ciclo de vida do software.

No Capítulo 3, são apresentados os trabalhos relacionados, contemplando pesquisas e estudos que tratam da segurança no desenvolvimento de aplicações web. São discutidas metodologias, frameworks e soluções já propostas por outros autores, fornecendo embasamento teórico para o estudo.

No Capítulo 4, é descrita a proposta de uma metodologia para a implantação segura de aplicações web, detalhando técnicas e contramedidas para mitigar vulnerabilidades e aprimorar a segurança dos sistemas.

No Capítulo 5, é apresentado o estudo de caso, no qual são discutidos os resultados obtidos com a aplicação da metodologia proposta, analisando sua eficácia na mitigação de vulnerabilidades.

Por fim, no Capítulo 6, o trabalho é concluído com as considerações finais, nas quais são sintetizados os principais achados da pesquisa, suas limitações e sugestões para estudos futuros.

## 2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são descritos conceitos e as principais vulnerabilidades em aplicações web, analisando as falhas mais comuns e os impactos que podem comprometer a segurança dos sistemas.

### 2.1 VULNERABILIDADES EM APLICAÇÕES WEB

Uma aplicação web é um software que é executado em um navegador (*browser*) para conectar os clientes com os serviços oferecidos pelas empresas, sem a necessidade de instalação ou configuração local de softwares.

A segurança de sistemas web é uma preocupação crescente devido ao aumento das ameaças cibernéticas e à importância cada vez maior das aplicações online. Essas vulnerabilidades podem ser exploradas por hackers e resultar em roubo de informações, invasões de privacidade, interrupção dos serviços e outros tipos de ataques prejudiciais (Hoffman, 2020). Para proteger as aplicações web, é necessário um conhecimento abrangente sobre as principais vulnerabilidades existentes, bem como métodos e técnicas para identificá-las de forma eficiente. Alguns dos conceitos e fundamentos relacionados que podem ser abordados são discutidos na sequência do texto.

#### 2.1.1 PRINCÍPIOS BÁSICOS DE SEGURANÇA EM SISTEMAS WEB

Os princípios básicos de segurança em sistemas web são os pilares fundamentais no desenvolvimento e operação de aplicativos seguros. Autenticação é o processo de verificação de uma afirmação de que uma entidade do sistema ou recurso do sistema tem um determinado valor de atributo. Quanto à sua aplicação, esses princípios são aplicados em diferentes camadas de um sistema web. A autenticação pode ser implementada por meio de processos de login com nome de usuário e senha. A autorização é geralmente realizada por meio de níveis de acesso ou papéis de usuário definidos na aplicação. O controle de acesso envolve a configuração de permissões e restrições de acesso a recursos específicos. Em termos de sua importância, os princípios acima mencionados são cruciais para garantir a segurança de um sistema web. A autenticação adequada evita o acesso não autorizado a contas de usuários. A autorização permite que os administradores controlem o que pode ser feito por cada usuário. O controle de acesso garante que

apenas as pessoas autorizadas tenham acesso a recursos e dados confidenciais. Porém, em relação às suas limitações, embora esses princípios sejam fundamentais, existem algumas limitações a serem consideradas. A autenticação baseada em nome de usuário e senha pode ser vulnerável a ataques de força bruta ou ataques de *phishing*. A autorização pode ser difícil de gerenciar em sistemas complexos com muitos usuários e níveis de acesso. O controle de acesso pode enfrentar desafios ao lidar com a evolução contínua de requisitos de segurança e políticas corporativas (Liu, 2016).

### 2.1.2 VULNERABILIDADES EM AMBIENTES WEB

Alguns tipos de vulnerabilidades são mais comuns em sistemas web, de acordo com os estudos (Hoffman, 2020), (Johnson and Lee, 2020), (OWASP, 2025):

- Injeção de código: manipulação de entrada do usuário que permite a execução de comandos maliciosos no servidor ou no cliente.
- Cross-site scripting (XSS): inclusão de código malicioso em páginas web para atacar usuários que acessam essas páginas.
- Cross-site request forgery (CSRF): é um ataque que força um usuário final a executar ações indesejadas em um aplicativo da web no qual está atualmente autenticado.
- Insegurança de sessão: falhas na geração, transmissão e armazenamento de identificadores de sessão, permitindo o acesso não autorizado a contas de usuários.
- Configuração incorreta: erros de configuração que podem expor informações sensíveis, como arquivos de configuração, bancos de dados e outros recursos.
- Vulnerabilidade de fixação de sessão: A fixação de sessão é um ataque que permite a um invasor sequestrar uma sessão de usuário válida. O ataque explora uma limitação na forma como a aplicação web gerencia o ID da sessão, mais especificamente a aplicação web vulnerável.
- Servidor Web permite preenchimento automático de senha: O servidor web remoto contém pelo menos um campo de formulário HTML que possui uma entrada do tipo 'senha', onde 'preenchimento automático' não está definido como 'desativado'. Embora isso não represente um risco para este servidor web em si, significa que os usuários que usam os formulários afetados podem ter suas credenciais salvas

em seus navegadores, o que, por sua vez, pode levar à perda de confidencialidade se algum deles usar um serviço compartilhado, host ou se sua máquina estiver comprometida em algum momento.

- Explorando Dependências de Terceiros: Uma das formas mais populares de explorar dependências de terceiros em um curto espaço de tempo é determinar vulnerabilidades conhecidas que ainda não foram corrigidas nas dependências do aplicativo e atacar essas dependências (OWASP, 2020) (Pinto, 2020).

Os aplicativos web atuais possuem um grande número de vulnerabilidades (Hoffman, 2020). Algumas dessas vulnerabilidades são facilmente classificadas, outras vulnerabilidades são mais exclusivas para um único aplicativo se esse aplicativo tiver um modelo de segurança incomum ou possuir recursos com arquitetura exclusiva não encontrados em outros lugares. Dentre as vulnerabilidades populares estão XSS e CSRF, mas somente obtendo uma compreensão profunda do modelo de negócios subjacente e da lógica de negócios em um aplicativo é possível identificar vulnerabilidades mais avançadas e específicas (Gandikota et al., 2023), (Saha and Shamsuddoha, 2021).

### 2.1.3 FRAMEWORKS PARA TESTES DE VULNERABILIDADES

Existem diferentes frameworks de teste de vulnerabilidade na web (Gandikota et al., 2023). Alguns são descritos na sequência do texto.

Frameworks de pentesting são conjuntos estruturados de ferramentas, técnicas e metodologias utilizadas por profissionais de segurança da informação para avaliar e explorar vulnerabilidades em aplicações, sistemas e redes. Eles são fundamentais no processo de testes de intrusão (pentests), permitindo simular ataques reais para descobrir falhas antes que sejam exploradas por agentes maliciosos.

Os objetivos principais dos frameworks de pentesting são:

- Simular ataques reais para avaliar a robustez de uma aplicação;
- Identificar vulnerabilidades como falhas de autenticação, injeções, e erros de configuração;
- Apoiar a elaboração de relatórios técnicos com evidências de exploração; e,

- Ajudar na validação da eficácia de contramedidas aplicadas.

O framework Metasploit é um exemplo popular de ferramenta de pentesting que oferece centenas de exploits prontos para uso.

A distribuição Linux chamada Kali, é voltada para testes de segurança que integra diversos frameworks e ferramentas, incluindo Metasploit, Nmap, Burp Suite e scanners de vulnerabilidades.

Os scanners de vulnerabilidades são ferramentas automáticas que analisam sistemas, aplicações e redes com o objetivo de detectar falhas conhecidas. Diferente dos frameworks de pentesting, os scanners priorizam a automação e escala, sendo ideais para processos recorrentes de avaliação de segurança (OWASP, 2020). Os scanners de vulnerabilidades têm como objetivos principais:

- Identificar vulnerabilidades conhecidas (XSS, CSRF, SQL Injection, e outros);
- Fornecer relatórios detalhados com base em assinaturas e padrões;
- Reduzir o tempo e esforço manual necessário para análises rotineiras; e,
- Servir como apoio para correções rápidas antes de implantar sistemas em produção.

As ferramentas de testes de segurança podem ser classificadas em dois tipos principais: DAST (Dynamic Application Security Testing), que testam aplicações em tempo de execução; e, SAST (Static Application Security Testing), que analisam o código-fonte sem executá-lo. A tabela 1 lista alguns tipos de ferramentas de testes de segurança.

| Tipo | Ferramenta                 | Descrição                                                                                                                      |
|------|----------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| DAST | Wapiti                     | Scanner de código aberto que simula ataques black-box em aplicações web, identificando XSS, diretórios expostos, entre outros. |
| DAST | OWASP ZAP                  | Um dos mais usados para testes automatizados de vulnerabilidades web.                                                          |
| DAST | Burp Suite (Community/Pro) | Ferramenta poderosa para inspeção e manipulação de tráfego HTTP e detecção de falhas.                                          |
| SAST | DeepCode                   | Usa aprendizado de máquina para detectar falhas diretamente no código-fonte e sugerir correções precisas.                      |

Tabela 1. Scanners de Vulnerabilidades

Por meio de avaliação de vulnerabilidades e testes de penetração, são identificadas lacunas nas aplicações web e o site é escaneado em busca de vulnerabilidades (Hoffman, 2020).

#### 2.1.4 REDES BAYESIANAS COMO FERRAMENTA DE ANÁLISE PREDITIVA DE VULNERABILIDADES EM APLICAÇÕES WEB

As redes Bayesianas são modelos gráficos probabilísticos que representam variáveis e suas relações de dependência por meio de um grafo direcionado acíclico. Elas permitem a modelagem de incertezas e a realização de inferências com base em evidências, utilizando o teorema de Bayes como fundamento matemático. Esse tipo de rede é amplamente aplicado em áreas que exigem análise de dados incertos ou incompletos, devido à sua capacidade de combinar conhecimento prévio com informações observadas para gerar previsões e diagnósticos (PEARL, 1988).

No contexto da segurança de aplicações web, as redes Bayesianas têm se mostrado eficientes para prever a probabilidade de ocorrência de determinadas vulnerabilidades, como o Cross-Site Scripting (XSS) e o Cross-Site Request Forgery (CSRF). Por meio da modelagem de características extraídas de aplicações vulneráveis, é possível identificar padrões que indicam a maior propensão a ataques

específicos. Esses modelos permitem que pesquisadores e equipes de segurança classifiquem os sistemas de acordo com o risco, priorizando medidas de mitigação. Ao treinar uma rede Bayesiana com dados históricos de ataques e atributos técnicos, pode-se, por exemplo, identificar que determinadas configurações de entrada de dados ou ausência de validações aumentam significativamente a probabilidade de ocorrência de XSS, enquanto falhas na verificação de tokens de sessão favorecem o CSRF (JENSEN; NIELSEN, 2007).

## 2.2 TESTES PRÁTICOS COM ALGUMAS FERRAMENTAS

Nessa seção são descritos alguns experimentos práticos que foram feitos para conhecer algumas ferramentas disponíveis na máquina Kali Linux. Esta máquina é usada para aprendizado de atividades de ataque e defesa cibernética. Além da Kali Linux, usada como máquina atacante, foi usada também a máquina Owasp Broken, usada como alvo e vítima dos ataques. A máquina Owasp Broken é propositalmente vulnerável e possibilita testes das vulnerabilidades da lista Owasp Top Ten (OWASP Broken, 2016) (OWASP, 2025).

A figura 1 apresenta uma varredura (scan) de portas e serviços usando-se as ferramentas legion e nmap. O seguinte comando foi usado no teste: `sudo nmap -sV 10.1.2.5` (IP da máquina vulnerável Owasp Broken).

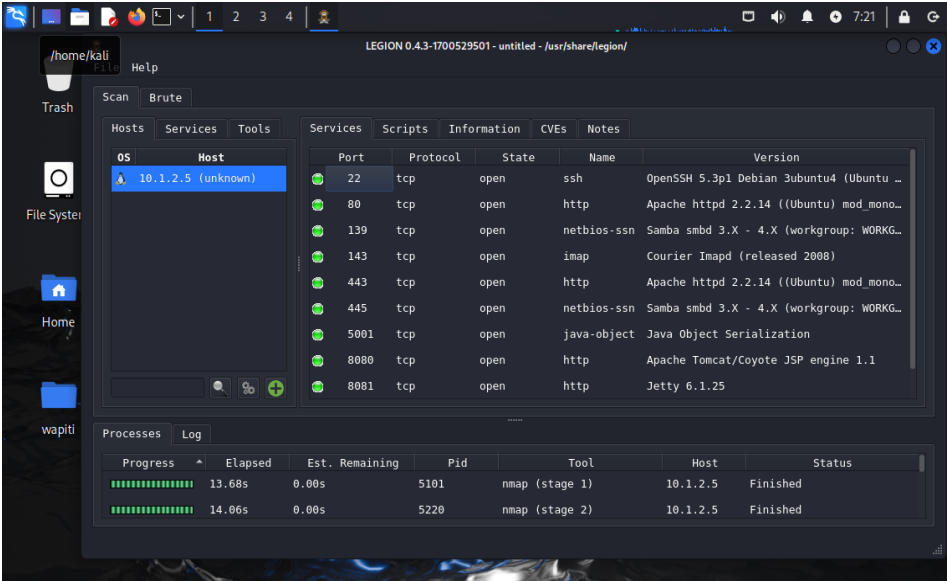


FIGURA 1. RESULTADO DO EXPERIMENTO COM A FERRAMENTA LEGION (SERVIÇOS)

A figura 1 demonstra que foram encontradas as portas dos serviços de rede da máquina com as respectivas versões dos serviços. Por exemplo, na porta 80 está sendo executado o servidor web Apache na versão 2.2.14, que é considerada antiga e vulnerável a vários tipos de ataques.

Na máquina Owasp Broken existe uma aplicação web vulnerável chamada Mutillidae. É possível testar o ataque de injeção de SQL nessa aplicação, preenchendo o campo Username (figura 2) com a string `' or 1=1 --`. Essa string faz com que a consulta SQL feita no banco de dados retorne a lista de usuários e respectivas senhas, apresentadas na figura 3.

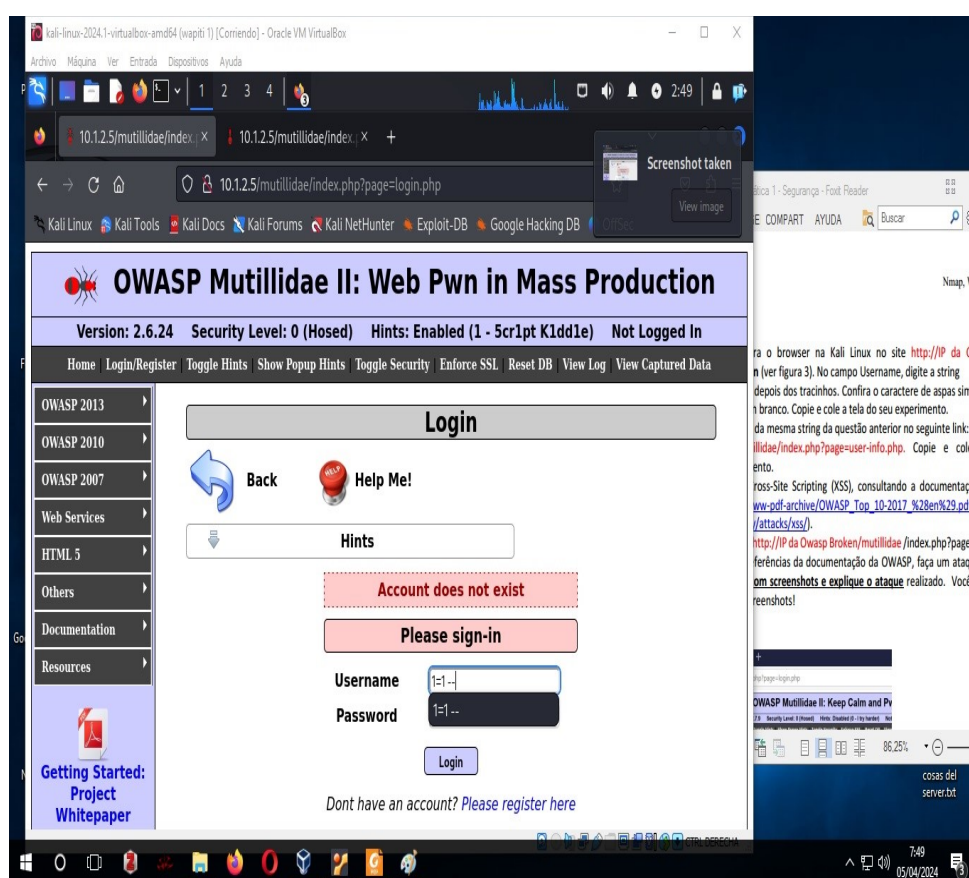


FIGURA 2. RESULTADO DO EXPERIMENTO COM A FERRAMENTA MUTILLIDAE

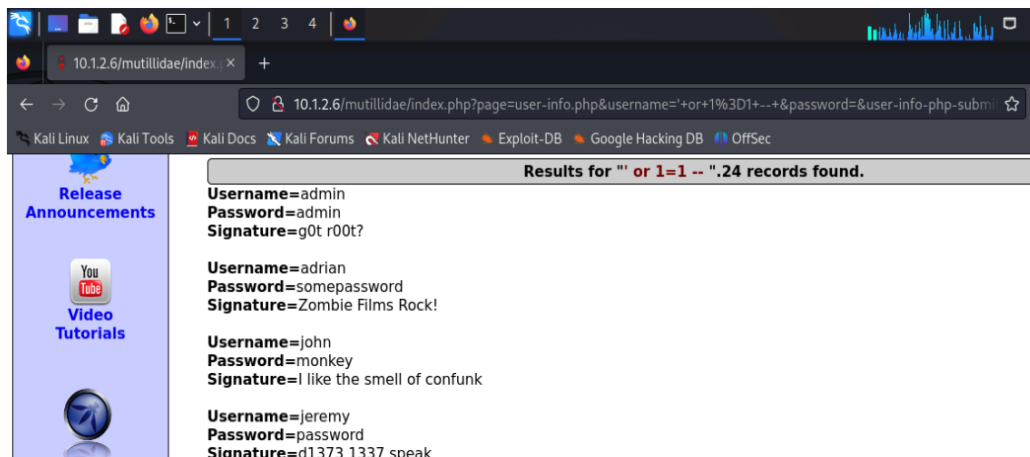
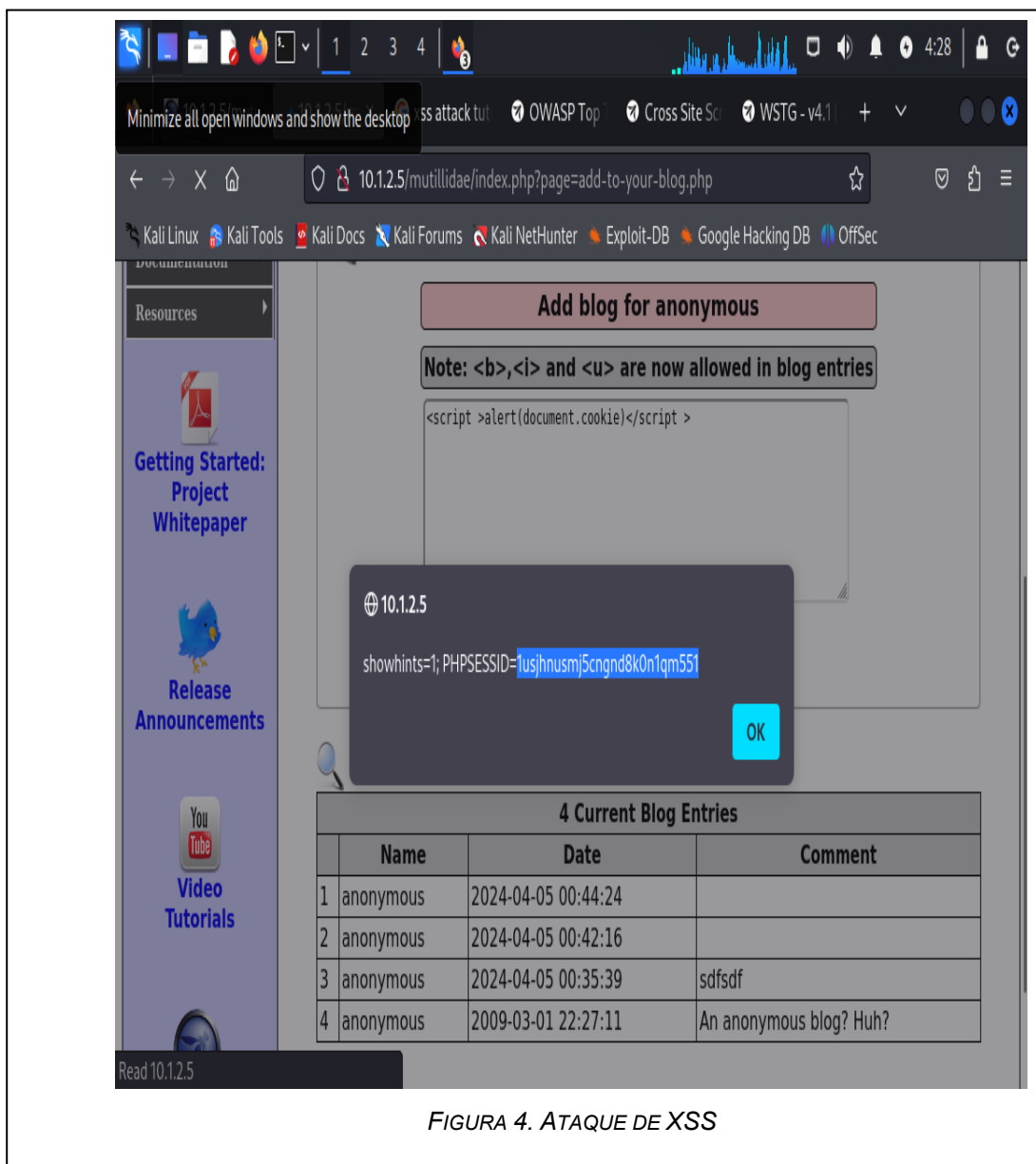


*FIGURA 3. SUCESSO NO ATAQUE DE SQL INJECTION*

O ataque XSS (*Cross Site Scripting*) tem sucesso, como demonstrado na figura 4, quando o usuário insere um script em uma página web de um blog e este script é executado sempre que um usuário carrega essa página web no seu browser.



## 2.3 DESENVOLVIMENTO SEGURO DE APLICAÇÕES WEB

Neste capítulo, é abordado o desenvolvimento seguro de aplicações web, destacando princípios, diretrizes e boas práticas para a construção de sistemas resilientes a ataques cibernéticos. São analisadas as ordagens e ferramentas que auxiliam na implementação da segurança desde as primeiras fases do ciclo de vida do software.

### 2.3.1 PADRÕES DE PROJETO

Padrões de segurança fornecem orientações para a concepção e avaliação de sistemas seguros. Geralmente, eles são apresentados textualmente ou com esquemas, por exemplo, diagramas UML, e são caracterizados por um conjunto de propriedades estruturais e comportamentais. Para autores do estado da arte, as classificações de padrões de segurança carecem de navegabilidade e compreensibilidade. Eles também observaram que todos são concebidos manualmente, através da interpretação de diferentes documentos para encontrar relações abstratas. Justificar estas classificações ou ampliá-las é muitas vezes difícil. Além disso, as relações entre os padrões muitas vezes não são dadas, mas notamos que alguns padrões são compatíveis entre si como o *strategy*, *factory method* e outros são conflitantes como o *singleton*. Como consequência, um designer ainda pode ficar confuso sobre a escolha do padrão (Salva and Regainia, 2019).

Para organizar os padrões, vários esquemas de classificação foram desenvolvidos. Cada esquema de classificação contém um conjunto de critérios ou atributos. Alguns esquemas de classificação importantes e seus critérios são referências para os autores em (Ponde et al., 2017). Na tabela 2 podemos observar detalhes desses padrões.

Tabela 2. Critérios de classificação de padrões de design de segurança. Fonte: (Ponde et al., 2017)

| Atributos       | Critérios de categoria                                                                                            |
|-----------------|-------------------------------------------------------------------------------------------------------------------|
| Interrogativas  | Propósito (Porquê), Dados (O quê), Função (Como), Tempo (Quando), Rede (Onde), Pessoas (Quem) e Scorecard (Teste) |
| Objetivo        | Estrutural, Comportamental, Genérico, Processual, Criacional                                                      |
| Tipo de sistema | Sistemas Disponíveis, Sistemas Protegidos                                                                         |

|                                   |                                                                                                                                                                                                                                                                                                |
|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Estágios do ciclo de vida         | Arquitetura, Requisitos, Análise, Design, Implementação, Integração, Implantação, Operação, Manutenção, Descarte                                                                                                                                                                               |
| Segurança princípios              | Confidencialidade, Integridade, Autenticação, Autorização, Disponibilidade, Confidencialidade, Controle de Acesso, Não Repúdio, Responsabilidade                                                                                                                                               |
| Localização                       | Núcleo, Perímetro, Exterior                                                                                                                                                                                                                                                                    |
| Camadas lógicas                   | Aplicativo, Host, Cliente, Lógica, Dados, Integração, Plataforma e SO, Distribuição, Transporte, Rede                                                                                                                                                                                          |
| Modelo de ameaça                  | Falsificação, adulteração, divulgação de informações, Repúdio, escalada de privilégios, negação de serviço                                                                                                                                                                                     |
| Restrição                         | Regulatória, Organizacional, Humana, Mecanismo                                                                                                                                                                                                                                                 |
| Resposta                          | Evitação, Dissuasão, Prevenção, Detecção, Mitigação, Recuperação, Análise Forense                                                                                                                                                                                                              |
| Fonte de Código                   | Novo, código aberto, tempo de execução, transformação de modelo, assistente código, biblioteca de reutilização, terceirizado, legado, pronto para uso, serviço web remoto                                                                                                                      |
| No âmbito da empresa<br>Segurança | Segurança Empresarial e Gerenciamento de Riscos, Identificação & Autenticação, Controle de Acesso, Acesso ao Sistema Controle, Controle de Acesso ao Sistema Operacional, Contabilidade Padrões, Arquitetura de Firewall, Internet Segura Aplicativos, gerenciamento de chaves criptográficas. |
| Domínio                           | Serviços Web, J2EE, Microsoft                                                                                                                                                                                                                                                                  |

Uma interpretação melhor é fornecida por (Gupta et al., 2020). , que sugere que algumas ferramentas já incluem tais padrões de segurança, mas não são amplamente utilizados pelos desenvolvedores. O autor comenta que os padrões de segurança foram classificados em Segurança Empresarial e Gerenciamento de Riscos, Identificação e Autenticação, Mecanismo de controle de acesso para sistema e sistema operacional, Padrões para registro e contabilidade, padrões de firewall e padrões de gerenciamento de chaves criptográficas. Os padrões na categoria Segurança Corporativa e A Gestão de Risco aborda questões, políticas e diretivas de segurança de alto nível, identificando e avaliando ativos, avaliando diversas ameaças e determinando riscos.

Os padrões de Identificação e Autenticação devem ser integrados com acesso mecanismos de controle para fornecer confidencialidade e integridade. Os padrões de registro, contabilidade e firewall garantem a responsabilização e o monitoramento de quaisquer atividades desviadas em um sistema de software. Os padrões de gerenciamento de chaves criptográficas fornecem soluções para hashing, criptografia e descryptografia e outras políticas de gerenciamento de chaves criptográficas (Gupta et al., 2020).

Para nossa pesquisa consideramos que os padrões de design de segurança são soluções comprovadas para problemas de segurança recorrentes. Portanto, as formas de desenvolver uma aplicação web segura estão relacionadas a tais padrões de segurança que fornecem aos desenvolvedores guias práticos de desenvolvimento, possibilitando mensurar o risco.

### 2.3.2 NORMAS EXISTENTES PARA SEGURANÇA NO DESENVOLVIMENTO

As aplicações cliente/servidor que utilizam o protocolo HTTP para interagir com usuários ou outros sistemas de forma segura são o resultado do progresso que as tecnologias de desenvolvimento de software têm obtido. Algumas das normas amplamente reconhecidas e utilizadas nesse contexto incluem:

- OWASP (Open Web Application Security Project): Fornece informações sobre as principais vulnerabilidades de segurança em aplicações web e oferece diretrizes para mitigar essas vulnerabilidades.
- RFC 7230-7235: Define os protocolos HTTP/1.1, incluindo orientações sobre segurança relacionadas ao protocolo.
- NIST (National Institute of Standards and Technology): Oferece uma série de publicações e diretrizes relacionadas à segurança da informação, incluindo boas práticas para o desenvolvimento seguro de software.
- ISO/IEC 27001: Estabelece requisitos para um sistema de gestão de segurança da informação, que pode ser aplicada ao desenvolvimento seguro de software.

Essas normas e diretrizes fornecem orientações valiosas para ajudar as organizações a desenvolver sistemas cliente/servidor seguros que utilizam o protocolo HTTP, abordando preocupações comuns relacionadas à segurança da informação, proteção contra ameaças cibernéticas e boas práticas de desenvolvimento seguro (KROLL, 2010).

### 2.3.3 SOLUÇÕES PARA DESENVOLVIMENTO SEGURO

A segurança de aplicações web constitui uma disciplina multifacetada, que demanda a integração de diversas técnicas e tecnologias. Entre os elementos fundamentais para garantir a proteção dessas aplicações, destacam-se os testes abrangentes, capazes de avaliar não apenas a funcionalidade, mas também a robustez dos mecanismos de segurança implementados. Conforme apontam Marashdih et al. (2019), a implantação e a proteção de aplicações web representam desafios significativos para desenvolvedores, exigindo metodologias rigorosas de verificação. Nesse contexto, tanto os testes de software quanto os testes de segurança desempenham papéis complementares e igualmente relevantes. Os testes estáticos (caixa branca) e dinâmicos (caixa preta) figuram entre as abordagens mais empregadas.

O teste de caixa branca, também denominado análise estática ou revisão de código, envolve a inspeção minuciosa do código-fonte, da arquitetura e do design do sistema, com o objetivo de identificar vulnerabilidades de segurança. Nessa abordagem, o avaliador possui acesso total ao funcionamento interno da aplicação, o que possibilita a detecção de falhas estruturais e lógicas ainda na fase de desenvolvimento. Diante do aumento contínuo das ameaças cibernéticas, torna-se imprescindível que as ferramentas empregadas para essa análise sejam robustas e confiáveis.

Por sua vez, o teste de caixa preta, ou análise dinâmica, simula a atuação de um agente externo mal-intencionado. Nessa metodologia, o testador não dispõe de informações sobre o código-fonte ou a lógica interna do sistema. A avaliação ocorre por meio da interação direta com a aplicação, reproduzindo o comportamento de um usuário final, de forma a identificar vulnerabilidades exploráveis em cenários reais. Essa técnica é particularmente eficaz, pois permite examinar a aplicação como um todo, englobando a interface do usuário (UI/UX), servidores web e de aplicação, bancos de dados, dependências e sistemas interconectados.

Assim, a combinação de testes de software e testes de segurança integrando tanto a visão interna (caixa branca) quanto a externa (caixa preta) constitui uma prática essencial para assegurar que as aplicações web atendam simultaneamente aos requisitos funcionais e aos padrões de segurança exigidos no cenário atual.

### 2.3.3.1 FERRAMENTAS DE SEGURANÇA

Dois exemplos de ferramentas de scan de vulnerabilidades que podem ser citadas são Owasp ZAP (owasp foundation, 2023) e BurpSuite (Portswigger Ltd., 2023). Essas ferramentas auxiliam nos processos de teste.

OWASP ZAP é uma ferramenta de teste de penetração de aplicativos da web de código aberto que auxilia na identificação de vulnerabilidades e fornece recursos para testá-las. Com uma interface de usuário intuitiva e recursos avançados de verificação, é ideal para iniciantes que tentam entender as vulnerabilidades de aplicativos da web e para profissionais experientes.

Burp Suite é uma das ferramentas mais conceituadas em segurança de aplicações web. É uma plataforma integrada que fornece diversas ferramentas para identificar vulnerabilidades de aplicações web. Com funcionalidades como interceptação de proxy, verificação automatizada e a capacidade de reutilizar os resultados do scanner de vulnerabilidade da web. Embora exista uma versão gratuita disponível, a versão paga oferece recursos estendidos.

Os firewalls chamados WAFs (Web Applications Firewalls) também são um mecanismo de proteção de front-end para aplicativos baseados na Web que estão continuamente sob ataque (Fredj et al., 2021).

Mas não só estes tipos de ferramentas podem ser utilizados para o desenvolvimento e proteção de aplicações web. Atualmente, há um crescente número de ferramentas baseadas em Inteligência Artificial (IA), que apresentam resultados promissores na proteção e detecção de vulnerabilidades de aplicações web (Shahid, 2023), (Thaqi et al., 2022).

A avaliação manual da vulnerabilidade é realizada por um especialista humano, com base em um conjunto específico de instruções destinadas a encontrar a vulnerabilidade de segurança. Este método requer muito tempo, esforço e recursos, e depende fortemente de conhecimento especializado, algo que é amplamente atribuído à sua escassez. A avaliação assistida de vulnerabilidades é realizada com a ajuda de ferramentas ou frameworks de varredura que procuram as fraquezas de segurança mais relevantes.

### 2.3.3.2 BOAS PRÁTICAS DE DESENVOLVIMENTO SEGURO

Podem ser consideradas boas práticas de desenvolvimento seguro:

- Codificação segura: utilização de práticas de programação segura, como sanitização de entrada, tratamento adequado de exceções e uso de funções de segurança fornecidas pela linguagem/framework.
- Validação de dados: verificação cuidadosa dos dados do usuário para evitar ataques, incluindo validação de formulários, filtros e validação de dados de entrada.
- Sanitização de entrada: remoção ou escapamento de caracteres especiais da entrada do usuário para evitar inserção de código malicioso.
- Implementação de princípios de least privilege (privilegio mínimo): atribuição de permissões mínimas necessárias aos usuários com base em suas funções e responsabilidades.
- Utilização de bibliotecas e frameworks seguros : adoção de bibliotecas e frameworks conhecidos por seguir práticas seguras de desenvolvimento (S Viana, 2013), (Cabral, 2022).

### 2.3.4 IMPLANTAÇÃO SEGURA DE APLICAÇÕES WEB

A implantação de aplicações web é considerada pelos autores da literatura uma das etapas mais importantes do processo de desenvolvimento de software, pois a estratégia utilizada para projetar, construir, implementar e testar influencia diretamente a velocidade com que pode responder às mudanças nas preferências ou requisitos do usuário e, muito importante, sua segurança.

Dados os resultados alcançados para a implantação segura de aplicações Web pelos diferentes autores do estado da arte, pode-se concordar com (Aslan et al., 2023) que para uma implantação segura é necessário atender a determinados requisitos, entre os quais podemos mencionar, a validação de entrada, aplicações seguras de terceiros (bibliotecas de códigos), entre outros. Por outro lado, segundo autor (Ponde et al., 2017), é necessário aprofundar e ampliar o uso correto de padrões de design seguros com seus papéis e padrões de sessão.

Além disso, é necessário o desenvolvimento de boas práticas na utilização de ferramentas automatizadas de testes de segurança.

Na prática, a implantação de uma aplicação dependerá do tipo de aplicação em questão, mas em termos gerais pode ser resumida em várias etapas, são elas:

- Planejar: Nesta etapa é traçado o cronograma de implantação da aplicação e avaliada a infraestrutura selecionada.
- Construção e lançamento automáticos: Habilitar a automação é uma etapa fundamental para a implantação de uma aplicação web, pois é necessária para minimizar a possibilidade de erros humanos.
- Integração contínua: a implantação pode ser manual ou automatizada.
- A automatizada é mais recomendada e utilizada atualmente. Neste caso, pode ser utilizada uma metodologia de trabalho CD/CI (implantação contínua, integração contínua, (Maggi, 2020).
- Criar scripts: esta etapa requer a identificação de alterações e diferenças ambientais no aplicativo, executando scripts de teste em um backup antes de passar para a versão final.
- Teste de aceitação: é preciso verificar se tudo está funcionando corretamente testando elementos-chave, como páginas de login.
- Desenvolver o rastreamento de implantação: neste ponto é preciso ativar os serviços de rastreamento para garantir que as equipes de operações possam rastrear facilmente quando as implantações ocorrem e identificar possíveis erros.
- Monitorar e Atualizar: Esta etapa ocorre quando o aplicativo Web é implantado.

Depois disso, é necessário continuar monitorando para que não ocorram erros e corrigindo tudo o que for necessário.

Mas não são apenas as ações na implantação que mantêm uma aplicação web segura. A arquitetura da aplicação desenvolvida também influencia. Nesse caso, na maioria das arquiteturas de aplicações web, devem ser observados os três lados principais do processo:

- Lado do cliente, ou frontend para interação com o usuário.
- Servidor de banco de dados, ou componente adicional para enviar dados do cliente para o servidor.
- Lado do servidor, ou back-end que armazena a lógica de negócios, processa solicitações e envia respostas.

Em cada um destes “lados” a ênfase deve ser colocada no desenvolvimento seguro através de padrões de design seguros (Gupta et al., 2020), (Salva & Regainia, 2019), ferramentas automatizadas de testes de segurança, entre outros (Abdulghaffar et al., 2023), (Alzahrani et al., 2017). É necessário estabelecer frameworks que definam modelos de maturidade que descrevam comportamentos, práticas e processos que devem ser realizados para obter resultados sustentáveis em termos de segurança. Dentre esses frameworks podemos citar o Software Assurance Maturity Model (OpenSAMM) da OWASP (Chandra, 2008), (Fucci et al., 2024).

Essas estruturas de tarefas cruzadas envolvem a adição de diferentes atividades de segurança ao processo de implantação de aplicações web, entre as quais pode-se citar: controles de segurança necessários durante a implantação de aplicações web, como inventário de software autorizado e não autorizado, aplicação de configurações seguras de software em servidores, limitação e controle de portas, protocolos e serviços de rede, uso controlado de privilégios administrativos, administração da configuração, proteção contra negação de serviço, e muito importante, a avaliação contínua de vulnerabilidades e remediações para estas, juntamente com o monitoramento contínuo dos controles de segurança implementados (Hernández Yeja & Porven Rubier, 2016), (Fucci et al., 2024).

Atualmente, estudos na área de Inteligência Artificial mostram um potencial, dada a capacidade de detectar padrões de ataques nunca antes vistos, podendo gerar informações precisas para o correto desenho de estratégias de proteção contra ataques maliciosos. Dentre essas ferramentas concordamos com os autores (Uwagbole et al., 2017), (Z. Li et al., 2019), (X. Li et al., 2021), que mostram as vantagens do uso de aprendizagem automatizada como um sistema de detecção de intrusão que monitora ou, em outros casos, detecta vulnerabilidades no código automaticamente.

### 3 TRABALHOS RELACIONADOS

Neste capítulo, é apresentada a revisão de literatura, contemplando pesquisas e estudos que tratam da segurança no desenvolvimento de aplicações web. São discutidas metodologias, frameworks e soluções já propostas por outros autores, fornecendo embasamento teórico para o estudo.

Foi feito um mapeamento sistemático sobre segurança no desenvolvimento de aplicações Web. A função de fazer um mapeamento sistemático da literatura nesta área é buscar uma compilação abrangente de estudos e pesquisas relevantes. Esta revisão pode ajudar a estabelecer diretrizes e boas práticas para a proteção de sistemas web. Um estudo sistemático ou metanálise é um método estatístico que busca combinar e sintetizar dados de várias pesquisas independentes para responder uma pergunta de pesquisa específica. Ela envolve a análise de dados primários de diversas fontes para formular uma resposta ou uma nova hipótese (Borenstein et al., 2011).

O processo de mapeamento sistemático, por sua vez, é uma abordagem sistemática que busca identificar, avaliar e resumir as evidências disponíveis em uma determinada área de pesquisa. A ideia é criar um mapa das informações existentes para entender as lacunas existentes no conhecimento e orientar futuras investigações (Petticrew and Roberts, 2006).

O processo de mapeamento sistemático foi dividido em três etapas:

- Planejamento e construção do protocolo;
- Seleção de critérios de inclusão e exclusão;
- Escrita dos resultados da revisão.

A crescente utilização de sistemas web para armazenamento de informações sensíveis e realização de transações online torna esses sistemas alvos frequentes de ataques cibernéticos. Essas vulnerabilidades podem permitir que invasores realizem atividades maliciosas, como roubo de informações confidenciais, interrupção de serviços online ou instalação de software malicioso. Portanto, é necessário identificar e corrigir essas vulnerabilidades de segurança em sistemas web para garantir a proteção das informações dos usuários e a disponibilidade dos serviços online. Nesse sentido, a abordagem abrangente para a proteção de aplicações online se torna fundamental para prevenir e mitigar os riscos de segurança em sistemas web. A revisão da literatura sobre esse tema pode fornecer informações valiosas sobre as

técnicas e ferramentas disponíveis para identificar vulnerabilidades em sistemas web e proteger as aplicações online contra possíveis ataques cibernéticos.

O objetivo desta pesquisa é identificar e analisar as vulnerabilidades de segurança comuns em sistemas web, bem como propor técnicas e contramedidas para mitigar essas vulnerabilidades a fim de proteger as aplicações online contra ameaças cibernéticas. Com base neste objetivo, foram definidas as questões de pesquisa (QP):

**QP1:** Quais são as principais vulnerabilidades de segurança encontradas em aplicações web e como elas podem ser identificadas?

**QP2:** Quais são as técnicas mais eficazes para proteger aplicações web contra ameaças emergentes de segurança?

**QP3:** Quais são as tendências, desafios e direções futuras dos testes de segurança em aplicações web?

**QP4:** De que forma as novas técnicas podem auxiliar na identificação de vulnerabilidades de segurança em aplicações web?

A busca de artigos relacionados ao nosso tema foi feita mediante as seguintes strings de pesquisa: ("web application vulnerability assessment") OR ("web application security vulnerabilities") OR ("vulnerability scanning") OR ("OWASP top ten vulnerabilities") OR ("Identification of security vulnerabilities in web systems").

Essas strings ajudaram a encontrar artigos que abordam diferentes aspectos da identificação de vulnerabilidades de segurança em sistemas web e fornecem medidas de proteção eficazes para aplicações online. Para a escolha das bases de dados nos baseamos na vasta quantidade e qualidade de publicações recentes relacionadas ao nosso tema. A busca foi feita em 5 bases de dados diferentes, que são apresentadas na tabela 3. Escolhemos essas plataformas porque deram melhores resultados na busca de trabalhos relacionados ao nosso tema.

Tabela 3. Bases de dados bibliográficas

| Bases de dados | Url                                                                                                                                                                        |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Google Scholar | <a href="https://scholar.google.com.br/">https://scholar.google.com.br/</a>                                                                                                |
| ACM            | <a href="https://dl.acm.org/">https://dl.acm.org/</a>                                                                                                                      |
| IEEE           | <a href="https://ieeexplore.org/xplore/home.jsp">https://ieeexplore.org/xplore/home.jsp</a>                                                                                |
| Elsevier       | <a href="https://www.elsevier.com/sciencedirect">https://www.elsevier.com/sciencedirect</a><br><a href="https://www.sciencedirect.com/">https://www.sciencedirect.com/</a> |
| Springer       | <a href="https://www.springer.com/">https://www.springer.com/</a>                                                                                                          |
| MDPI           | <a href="https://www.mdpi.org">https://www.mdpi.org</a>                                                                                                                    |

### 3.1 SELEÇÃO E EXECUÇÃO

Inicialmente as buscas nas bases de dados apresentadas na tabela 3 retornaram um total de 94 artigos. Em seguida foram realizadas análises dos artigos e aplicados os critérios de inclusão e exclusão. Para reduzir esse número na etapa 1 removemos os duplicados. Em seguida atendemos os artigos que atenderam os critérios descritos na sequência do texto. A Figura 5 representa o processo realizado.

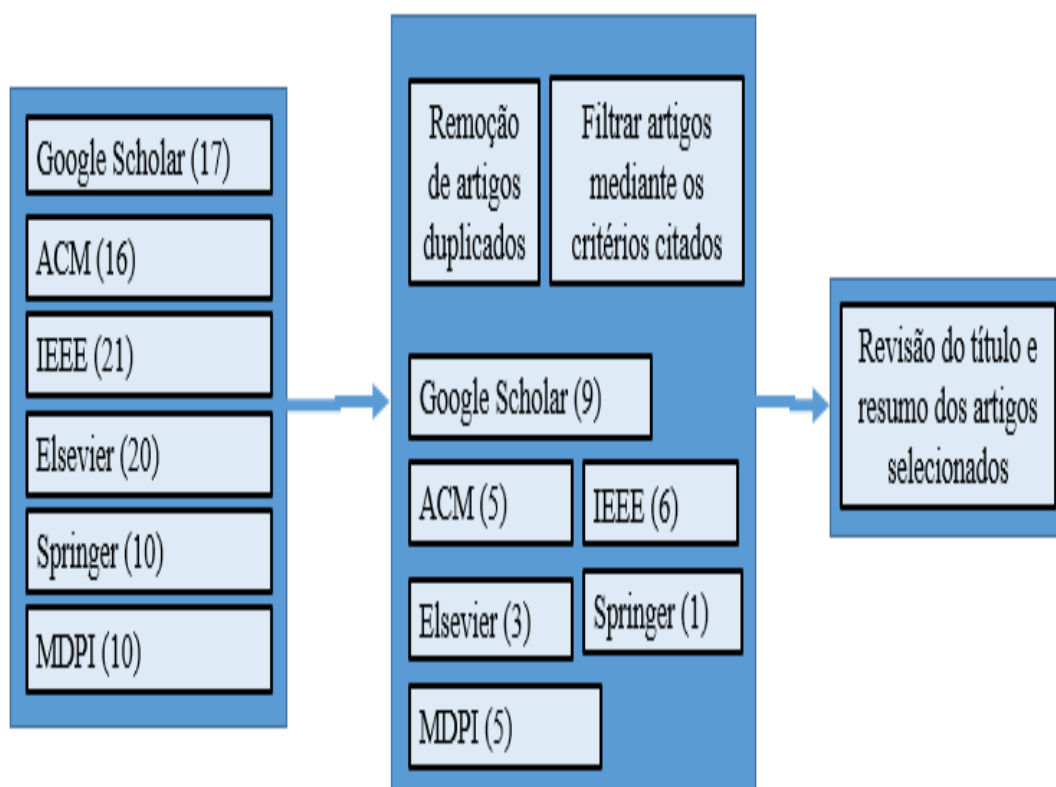


FIGURA 5. PROCESSO EMPREGADO PARA LOCALIZAR E SELECIONAR OS ARTIGOS ANALISADOS

Na etapa 1, algumas possíveis avaliações dos artigos apresentados que levamos em consideração:

- Relevância do tema abordado e da problemática estudada;
- Originalidade e inovação dos métodos e técnicas de análise de vulnerabilidades em sistemas de gestão web;
- Contribuição para o avanço do conhecimento na área de segurança da informação em sistemas web;
- Consistência e qualidade dos resultados obtidos e das possíveis soluções propostas;
- Adequação do referencial teórico e bibliográfico (C. A. N. E. V. F. Santos, 2019).

Na etapa 2, analisamos manualmente o título e o resumo dos 40 artigos resultantes da etapa 1 e removemos os que não atendem os critérios acima listados.

Na etapa 3, revisamos os artigos resultantes da segunda etapa e selecionamos 32 artigos.

### 3.2 RESULTADOS DA REVISÃO

A Tabela 4 lista os 32 artigos selecionados. Com base nas pesquisas realizadas é possível classificar os textos dos artigos como abordagens sobre a segurança de aplicações web, incluindo a detecção de vulnerabilidades, análise de riscos e técnicas de mitigação. Alguns artigos também focam em abordagens específicas usando tecnologias de machine learning ou análise de código estático.

Os artigos de pesquisa selecionados abordam o tema das vulnerabilidades de segurança em aplicações web. Eles exploram diferentes aspectos relacionados à identificação, detecção e mitigação dessas vulnerabilidades, bem como discutem técnicas, abordagens e ferramentas utilizadas para avaliar e prevenir tais vulnerabilidades. Os autores analisam os desafios enfrentados na segurança de aplicativos web e propõem soluções para lidar com essas vulnerabilidades. Esses artigos visam fornecer insights e conhecimento para profissionais de segurança, desenvolvedores e pesquisadores interessados em fortalecer a segurança de aplicativos da web.

Tabela 4. Artigos obtidos

| Trabalhos                                                                                                                                               | Bases de dados | Fontes                           |
|---------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|----------------------------------|
| 1. Automated vulnerability assessment and security testing in web applications                                                                          | Elsevier       | (Yadav et al., 2023)             |
| 2. Secure web application development: identifying and mitigating security vulnerabilities                                                              | IEEE           | (Santra and Dandapat, 2022)      |
| 3. Web application security: current trends and emerging threats                                                                                        | ACM            | (Rai and Mishra, 2022)           |
| 4. Web application security testing: trends, challenges, and future directions                                                                          | ACM            | (Saha and Shamsuddoha, 2021)     |
| 5. Comparative study of web application security vulnerabilities and counter measures                                                                   | ACM            | (Kumar et al., 2019)             |
| 6. Web application security: a comparative study of vulnerability scanning tools                                                                        | Google scholar | (Brown and Wilson, 2022)         |
| 7. Identifying and mitigating OWASP top ten vulnerabilities in web applications                                                                         | Google scholar | (Johnson and Lee, 2020)          |
| 8. Web Application Security: Emerging Threats and Techniques for Protection                                                                             | Google scholar | (Rawat and Bhateja, 2020)        |
| 9. Application Security: A Study on Security Vulnerabilities and Countermeasures                                                                        | Google scholar | (Sethi and Roy, 2019)            |
| 10. Web Application Security through Comprehensive Vulnerability Assessment                                                                             | Elsevier       | (Gandikota et al., 2023)         |
| 11. Enhancing Web Application Security through Automated Penetration Testing with Multiple Vulnerability Scanners                                       | MPDI           | (Abdulghaffar et al., 2023)      |
| 12. Evaluation of Black-Box Web Application Security Scanners in Detecting Injection Vulnerabilities                                                    | MPDI           | (Althunayyan et al., 2022)       |
| 13. An Empirical Comparison of Pen-Testing Tools for Detecting Web App Vulnerabilities                                                                  | MPDI           | (Albahar et al., 2022)           |
| 14. Enhancing Burp Suite with Machine Learning Extension for Vulnerability Assessment of Web Applications                                               | Google scholar | (Thaqi et al., 2022)             |
| 15. Evaluating Performance of Web Application Security Through a Fuzzy Based Hybrid Multi-Criteria Decision-Making Approach: Design Tactics Perspective | IEEE           | (Alenezi et al., 2020)           |
| 16. Detecção heurística de phishing e metodologia de verificação de URL baseada em scraping e web crawling                                              | IEEE           | (R. Almeida e C. Westphall 2020) |
| 17. Implementation of a Security Strengthening System to Mitigate Vulnerabilities in Academic Web Applications                                          | Google scholar | (Márquez et al., 2023)           |
| 18. Metamorphic Security Testing for Web Systems                                                                                                        | Google scholar | (Mai et al., 2019)               |

|                                                                                                                       |                |                               |
|-----------------------------------------------------------------------------------------------------------------------|----------------|-------------------------------|
| 19. A Comprehensive study of Cyber Attacks & Counter Measures for web systems                                         | ACM            | (Jamil et al., 2018)          |
| 20. A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners | IEEE           | (Alazmi and Leon, 2022)       |
| 21. AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection                                         | IEEE           | (Kim et al., 2020)            |
| 22. An Enhanced Anomaly Detection in Web Traffic Using a Stack of Classifier Ensemble                                 | IEEE           | (Tama et al., 2020)           |
| 23. A Comparative Study of Web Application Security Parameters: Current Trends and Future Directions                  | MPDI           | (Shahid et al., 2022)         |
| 24. Hybrid Computational Modeling for Web Application Security Assessment                                             | Google scholar | (Hussain Seh et al., 2022)    |
| 25. A systematic review and taxonomy of web applications threats                                                      | Google scholar | (Sadqi and Maleh, 2022)       |
| 26. Exploring the Benefits of Combining DevOps and Agile                                                              | Google scholar | ( Simões e Lopes 2022)        |
| 27. Review into State of the Art of Vulnerability Assessment using Artificial Intelligence                            | Google scholar | (Khan and Parkinson, 2018)    |
| 28. A mixed-method study of DevOps challenges                                                                         | IEEE           | (Tanzil et al. 2023)          |
| 29. Mobile application development methodologies                                                                      | Springer       | (Molina Ríos et al. 2021)     |
| 30. Web Application Security: Protection from Advanced Persistent Threat                                              | Google scholar | (Yadav and Kumar, 2018)       |
| 31. A catalog that associates security patterns and attack steps to design secure applications                        | ACM            | (Salva e Regainia 2019)       |
| 32. Securing web applications using security patterns                                                                 | IEEE           | (Gupta, Singh Mohapatra 2020) |

### 3.3 DESCRIÇÃO DOS TRABALHOS RELACIONADOS

O trabalho de Almeida e Westphall (2020) propõem um método heurístico para a detecção de phishing, utilizando técnicas de web scraping e web crawling para verificar URLs suspeitas. O estudo baseia-se na extração de características das páginas da web e na análise automática de elementos, como a presença de palavras-chave associadas a ataques de phishing e a estrutura dos links. Embora o foco principal da pesquisa não seja a implantação segura de aplicações web, a abordagem apresentada contribui para a mitigação de ameaças cibernéticas, reduzindo a possibilidade de que usuários sejam vítimas de ataques de engenharia social.

Molina Ríos et al. (2021) realizam um estudo abrangente sobre metodologias de desenvolvimento de aplicações móveis, analisando práticas seguras aplicáveis ao

processo de concepção, design e implementação de sistemas. Os autores enfatizam a necessidade de incorporar estratégias de segurança desde as fases iniciais do desenvolvimento, incluindo revisões de código, testes de segurança e auditorias periódicas. Além disso, a pesquisa discute os desafios da fragmentação do ecossistema mobile, destacando como diferentes plataformas e sistemas operacionais podem demandar abordagens específicas para garantir a proteção contra vulnerabilidades.

Gupta, Singh e Mohapatra (2020) abordam a adoção de padrões de segurança em aplicações web, propondo diretrizes para a implementação de controles que reforcem a proteção contra ameaças cibernéticas. O estudo destaca a importância de frameworks e normas de segurança amplamente reconhecidos, como o OWASP Top 10, e analisa sua aplicabilidade em diferentes contextos de desenvolvimento. Além disso, os autores apresentam um modelo baseado na identificação de riscos e aplicação de contramedidas específicas, com ênfase em práticas como validação de entrada, gerenciamento seguro de sessões e configuração segura de servidores web.

Salva e Regainia (2019) contribuem para essa discussão ao propor um catálogo que associa padrões de segurança a etapas de ataque, facilitando a concepção de aplicações mais seguras desde as fases iniciais do desenvolvimento. O estudo analisa ataques comuns, como injeção de SQL, cross-site scripting (XSS) e ataques de força bruta, e correlaciona essas ameaças com técnicas específicas de mitigação. Os autores defendem que a implementação de padrões de segurança deve ser orientada por um entendimento detalhado do comportamento dos atacantes, permitindo que desenvolvedores projetem aplicações que resistam a tentativas de exploração.

No contexto da automação e integração contínua, Tanzil et al. (2023) investigam os desafios na adoção de DevOps e seu impacto na segurança do software. O estudo identifica que, apesar das vantagens oferecidas pelo DevOps, como maior agilidade e automação de processos, muitas organizações enfrentam dificuldades para incorporar práticas robustas de segurança no ciclo de vida do desenvolvimento de software (SDLC). Os autores sugerem soluções baseadas em Security as Code, enfatizando a necessidade de testes de segurança contínuos e

automação de auditorias para minimizar riscos durante a implantação de aplicações web.

Em uma abordagem complementar, Almeida, Simões e Lopes (2022) exploram os benefícios da combinação entre DevOps e Agile, destacando como essas metodologias podem contribuir para um desenvolvimento mais seguro e eficiente. A pesquisa analisa a integração de ferramentas de segurança dentro dos processos ágeis e como práticas como Continuous Security e DevSecOps podem melhorar a resposta a vulnerabilidades. O estudo reforça a importância da cultura organizacional, apontando que a segurança deve ser uma responsabilidade compartilhada entre desenvolvedores, testadores e equipes de operações. O trabalho de (Yadav et al., 2023) tem como objetivo apresentar uma revisão sobre as técnicas e ferramentas de avaliação automatizada de vulnerabilidades e testes de segurança em aplicações web. Entre as vulnerabilidades abordadas no artigo estão: injeção SQL, XSS (cross-site scripting), CSRF (cross-site request forgery), vulnerabilidades de autenticação e gerenciamento de sessão, manipulação de entrada (input tampering) e injeção de comandos. Quanto ao método utilizado para eliminar as vulnerabilidades apresentados nos artigos, os autores apresentaram técnicas como patching (aplicação de patches), remoção de código malicioso, adição de recursos de segurança de defesa em profundidade e a implementação de melhores práticas de segurança. Este artigo é uma revisão de literatura. Os autores do artigo apontaram sugestões de trabalhos futuros, como: o estudo de técnicas de avaliação automatizada de vulnerabilidades, a análise de métricas para avaliar a eficácia das técnicas de teste e treinamento empresarial em testes de segurança.

Ja o trabalho de (Brown and Wilson, 2022) tem como objetivo realizar um estudo comparativo de ferramentas de varredura de vulnerabilidades em aplicações web. A metodologia utilizada pelos autores consistiu em selecionar e avaliar quatro ferramentas de varredura de vulnerabilidades populares: Acunetix, Burp Suite Professional, Nessus Professional e QualysGuard. Essas ferramentas foram testadas em um ambiente controlado, simulando um ataque a uma aplicação web, e foram comparadas em termos de velocidade e precisão na identificação de vulnerabilidades. As vulnerabilidades abordadas no estudo incluíram: injeção de SQL, cross-site scripting (XSS), inclusão de arquivos locais (LFI), inclusão remota de arquivos (RFI), falsificação de solicitação entre sites (CSRF), bypass de autenticação,

vulnerabilidades de servidor web e vulnerabilidades de servidor de aplicativos. Para eliminar as vulnerabilidades identificadas, os autores recomendaram a implementação das práticas recomendadas de segurança de aplicativos da OWASP (Open Web Application Security Project). O artigo é uma pesquisa empírica que utiliza a abordagem de estudo comparativo. Os resultados do estudo indicaram que a ferramenta Burp Suite Professional foi mais precisa na identificação de vulnerabilidades, enquanto a ferramenta Acunetix foi mais rápida. Os autores também discutiram as limitações do estudo e forneceram recomendações para trabalhos futuros na área como: replicar os estudos em diferentes ambientes para avaliar o desempenho das ferramentas em configurações diferentes, investigar o impacto da ferramentas de varredura de vulnerabilidades no desempenho de precisão, identificar e abordar as limitações do estudo atual como suas ferramentas selecionadas e o ambiente de teste.

O trabalho de (Santra and Dandapat, 2022) apresenta um método de desenvolvimento seguro de aplicativos da web que permite a identificação e mitigação de vulnerabilidades de segurança desde o estágio inicial do processo de desenvolvimento. Metodologia de identificação de vulnerabilidades: A metodologia proposta envolve uma combinação de testes de segurança automáticos e manuais, análise de código estática e revisão do desenho arquitetural do aplicativo. Os testes de segurança automáticos incluem varreduras de vulnerabilidades conhecidas em todo o código-fonte do aplicativo e, em seguida, uma equipe de testes de segurança conduz testes manuais para identificar outras vulnerabilidades. após a identificação das vulnerabilidades, o artigo propôs uma série de medidas para mitigá-las, como melhorias no código-fonte do aplicativo, implementação de controles de segurança e configuração adequada de servidores. Este artigo apresenta uma pesquisa aplicada, pois propõe uma metodologia prática para lidar com questões relevantes e práticas em segurança de aplicativos da web. O artigo sugere que, como trabalho futuro, a metodologia proposta possa ser expandida para incluir a integração de ferramentas de segurança no processo de desenvolvimento, bem como a melhoria do processo de avaliação da eficácia das medidas de segurança implementadas.

O trabalho (Rai and Mishra, 2022) tem como objetivo discutir as tendências atuais e as ameaças emergentes relacionadas à segurança de aplicativos web. O artigo também aborda os principais desafios enfrentados pelos desenvolvedores de

web e as melhores práticas para garantir a segurança dos aplicativos. O artigo não descreve uma metodologia específica para identificação de vulnerabilidades. Em vez disso, ele oferece uma visão geral das vulnerabilidades mais comuns encontradas em aplicativos web e como elas podem ser evitadas. Entre as vulnerabilidades abordadas no artigo estão as seguintes: Injeção de SQL, Cross-site scripting (XSS), Cross-site RequestForgery (CSRF), Vulnerabilidades de autenticação e autorização, Exposição de informações sensíveis, Vulnerabilidades de descriptografia e criptografia. O artigo não descreve um método específico para eliminar vulnerabilidades, mas oferece sugestões e estratégias para minimizar os riscos de segurança, como o uso de bibliotecas de segurança, configurações adequadas de servidor e aplicação e a implementação de SSL /TLS. O artigo é uma revisão da literatura que abrange as tendências atuais e as ameaças emergentes em relação à segurança de aplicativos web. O artigo sugere que futuras pesquisas possam abordar tendências emergentes em ataques cibernéticos, como ataques de homográficos que é uma forma sofisticada de phishing, na qual os cibercriminosos criam sites fakes que se assemelham muito aos sites legítimos, ataques de injeção de HTML e ataques de ransomware. O artigo também recomenda que as pesquisas futuras se concentrem na implementação de mecanismos adicionais de segurança, como autenticação multifatorial e criptografia de ponta a ponta.

O objetivo do trabalho de (Saha and Shamsuddoha, 2021) é identificar as tendências, desafios e direções futuras dos testes de segurança de aplicativos da web. O artigo examina as metodologias usadas para identificar vulnerabilidades comuns em aplicativos da web, discute desafios emergentes para testes de segurança de aplicativos da web e aborda direções futuras na pesquisa de testes de segurança de aplicativos da web. O artigo identifica várias metodologias usadas para testes de segurança de aplicativos da web, incluindo testes de penetração, varreduras automatizadas, testes manuais e revisões de código. O artigo destaca a importância de uma abordagem integrada para testes de segurança e destaca a necessidade de testar aplicativos da web em diferentes camadas, incluindo servidor, aplicativo e cliente. O artigo aborda várias vulnerabilidades comuns em aplicativos da web, incluindo SQL Injection, Cross-Site Scripting (XSS), injeção de código, quebra de autenticação, vulnerabilidades de sessão e problemas de configuração. O artigo também discute a importância da segurança em aplicativos da web IoT. O artigo

destaca a necessidade de testes regulares e contínuos de segurança, bem como a implementação de medidas de segurança, como firewalls, criptografia e autenticação de usuários. O artigo também aborda a importância de treinamento em segurança para desenvolvedores de aplicativos e usuários finais. O artigo é uma revisão da literatura e apresenta uma análise crítica dos trabalhos existentes na área de testes de segurança de aplicativos da web. O artigo baseia-se em fontes primárias e secundárias, incluindo artigos acadêmicos e relatórios de organizações de segurança. O artigo destaca a necessidade de mais pesquisas para abordar novas vulnerabilidades em aplicativos da web, como segurança em aplicativos de IoT e segurança em aplicativos de inteligência artificial. O artigo também sugere a necessidade de integrar testes de segurança de aplicativos da web em processos de desenvolvimento de software para reduzir a ocorrência de vulnerabilidades.

O trabalho de (Rawat and Bhateja, 2020) tem como objetivo explorar as ameaças emergentes à segurança de aplicações web e apresentar técnicas para a proteção contra essas ameaças. Os autores também buscam fornecer orientações e recomendações para desenvolvedores e profissionais de segurança interessados em fortalecer a segurança de suas aplicações web. Os autores utilizaram uma abordagem baseada em pesquisa, onde eles revisaram e analisaram estudos anteriores, relatórios de ameaças e casos reais para identificar as vulnerabilidades emergentes em aplicações web. Eles também consideraram as técnicas e ferramentas disponíveis para explorar essas vulnerabilidades. O artigo aborda uma variedade de vulnerabilidades encontradas em aplicações web, incluindo injeção de SQL, cross-site scripting (XSS), ataques de força bruta, ataques de falsificação de solicitação entre sites (CSRF), vulnerabilidades de autenticação e acesso não autorizado, entre outras. Os autores apresentam uma série de técnicas e melhores práticas para eliminar as vulnerabilidades em aplicações web. Eles discutem a importância de uma programação segura, validação de entrada de dados, sanitização de entrada, uso correto de autenticação e autorização, implementação de firewalls de aplicação web, atualização regular de software e adoção de uma abordagem em camadas para a segurança. O artigo é uma revisão de literatura baseada em pesquisa, onde os autores revisam diversas fontes e estudos anteriores para identificar as ameaças emergentes e apresentar técnicas de proteção.

Já o trabalho (Johnson and Lee, 2020) visa identificar e mitigar as dez principais vulnerabilidades listadas pela OWASP (Open Web Application Security Project) em aplicações web. Os autores procuram fornecer um panorama abrangente dessas vulnerabilidades e também apresentar técnicas eficazes para mitigá-las. Os autores realizaram uma revisão da literatura e análise de estudos anteriores para identificar as principais vulnerabilidades listadas pela OWASP em aplicações web. Eles também examinaram relatórios de casos reais e consideraram as técnicas e ferramentas disponíveis para identificar essas vulnerabilidades. Os autores abordam as dez principais vulnerabilidades listadas pela OWASP, que incluem: injeção de SQL, cross-site scripting (XSS), quebra de autenticação e gerenciamento de sessões, exposição de dados sensíveis, vulnerabilidades de controle de acesso, falhas de segurança em configurações, cross-site requestforgery (CSRF), uso de componentes com vulnerabilidades conhecidas, validação de entrada inadequada e redirecionamento e encaminhamento inseguros. Os autores apresentam diferentes técnicas e melhores práticas para mitigar cada uma das vulnerabilidades listadas pela OWASP. Eles enfatizam a importância da validação e sanitização de entrada de dados, uso correto da autenticação e gerenciamento de sessões, criptografia de dados sensíveis, controle de acesso adequado, configuração segura de servidores, implementação de mecanismos para evitar ataques CSRF, atualização de componentes com vulnerabilidades conhecidas e adoção de práticas de validação de entrada e saída seguras. O artigo é uma revisão de literatura baseada em pesquisa, onde os autores fazem uma análise e revisão de estudos anteriores para identificar e mitigar as vulnerabilidades listadas pela OWASP.

O trabalho de (Kumar et al., 2019) realizou um estudo comparativo das vulnerabilidades de segurança em aplicações web e as contramedidas disponíveis para mitigar essas vulnerabilidades. Os autores procuram fornecer uma visão geral abrangente das vulnerabilidades mais comuns em aplicações web e das melhores práticas para mitigá-las. Os autores realizaram uma revisão da literatura para identificar as vulnerabilidades mais comuns encontradas em aplicações web. Eles revisaram estudos anteriores, relatórios de segurança e recomendações de organizações de segurança para identificar essas vulnerabilidades. Os autores abordam diversas vulnerabilidades de segurança em aplicações web, incluindo: injeção de SQL, cross-site scripting (XSS), quebra de autenticação e gerenciamento

de sessões, exposição de dados sensíveis, vulnerabilidades de controle de acesso, cross-site requestforgery (CSRF), configuração inadequada de segurança, validação de entrada insuficiente e redirecionamento e encaminhamento inseguros. Os autores apresentam uma análise das principais contramedidas e melhores práticas disponíveis para mitigar as vulnerabilidades identificadas. Eles enfatizam a importância de implementar práticas seguras de codificação, validar e sanitizar a entrada de dados, utilizar mecanismos adequados de autenticação e controle de acesso, criptografar dados sensíveis, configurar corretamente os servidores, implementar mecanismos anti-CSRF e garantir a validação adequada de entrada e saída. O artigo é uma revisão de literatura baseada em pesquisa, onde os autores fazem uma análise comparativa das vulnerabilidades de segurança em aplicações web e suas contramedidas.

No trabalho (Yadav and Kumar, 2018), os autores estudam ataques baseados na web. Os autores sugerem que exijamos aplicativos de proteção da web para proteger nosso sistema contra tais ataques. Além disso, as técnicas mais comuns usadas pelo invasor são técnicas adaptativas de phishing e spear phishing, cross-site scripting e injeção de SQL. Neste trabalho os autores descrevem um modelo que ajuda a compreender vários vetores de ataque comumente usados para lançar e ativar ataques APT. Em segundo lugar, propõem uma estrutura que pode ser usada como guia para analisar, prevenir e detectar atividades de APT na aplicação, no sistema e na rede (Yadav and Kumar, 2018).

O trabalho de (Sethi and Roy, 2019) fornece uma visão geral das vulnerabilidades de segurança mais comuns em aplicações web e discute as técnicas utilizadas para preveni-las. A metodologia utilizada para identificação das vulnerabilidades foi uma revisão sistemática da literatura. Os pesquisadores examinaram artigos e relatórios publicados em várias bases de dados, incluindo ACM, IEEE, Google Scholar e Scopus, para identificar as vulnerabilidades mais comuns encontradas em aplicações web. As vulnerabilidades discutidas no artigo incluem: ataques de injeção (SQL, LDAP, comando, etc.), autenticação e gerenciamento de sessões, cross-site scripting (XSS), cross-site requestforgery (CSRF), vulnerabilidades de segurança de arquivos, vulnerabilidades de segurança de configuração e gerenciamento de logs. As técnicas mencionadas no artigo para prevenir essas vulnerabilidades incluem: codificação segura, validação de entrada de

usuários, segregação de privilégios, autenticação forte e gerenciamento de sessões, uso de políticas de segurança, limite de entrada de dados, entre outras. O tipo de pesquisa realizada neste artigo é uma revisão sistemática da literatura. Trabalhos futuros sugeridos pelos autores incluem a realização de pesquisas adicionais para identificar outras vulnerabilidades e potenciais soluções para resolver problemas específicos de segurança em aplicações web.

O documento fornece uma visão geral completa dos testes de avaliação e penetração de vulnerabilidades do aplicativo da Web, enfatizando a necessidade de medidas proativas de segurança para proteger dados confidenciais e preservação da integridade do aplicativo. O estudo apresentado como objetivo é identificar e classificar as vulnerabilidades dos aplicativos da Web que podem comprometer a segurança do aplicativo e resultar em violações de dados do usuário. A principal abordagem deste projeto é a segurança e a privacidade dos dados do usuário. Integração de práticas de segurança no ciclo de vida do desenvolvimento de software através da investigação de estudos de caso real e padrões do setor. No final, a pesquisa apresentada pelos autores aumenta o crescente corpo de conhecimento na segurança dos aplicativos da Web, ajudando as organizações a criar defesas resistentes contra ameaças cibernéticas (Gandikota et al., 2023).

Os testadores de penetração adotaram cada vez mais scanners de teste de penetração múltiplos para garantir a robustez dos aplicativos da Web. No entanto, uma limitação notável de muitas técnicas de varredura sua suscetibilidade à produção de falsos positivos. Na pesquisa apresentada pelos autores, um novo quadro é introduzido projetado para automatizar a operação de vários scanners de vulnerabilidade de aplicativos da Web (WAV, waveform audio file format) dentro da plataforma (ASLEDLE ou plataforma de aprendizagem online. A estrutura gera um relatório de vulnerabilidades combinadas usando dois algoritmos: um algoritmo de automação e um novo algoritmo de combinação que produz listas abrangentes de vulnerabilidades detectadas. A estrutura aproveita as habilidades de duas varreduras de vulnerabilidade da web, usando as ferramentas Arachni e OWASP ZAP. O estudo começa com uma extensa revisão da literatura científica existente, com foco nas ondas de código aberto( ou recursos digitais disponibilizados de forma aberta e colaborativa para que pessoas de todo mundo possam acessar, utilizar e modificar) e explorando o OWASP 2021. Depois disso, a fase de desenvolvimento da estrutura

aborda o desafio de diferentes resultados obtidos de diferentes ondas. O objetivo central dessa estrutura é combinar os resultados de múltiplos WAV em um relatório de vulnerabilidade consolidado, melhorando as taxas de detecção e a segurança geral. O estudo mostra que os resultados combinados produzidos pela estrutura proposta pelos autores obtêm maior precisão em comparação com os resultados individuais da varredura obtidos por Arachni e OWASP ZAP. Essa investigação é importante porque revela que a lista de organizações que representam os interesses dos trabalhadores excede os scanners individuais, particularmente no que diz respeito à retirada e à medida F ou métrica relacionada a vulnerabilidades em sistemas de softwares. Consequentemente, a adoção de múltiplos scanners de vulnerabilidade é recomendada como efetiva para reforçar a detecção de vulnerabilidade em aplicações da Web (Abdulghaffar et al., 2023).

Para encontrar vulnerabilidades da Web perante um invasor, os especialistas em segurança usam scanners de vulnerabilidades de aplicativos da Web de caixa preta para verificar as vulnerabilidades da segurança nos aplicativos da Web. A maioria dos estudos avalia e usa scanners de caixa preta em vários aplicativos Web vulneráveis. No entanto, a maioria dos aplicativos testados de forma tradicional (não-dinâmica) e não reflete a situação atual da web. No presente trabalho, os autores avaliam a precisão de diferentes varreduras para a detecção de vulnerabilidade do aplicativo da Web. As vulnerabilidades comprovadas são a injeção de código, em particular a injeção de linguagem de consulta estruturada (SQLI ou sql injection), não apenas o SQL (NOSQL) e a injeção de modelo lateral do servidor (SSTI ou injeção de template do servidor). Os scanners de caixa preta também foram avaliados de quatro maneiras para identificar suas limitações. Os resultados do estudo mencionados mostram que os scanners de caixa preta ignoram a maioria das vulnerabilidades, uma vez que todos os modos e alguns scanners perderam todas as vulnerabilidades (Althunayyan et al., 2022).

Atualmente, os aplicativos da Web devem ser protegidos e atender a todos os requisitos de segurança. Os testadores de penetração devem garantir que os invasores encontrem fraquezas para destruir, explorar ou disseminar informações na Web. Portanto, o uso de ferramentas automatizadas de avaliação de vulnerabilidades é a melhor e mais fácil para os aplicativos de Pentesting na Web, mas essas ferramentas têm pontos fortes e fracos. Portanto, o uso da ferramenta incorreta pode

transportar vulnerabilidades esperadas ou conhecidas que podem abrir portas para ataques cibernéticos. Na pesquisa desenvolvida, é apresentada uma comparação empírica das ferramentas de teste de penetração para detectar vulnerabilidades de aplicativos da Web usando padrões e métodos aprovados para facilitar a seleção de ferramentas apropriadas de acordo com as necessidades dos testadores de penetração. Os autores do trabalho mediram as habilidades da ferramenta usando uma análise comparativa baseada na pontuação. Os resultados mostraram que o Burp Suite Professional obtém os melhores resultados por ferramentas comerciais, enquanto o OWASP Zap obteve o maior número de pontos do lado das ferramentas não comerciais (Albahar et al., 2022).

No documento (Thaqi et al., 2022), os autores apresentam uma nova abordagem para os algoritmos de aprendizado automático de última geração aplicados à extensão da suíte Burp. Esses algoritmos foram usados para inserir injeção de SQL, falsificação de aplicações cruzadas e vulnerabilidades da entidade em aplicativos Web da universidade. Os resultados mostram que os algoritmos estudados podem ser uma alternativa para adicionar segurança à web (Thaqi et al., 2022).

O design do software pode ter um grande impacto na segurança geral do software. O design seguro de aplicativos da web é um desafio que depende de decisões diferentes e difíceis que determinam a segurança final. O aumento no número de vulnerabilidades aumenta o nível de requisitos de segurança. Portanto, as táticas de design de segurança serão adotadas para atender a esses requisitos de segurança. As táticas de design de segurança são os mecanismos para definir, detectar e mitigar vulnerabilidades e ataques. Portanto, as falhas na aplicação de táticas de segurança ou seu enfraquecimento durante a manutenção do aplicativo da Web podem ser uma das razões por trás do aparecimento de novas e graves vulnerabilidades. É necessária uma análise aprofundada das táticas de segurança e sua priorização para determinar o fator mais priorizado. Isso ajudará ainda mais a obter um sistema mais seguro. No estudo de pesquisa apresentado pelos autores, eles usaram um método híbrido de TOPSIS AHP nebuloso (processo de hierarquia analítica para ordem de referência para a solução ideal da similaridade) para a avaliação das táticas de design de segurança e seus atributos. A eficiência da abordagem proposta foi validada em um aplicativo da Web real, obtendo bons

resultados. Além disso, diferentes aplicativos da Web da universidade foram usados para aumentar os resultados obtidos. Estudos como os apresentados pelos autores contribuirão para a avaliação do design de aplicativos da Web mais desafiadores para os ataques cibernéticos (Alenezi et al., 2020).

Os autores da investigação descrita em (Alsaedi, 2022) propõem uma abordagem que aborda as dificuldades apresentadas nos aplicativos da Web através do uso de técnicas de análise dinâmica da Black Box. Também realizam uma análise de validação do lado do cliente para aumentar a cobertura e, portanto, identificar mais vulnerabilidades. Além disso, a abordagem proposta com uma ferramenta é implementada e avaliada sua eficácia usando aplicativos da Web do mundo real. Finalmente, o sistema descobriu URLs exclusivos, enviou 102 formulários da Web com sucesso e descobriu 2 vulnerabilidades automaticamente. Uma comparação detalhada das abordagens de última geração mostrou que a proposta as excede em cobertura, o número de vulnerabilidades detectadas e desempenho (Alsaedi, 2022).

O trabalho de (Márquez et al., 2023) propõe a implementação de um sistema para fortalecer a segurança dos aplicativos acadêmicos da Web com base em um estudo realizado em uma universidade pública no Equador. Dentro de uma análise comparativa, o OWASP é selecionado como a metodologia de teste básica, OWASP ZAP como uma ferramenta de teste de penetração e, dentro da varredura de vulnerabilidade, as dez (10) categorias da metodologia OWASP 2021 são levadas em consideração. Os autores do trabalho se concentraram em dois (2) cenários de teste: exame das vulnerabilidades nos aplicativos da Web em produção; e, aplicação da metodologia proposta, mas em um ambiente controlado. De acordo com os dados obtidos a partir dos cenários, é contrastado que a aplicação desse sistema de fortalecimento de segurança reduz os aplicativos da Web em aplicativos da Web acadêmica em 98,83%, e com o teste estatístico do quadrado do Chi com o nível de varredura de 95%, é demonstrado Essa metodologia reduz o número de vulnerabilidades, resultando no uso de aplicativos acadêmicos mais seguros (Márquez et al., 2023).

No trabalho de (Hussain Seh et al., 2022), os autores propõem fornecer segurança eficaz especificamente para a área de saúde. O trabalho desenvolve um modelo computacional híbrido de tomada de decisão multicritério baseado no processo de hierarquia analítica e na técnica de ordem de preferência por similaridade

a soluções ideais no ambiente fuzzy vacilante. Os conjuntos difusos de ondulação fornecem soluções eficazes para resolver problemas de tomada de decisão, onde os especialistas neutralizam as dúvidas para tomar uma decisão. O esforço de pesquisa proposto ajudará designers e desenvolvedores a identificar, selecionar e priorizar os melhores atributos de segurança para o desenvolvimento de aplicações web. A análise empírica dos autores conclui que a robustez recebeu a maior prioridade entre os atributos de segurança avaliados, seguida pela criptografia, autenticação, acesso limitado, acesso de acesso, validação de dados e manutenção de trilhas de auditoria. Os resultados deste esforço de pesquisa representam que este procedimento computacional proposto seria o mecanismo mais familiar para determinar a segurança de aplicações web. O estudo também estabelece diretrizes que os desenvolvedores podem consultar para identificar e priorizar atributos de segurança para criar aplicações com uma base web mais segura e confiável (Hussain Seh et al., 2022).

Os testes de segurança verificam se os dados e recursos nos sistemas de software estão protegidos contra invasores. Infelizmente, ele sofre do problema do oráculo, que se refere ao desafio, dado o input de um sistema, de distinguir o comportamento correto do incorreto. Em muitas situações, ao testar vulnerabilidades potenciais, um oráculo de teste pode não existir ou pode não ser prático devido às muitas entradas para as quais oráculos específicos devem ser definidos. Neste trabalho, os autores propõem uma abordagem de teste metamórfico que alivia o problema do Oracle em testes de segurança. Ele permite que os engenheiros especifiquem relações metamórficas (MR) que capturam as propriedades de segurança do sistema. Esses MRs são então usados para automatizar testes e detectar vulnerabilidades. Em resumo, é proposto um catálogo de 22 MRs independentes de sistema para automatizar testes de segurança em sistemas web. Como resultado da investigação, 39% das atividades de testes de segurança da OWASP não são automatizadas utilizando técnicas de última geração. Ele detectou automaticamente 10 das 12 vulnerabilidades que afetam dois sistemas amplamente utilizados, um comercial e um de código aberto (Jenkins) (Mai et al., 2019).

No trabalho de (Jamil et al., 2018), os autores mapeiam diferentes tipos de ataques cibernéticos, brechas que tornam possível o hacking e contramedidas para prevenir ataques de hackers. Além disso, eles desenvolvem diversos experimentos para detectar esses ataques e testar contramedidas. Finalmente, eles sugerem que

contramedidas devem ser utilizadas por desenvolvedores, testadores de penetração e especialistas em segurança para garantir a segurança dos sistemas web.

As aplicações Web têm sido um dos principais alvos de violações de segurança bem-sucedidas nos últimos anos. Atualmente, eles se protegem, como método principal, procurando vulnerabilidades com ferramentas especializadas chamadas scanners de vulnerabilidade de aplicativos da web. Embora estas abordagens de testes dinâmicos tenham algumas vantagens, ainda existem poucos estudos que exploram sistematicamente as suas características e capacidades de detecção. Este artigo relata os resultados de uma revisão sistemática da literatura (RSL) para analisar as características e a eficácia do scanner de vulnerabilidades de web mais utilizados. Um total de 90 artigos de pesquisa foram cuidadosamente avaliados. Trinta (30) WVS foram coletados e relatados, e apenas 12 tiveram pelo menos uma avaliação quantitativa de eficácia. Esses 12 WVS foram avaliados usando 15 estudos de avaliação originais. Descobrimos que essas avaliações testaram principalmente apenas dois dos dez principais tipos de vulnerabilidade do Open Web Application Security Project (OWASP): injeção de SQL (SQLi) (13/15) e script entre sites (XSS) (8/15). Além disso, apenas um artigo avaliou seis dos dez principais tipos de vulnerabilidade do OWASP e apenas um scanner. Eles também descobriram que as taxas de detecção relatadas eram muito diferentes entre essas 15 avaliações. Com base nestes resultados surpreendentes, são sugeridos caminhos para direções futuras (Alazmi and Leon, 2022).

O aprendizado profundo tem sido amplamente aplicado para detectar problemas de diversos ataques à rede. No entanto, nenhum caso sobre segurança de rede mostrou aplicações de vários algoritmos de aprendizagem profunda em serviços em tempo real além das condições experimentais. Além disso, devido à integração da computação de alto desempenho, é necessária a aplicação de sistemas que possam lidar com tráfego em grande escala. Por esse motivo, no presente trabalho, os autores desenvolvem um sistema de Detecção de Intrusão baseado em Inteligência Artificial. A interessante proposta consiste em um modelo ideal de rede neural convolucional e rede de memória de curto prazo (CNNLSTM), codificação de caracteres UTF-8 normalizada para aprendizagem de características espaciais (SFL), para extrair adequadamente características do tráfego HTTP em tempo real. entropia ou compressão. Os autores demonstram a excelência do modelo proposto através de

experimentos repetidos em dois conjuntos de dados públicos (CSIC-2010, CICIDS2017) e dados fixos em tempo real. Ao treinar cargas que analisaram verdadeiros ou falsos positivos com uma ferramenta de rotulagem, o AI-IDS distingue ataques sofisticados, como padrões desconhecidos, ataques embaralhados ou ofuscados, de tráfego benigno. É um sistema flexível e escalável que é implementado com base em imagens Docker, separando funções definidas pelo usuário por imagens independentes. Também ajuda a escrever e melhorar regras do Snort para IDS baseados em assinaturas com base em padrões recém-identificados. Como o modelo calcula a probabilidade maliciosa por meio de treinamento contínuo, ele pode analisar com precisão ataques desconhecidos na Web (Kim et al., 2020).

Um sistema de proteção contra ataques na web é essencial na era da informação atual. Neste artigo, os autores propõem sistemas de detecção de intrusão baseados em anomalias em uma aplicação web. Ao contrário do empilhamento convencional, onde alguns classificadores fracos são predominantemente usados, o conjunto empilhado proposto é uma arquitetura de conjunto, mas seus classificadores básicos são outros classificadores de conjunto, ou seja, floresta aleatória, máquina de aumento de gradiente e XGBoost. Para demonstrar a generalização do modelo proposto, dois conjuntos de dados que são usados especificamente para detecção de ataques em uma aplicação web, ou seja, CSIC-2010v2 e CICIDS-2017, são utilizados no experimento. Além disso, o modelo proposto supera significativamente as técnicas existentes de detecção de ataques na web em termos de precisão e métricas de taxa de falsos positivos. Os resultados de validação dos conjuntos de dados CICIDS-2017, NSL-KDD e UNSW-NB15 também melhoram aqueles obtidos utilizando algumas técnicas recentes. Finalmente, o desempenho de todos os algoritmos de classificação é analisado em termos de um teste de significância estatística em duas etapas, proporcionando uma contribuição de valor agregado à literatura atual (Tama et al., 2020).

Um scanner de vulnerabilidade de aplicativos da Web disponível publicamente é um programa de computador que avalia a segurança de aplicativos da Web empregando técnicas automatizadas de teste de penetração que reduzem o tempo, o custo e os recursos necessários para testes de penetração de aplicativos da Web e eliminam a dependência dos engenheiros de teste do conhecimento humano. No entanto, esses scanners de segurança têm vários pontos fracos: eles não verificam

aplicativos web inteiros e geram resultados de testes incorretos. Por esta razão, os autores desta pesquisa avaliam o desempenho de scanners de vulnerabilidade de aplicações web testando aplicações vulneráveis definidas intencionalmente e o nível de sua respectiva precisão e exatidão. Além disso, eles conduzem pesquisas intensivas para enumerar quantitativamente os resultados dos scanners de segurança de aplicações web para inspecionar sua eficácia e limitações. Além disso, como resultado, foi possível classificar as ferramentas analisadas utilizando os parâmetros mais comuns. A avaliação é baseada em uma lista de vulnerabilidades extraídas do OWASP (Open Web Application Security Project) (Shahid et al., 2022).

Os autores (Sadqi and Maleh, 2022) desenvolvem uma análise de várias taxonomias robustas na literatura sobre ataques. Eles expõem sua classificação, vantagens e limitações para cada taxonomia. Globalmente, o objetivo desta análise, sugerem os autores, é fornecer um resumo das vantagens e desvantagens de cada classificação. Como resultado, a taxonomia proposta aproveita os benefícios das taxonomias existentes e fornece uma abordagem integrada para classificar ataques do lado do cliente e do lado do servidor. Essa descoberta, segundo os autores, ajudará os pesquisadores a encontrar uma taxonomia clara e detalhada de diferentes áreas relacionadas às aplicações web.

### 3.3.1 ANÁLISE COMPARATIVA DOS TRABALHOS RELACIONADOS

A tabela 5 mostra um resumo dos trabalhos relacionados e as técnicas de classificação. Foram identificadas as seguintes características comuns nas descrições dos trabalhos: tipos de ataques descritos, ferramentas usadas, técnicas de mitigação sugeridas, metodologia seguida, descrição de testes, ambiente de redes, uso de inteligência artificial e proposição de uso de métricas para medição.

Embora a análise tenha considerado um total de 31 artigos, um subconjunto de 15 estudos foi selecionado para uma análise comparativa mais detalhada. A escolha desses artigos baseou-se em critérios adicionais, como inovação metodológica, impacto na área e aplicabilidade dos resultados.

Segue uma descrição breve de cada coluna para orientar a leitura da tabela antes de observar os valores “Sim” e “Não”:

- Ref. - Indica a referência bibliográfica (autor e ano) do estudo ou proposta analisada.
- Ataques - Informa se o trabalho abordou ou analisou ataques a aplicações web.
- Ferramentas - Indica se o estudo utilizou ferramentas específicas para análise, teste ou segurança.
- Combinação de Ferramentas - Mostra se houve uso conjunto de mais de uma ferramenta para complementar a análise.
- Mitigação - Assinala se o trabalho apresentou medidas ou técnicas para mitigar vulnerabilidades.
- Metodologia - Indica se o estudo seguiu uma metodologia estruturada para a execução das atividades.
- Testes - Mostra se o trabalho incluiu a execução de testes (de segurança ou funcionais) como parte do estudo.
- Redes - Informa se houve uso ou análise de redes (por exemplo, redes locais, redes simuladas ou internet real) no contexto do estudo.
- IA - Indica se foi utilizada Inteligência Artificial para apoiar a análise, detecção ou mitigação de vulnerabilidades.
- Uso de Métricas - Assinala se o estudo aplicou métricas quantitativas ou qualitativas para avaliar desempenho, eficácia ou segurança.

Tabela 5. Análise comparativa dos trabalhos

| Ref.                       | Ataques | Ferramentas | Combinação de Ferram. | Mitigação | Metodologia | Testes | Redes | IA  | Uso de Métricas |
|----------------------------|---------|-------------|-----------------------|-----------|-------------|--------|-------|-----|-----------------|
| Almeida & Westphall (2021) | Sim     | Sim         | Sim                   | Sim       | Sim         | Sim    | Não   | Não | Não             |
| Miliona Rios et al. (2024) | Sim     | Não         | Não                   | Sim       | Não         | Não    | Não   | Não | Não             |
| Gupta et al. (2020)        | Sim     | Sim         | Não                   | Não       | Não         | Sim    | Não   | Não | Não             |
| Selva & Regina (2019)      | Sim     | Sim         | Sim                   | Sim       | Sim         | Sim    | Não   | Não | Não             |
| Yadav et al. (2021)        | Não     | Não         | Sim                   | Sim       | Sim         | Sim    | Sim   | Não | Não             |
| Rai & Mishra (2022)        | Sim     | Sim         | Sim                   | Sim       | Sim         | Sim    | Não   | Não | Não             |
| Albahar et al. (2022)      | Sim     | Sim         | Sim                   | Sim       | Sim         | Sim    | Não   | Não | Não             |
| Jamil et al. (2018)        | Sim     | Sim         | Sim                   | Sim       | Sim         | Sim    | Não   | Não | Não             |
| Ali Tanzi et al. (2023)    | Sim     | Sim         | Sim                   | Sim       | Sim         | Sim    | Não   | Não | Não             |
| Yadav & Kumar (2018)       | Sim     | Sim         | Sim                   | Sim       | Sim         | Sim    | Sim   | Não | Não             |
| Almeida et al. (2022)      | Sim     | Sim         | Sim                   | Não       | Sim         | Não    | Não   | Não | Não             |
| Brown & Wilson (2021)      | Sim     | Sim         | Não                   | Sim       | Não         | Sim    | Não   | Não | Não             |
| Santra (2022)              | Sim     | Sim         | Sim                   | Sim       | Sim         | Sim    | Sim   | Não | Não             |
| Hoque & Shamsuddoh (2021)  | Não     | Sim         | Sim                   | Sim       | Sim         | Sim    | Não   | Não | Não             |
| Rawat & Bhateja (2022)     | Sim     | Sim         | Sim                   | Sim       | Sim         | Sim    | Sim   | Não | Não             |
| M-MADAWS (Proposta)        | Sim     | Sim         | Sim                   | Sim       | Sim         | Sim    | Não   | Sim | Não             |

### 3.3.2 RESPOSTAS DAS QUESTÕES DE PESQUISA

Na presente seção serão abordadas as respostas as questões de pesquisa (RQP) definidas na subseção 3.1.

**RQP1.** Esta seção responde “Quais são as principais vulnerabilidades de segurança encontradas em aplicações web e como elas podem ser identificadas?”

Hoje em dia as páginas web podem ser atacadas de diversas formas, sendo as mais comuns e não menos complexas de prevenir, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF) e Injeção de SQL, e discute como essas vulnerabilidades podem ser identificadas por meio do uso dessas ferramentas. O artigo "Web Application Security: A Comparative Study of Vulnerability Scanning Tools" (Brown and Wilson, 2022) realiza um estudo comparativo de diferentes ferramentas de varredura de vulnerabilidades em aplicações web. O estudo identifica vulnerabilidades comuns. No presente estudo confirmamos o que foi observado no estado da arte, onde dos trabalhos revisados 9/31 (29%) desenvolvem estudos críticos sobre as vulnerabilidades comumente encontradas, onde são destacadas as vulnerabilidades mencionadas anteriormente e outras menos comuns, mas de igual importância que possam ser o uso de software de terceiros.

**RQP2.** Resposta a “Quais são as técnicas mais eficazes para proteger aplicações web contra ameaças emergentes de segurança?”

Atualmente existem técnicas endossadas como corretas para garantir a segurança das páginas web contra ameaças emergentes. Estas incluem, de acordo com investigadores de última geração, como validar a entrada do utilizador, utilizar encriptação apropriada e implementar mecanismos de autenticação e autorização. Porém, o uso de testes automatizados de segurança, como estáticos ou dinâmicos, vem ganhando força nesse campo. Na revisão de literatura realizada neste trabalho pudemos observar que 12/31 (38%) trabalhos analisados comentaram sobre avaliação de vulnerabilidades de segurança, por outro lado, 7/31 (22%) trabalhos desenvolveram estudos sobre identificação, detecção e prevenção de vulnerabilidades. Em aspectos gerais, as sugestões de segurança mencionadas anteriormente, que podem ser aplicadas em todo o processo de desenvolvimento, são adequadas para a proteção de aplicações web.

**RQP3.** Resposta a “Quais são as tendências, desafios e direções futuras dos testes de segurança em aplicações web?”

Em relação à segurança de aplicações web, hoje em dia existem tendências que influenciam inclusive no desenvolvimento das próprias aplicações, propondo ambientes seguros e automatizados. Pudemos perceber no estudo sistêmico realizado que entre as principais tendências estão a utilização de testes dinâmicos de Inteligência Artificial. No entanto, reconhece-se que a utilização destes por si só não é tudo, mas sim a sua combinação com outras ferramentas. Na revisão pudemos encontrar estudos recentes que mostram as vantagens do uso de novas ferramentas. Onde 11/31 (35%) empregos utilizam aprendizado de máquina como técnicas avançadas para detecção e prevenção de vulnerabilidades.

**RQP4.** Respondendo a pergunta: “De que forma as novas técnicas podem auxiliar na identificação de vulnerabilidades de segurança em aplicações web?”

Atualmente, o aprendizado de máquina constitui um ramo da Inteligência Artificial cuja aplicação está se expandindo para diversos campos e com muito bons resultados. A segurança de computadores e a segurança de aplicações web, mais especificamente, não são exceção. Diferentes contribuições foram desenvolvidas, por exemplo, em o artigo “Machine Learning for Detection and Mitigation of Web Vulnerabilities and Web Attacks”(Shahid, 2023), aborda diferentes ameaças emergentes, como ataques Cross-Site Scripting (XSS) e Cross-Site Request Forgery (CSRF). Além disso, os autores do estudo analisam diferentes formas de proteção por meio de aprendizagem automatizada.

Tais métodos para sua aplicação requerem grandes quantidades de dados, caso contrário seus resultados (precisão) poderão ser diminuídos. Porém, o estado da arte garante que, no presente e no futuro da segurança de aplicações web, ferramentas baseadas em aprendizado de máquina constituirão a base da análise de vulnerabilidades, servindo de apoio a especialistas, gerando conteúdo de ataques e simulando-os, entre outras questões relevantes para a área de segurança de servidores. Conseguimos constatar em nossa revisão da literatura científica que eles podem sim fornecer pistas sobre como os ataques se desenvolvem, sendo ferramentas de grande valor para as equipes de segurança. Dos trabalhos revisados,

11/31 (35%) destes referem-se ao uso dessas ferramentas, sua combinação com outros tipos de ferramentas e suas contribuições.

Conseguimos verificar em nossa pesquisa, onde 7/31 (22%), que as ferramentas que ampliam o deep learning estão sendo aplicadas principalmente na detecção de ataques Cross-Site Scripting (XSS) e Cross-Site Request Forgery (CSRF), embora consigam resultados muito bons em outras vulnerabilidades de acordo com os autores em “Efficiency and Effectiveness of Web Application Vulnerability Detection Approaches: A Review” (Zhang et al., 2022). Apesar das suas vulnerabilidades, entre as quais podemos citar a dificuldade de explicar as suas previsões ou o elevado custo computacional necessário para o treinamento, a aprendizagem profunda constitui uma ferramenta fundamental para o desenvolvimento da segurança real. Onde, por um lado, tais algoritmos são capazes de distinguir scripts maliciosos e benignos com alta precisão e por outro lado, permitem o monitoramento em tempo real da segurança das aplicações.

### 3.4 LEVANTAMENTO DE PROBLEMAS E DESAFIOS PARA A SEGURANÇA NO DESENVOLVIMENTO DE APLICAÇÕES WEB

Os aplicativos da Web têm naturalmente uma grande superfície de ataque e talvez recursos com um grande número de vários elementos potencialmente vulneráveis. Autores de última geração sugerem que, mesmo quando estamos confiantes devido a um servidor web seguro rodando em um sistema operacional que é conhecido por ser seguro, falhas de segurança podem permanecer porque são principalmente devidas a erros de programação no desenvolvimento da aplicação web. ciclo de vida ou erros na configuração do servidor que o hospeda (Hoffman, 2020). Portanto, o Open Web Application Security Project (OWASP) fornece o projeto Top Ten, que é uma lista das 10 falhas de segurança de aplicações Web mais perigosas atuais, juntamente com métodos eficazes para lidar com essas falhas (Johnson and Lee, 2020).

Como principal desafio de segurança para a segurança das aplicações web, podemos apontar três aspectos que englobam esses principais problemas, tais aspectos são enumerados a seguir:

- Como validação de entrada de usuário
- Segurança de componentes de terceiros

- Implementação de mecanismos de autenticação e autorização.

Primeiramente, não validar corretamente a entrada de dados de parte dos usuários pode rastrear a injeção de código indesejada (Thomé et al., 2018), (Althunayyan et al., 2022), onde dados não confiáveis são enviados a um intérprete como parte de um comando ou consulta. Os dados hostis do invasor podem induzir o intérprete a executar comandos não intencionais ou acessar dados não autorizados.

Segundo aspecto, os componentes de terceiros são um dos pontos de acesso favoritos dos invasores porque a maioria dos aplicativos da web os incorpora. Usar bibliotecas ou frameworks é uma prática comum em nossos dias. Ele permite que as equipes de desenvolvedores desenvolvam aplicativos de forma rápida e não menos importante com o apoio da comunidade de desenvolvedores. No entanto, esses componentes de terceiros estão altamente expostos a ataques na web. E para piorar a situação, quanto mais popular é um componente, mais ele é alvo de ataques. Na verdade, quando uma falha de segurança é descoberta num componente, muitas vezes esta torna-se pública (Hoffman, 2020).

Terceiro, é certamente o aspecto mais visado pelos atacantes maliciosos, que desejam “aumentar os seus privilégios” para atingir os seus fins. A escalada de privilégios ocorre quando um usuário explora uma falha, falha de design ou erro de configuração em uma aplicação, para obter alto acesso a recursos que normalmente deveriam ser inacessíveis. Normalmente, os invasores começam explorando uma vulnerabilidade de direitos de acesso em um sistema ou aplicativo alvo, permitindo-lhes substituir as limitações da conta de usuário atual. O autor sugere que, com esse tipo de ataque, eles podem acessar recursos e dados de outro usuário (Hoffman, 2020).

### 3.4.1 SUGESTÕES DE POSSÍVEIS SOLUÇÕES PARA OS PROBLEMAS

A avaliação de vulnerabilidades em aplicações web pode ser abordada de diversas formas conforme mencionado nos estudos apresentados nas diferentes seções desenvolvidas anteriormente. Resumindo, pode ser manual, assistido (automatizado) e inteligente (totalmente automatizado). Cada um tem suas vantagens e desvantagens, onde a avaliação manual é desenvolvida por um especialista humano com base em um conjunto específico de instruções destinadas a encontrar a

vulnerabilidade de segurança (Hoffman, 2020). Este método requer muito tempo, esforço e recursos, e depende fortemente de conhecimento especializado, algo que é amplamente atribuído à sua escassez. Por outro lado, a avaliação de vulnerabilidade assistida (automatizada) é realizada com a ajuda de ferramentas ou frameworks de varredura geralmente atualizados e que procuram as fraquezas de segurança mais relevantes (Alazmi and Leon, 2022), (Zhang et al., 2022). No entanto, a falta de flexibilidade, compatibilidade e manutenção regular das ferramentas, por conterem conhecimento estático, torna-as obsoletas e não fornecem informações benéficas (em termos de profundidade e âmbito dos testes) sobre o estado de segurança. A avaliação de vulnerabilidade inteligente (totalmente automatizada) aproveita técnicas de IA para produzir decisões semelhantes às de especialistas, sem assistência humana, e é considerada de longe, segundo especialistas de última geração, o método mais desejável. pode-se citar a redução de tempo e dinheiro para o usuário final, a capacidade de reconhecer anomalias, a possibilidade de descobrir conhecimento e explicá-lo (Pan et al., 2019), (Kim et al., 2020). De forma geral, dos estudos previamente revisados neste trabalho de pesquisa, podem ser extraídas diversas sugestões de possíveis soluções para problemas de vulnerabilidade em aplicações web, dentre estas podemos citar:

- Teste de segurança automatizados
- Técnicas avançadas de detecção de vulnerabilidades
- Integração de segurança no ciclo de vida do desenvolvimento de software.
- Implementar padrões de segurança para projetos de aplicação web

## 4 METODOLOGIA MISTA PARA AUXILIAR O DESENVOLVIMENTO DE APLICAÇÕES WEB

A maior dependência de aplicativos web levou a um aumento na superfície de ataque, tornando a segurança da implantação uma prioridade. Ataques cibernéticos, como injeção de SQL, script entre sites (XSS - Cross-site Scripting) e falsificação de solicitação entre sites (CSRF - Cross-site Request Forgery), provaram ser devastadores para organizações que não conseguem implementar medidas de segurança adequadas. De acordo com o OWASP Top Ten, essas vulnerabilidades estão entre as dez mais críticas e devem ser tratadas com urgência (OBFREDJ et al., 2021; LI, 2020).

Existem diferentes formas e estratégias agrupadas em metodologias para desenvolver processos para a segurança na implantação de aplicações web. No entanto, elas são muito diversas e possuem objetivos distintos (ALMEIDA; WESTPHALL, 2020; MOLINA RÍOS et al., 2021).

Uma metodologia é definida como um conjunto sistemático de métodos, técnicas e procedimentos utilizados para realizar uma pesquisa ou projeto (MOLINA RÍOS et al., 2021). Clareza na definição da metodologia é essencial para a reprodutibilidade e compreensão do problema a ser resolvido, no nosso caso, a implantação segura de aplicações web. O uso de metodologias mistas, que combinam elementos de diferentes abordagens, também pode ser interessante. Cada tipo de metodologia tem suas próprias vantagens e desvantagens, o que implica que a escolha deve ser baseada nos objetivos específicos do caso.

De acordo com o estado da arte descrito no capítulo 3, os componentes comuns para uma metodologia de implantação segura são os seguintes:

- **Planejamento e Construção:** Esta etapa envolve a criação de um cronograma para a implementação do aplicativo e a avaliação da infraestrutura selecionada. Os controles de segurança devem ser estabelecidos desde o início do processo de desenvolvimento.
- **Automação de Implantação:** Habilitar a automação é essencial para reduzir a possibilidade de erro humano. Metodologias de integração contínua (CI) e implantação contínua (CD) são essenciais para garantir que os aplicativos sejam entregues de forma segura e eficiente.

- Teste de segurança: Ferramentas de escaneamento automatizadas, como OWASP ZAP, Burp Suite e Wapiti, devem ser aplicadas para identificar vulnerabilidades conhecidas. Essas ferramentas permitem análise de segurança proativa e devem ser complementadas com testes manuais para abordar vulnerabilidades mais sofisticadas.
- Validação e Sanitização de Entrada: A validação adequada da entrada do usuário é crucial para evitar ataques de injeção. Bibliotecas de segurança devem ser usadas para garantir que a entrada seja tratada adequadamente.
- Monitoramento e Atualização: Uma vez que o aplicativo tenha sido implantado, é vital continuar monitorando seu desempenho e segurança. Ferramentas de monitoramento devem ser implementadas para detectar e responder a incidentes de segurança em tempo .
- Uso da Inteligência Artificial: A implementação de algoritmos de machine learning pode melhorar a detecção de padrões de ataque e a resposta a incidentes. Isso inclui o uso de redes neurais para identificar comportamento anômalo no tráfego da web.

#### 4.1 METODOLOGIA MISTA PARA AUXILIAR O DESENVOLVIMENTO DE APLICAÇÕES WEB SEGURAS (M-MADAWS)

Nesta dissertação, é proposta uma Metodologia Mista para Auxiliar o Desenvolvimento de Aplicações Web Seguras (M-MADAWS), que usa métricas como precisão e taxa de detecção para medir o desempenho da implantação.

Pretende usar de forma mista, ferramentas de análise automática e ferramentas que utilizam inteligência artificial. Além disso, propõe o uso de métricas como precisão e taxa de detecção para medir o desempenho da implantação.

A arquitetura da metodologia proposta é representada na figura 6. Essa figura conecta os diferentes componentes que integram todo o fluxo de trabalho da metodologia.

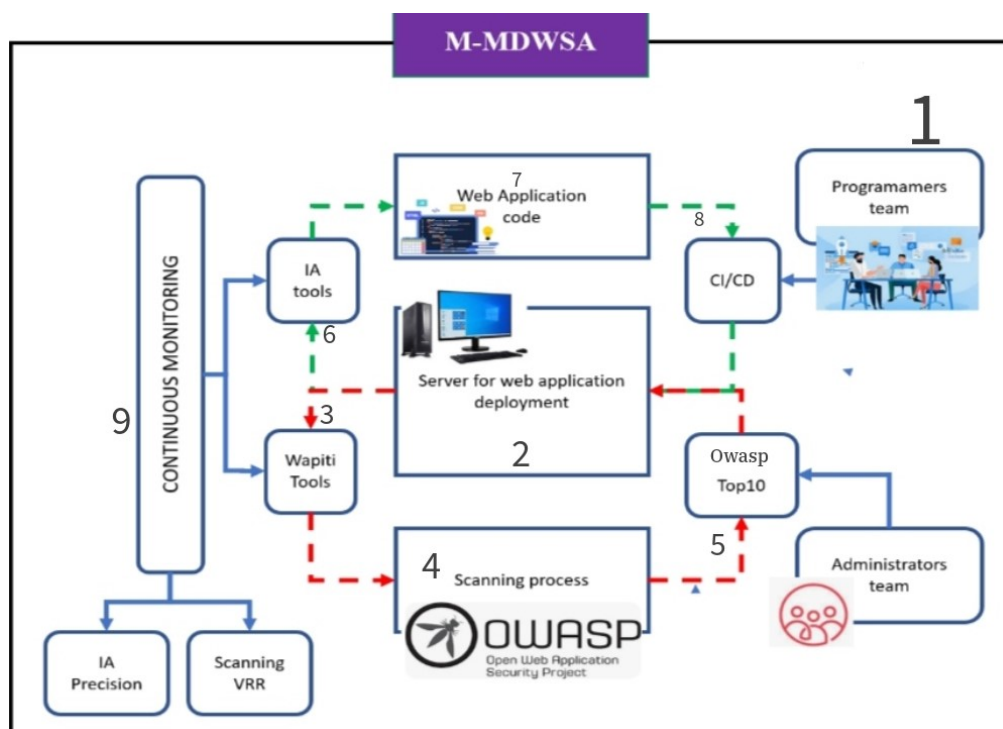


FIGURA 6. ARQUITETURA DA METODOLOGIA M-MADAWS

A metodologia desenvolvida na nossa proposta baseia-se na concretização das seguintes etapas descritas na sequência do texto. As etapas 3, 4 e 6 podem ser repetidas algumas vezes, de acordo com o gestor do projeto, antes da implantação da aplicação.

Segue uma explicação que conecta as 9 etapas à numeração da figura, de forma que o leitor compreenda o fluxo lógico e a função de cada elemento.

1. Definição do modelo de ameaças do ambiente - O processo inicia com a identificação das possíveis ameaças ao ambiente, considerando a aplicação cliente, o servidor e o banco de dados. Essa etapa fornece a base para a análise de riscos.
2. Definição de requisitos de segurança do ambiente e implantação no servidor - A partir das ameaças identificadas, são estabelecidos os requisitos de segurança e aplicadas as configurações necessárias no servidor, garantindo uma infraestrutura inicial protegida.

3. Varredura inicial com a ferramenta Wapiti - Utiliza-se uma ferramenta de varredura automatizada para identificar vulnerabilidades existentes na aplicação antes da aplicação de correções.
4. Desenvolvimento do plano de testes e análise de vulnerabilidades - Com base nos resultados da varredura, elabora-se um plano de testes detalhado para investigar as vulnerabilidades encontradas e avaliar o seu impacto.
5. Consulta ao OWASP Top 10 - As vulnerabilidades identificadas são confrontadas com a lista atualizada das dez ameaças mais críticas de segurança em aplicações web, segundo a OWASP, garantindo que as medidas estejam alinhadas com as melhores práticas.
6. Integração de ferramentas baseadas em Inteligência Artificial - São incorporadas soluções de IA para auxiliar na detecção, classificação e priorização de vulnerabilidades, aumentando a eficiência do processo.
7. Desenvolvimento ou correção do código da aplicação - A aplicação é desenvolvida ou ajustada para eliminar as vulnerabilidades detectadas, seguindo padrões de codificação segura.
8. Integração Contínua (CI/CD) com varredura diária automatizada - Implementa-se um pipeline de integração e entrega contínua, onde testes e varreduras de segurança são executados de forma automática e regular.
9. Monitoramento contínuo - Após a implantação, a aplicação permanece sob vigilância constante, com monitoramento ativo para detectar e responder rapidamente a novas ameaças ou falhas.

#### 4.1.1 MÉTRICAS DE AVALIAÇÃO

Nossa proposta propõe o uso de métricas a serem aplicadas no caso de ferramentas de escaneamento. Essas métricas têm a intenção de medir o auxílio que as ferramentas de escaneamento proporcionam na metodologia proposta.

- Taxa de detecção de vulnerabilidade (VDR): Mede a porcentagem de vulnerabilidades detectadas em comparação ao número total de scans realizados.

$$\text{VDR} = \frac{\text{Identified vulnerabilities}}{\text{Number of scans}}$$

- Taxa de resolução de vulnerabilidade (VRR): mede a taxa de solução de vulnerabilidades resolvidas pelo número total de vulnerabilidades identificadas.

$$\text{VRR} = \frac{\text{Vulnerabilities resolved}}{\text{Identified vulnerabilities}}$$

As métricas para análise dos resultados da aplicação de ferramentas baseadas em inteligência artificial são: precisão, recall e desempenho geral. Essas métricas são comuns em ambientes de aplicação de ferramentas de aprendizagem automatizada.

- Precisão: essas métricas medem a proporção de verdadeiros positivos (VP). A primeira considera o total de positivos previstos (verdadeiros positivos + falsos positivos), enquanto a segunda se baseia no total de positivos reais (verdadeiros positivos + falsos negativos).

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

- Recall: mede a capacidade do modelo de identificar corretamente os exemplos de uma determinada classe.

$$\text{Recall} = \frac{\text{VP}}{\text{VP} + \text{FN}}$$

- Desempenho geral: refere-se à capacidade do modelo em fazer previsões corretas.

$$= \frac{\text{VP} + \text{VN}}{\text{Total de amostras}}$$

## 4.2 APLICAÇÕES E FERRAMENTAS USADAS NOS EXPERIMENTOS

Para avaliar o passo 4 da metodologia e verificar a eficácia das ferramentas de detecção de vulnerabilidades, foi realizada uma série de experimentos para identificar falhas do tipo XSS (Cross-Site Scripting) e CSRF (Cross-Site Request Forgery) nas seguintes aplicações: DVWA (Damn Vulnerable Web Application), SC-Minfo and SG-ZIP. O objetivo era garantir que houvesse um cenário realista para a análise.

DVWA é uma aplicação intencionalmente vulnerável, disponível no site <https://github.com/digininja/DVWA>, que pode ser usada em testes.

A aplicação real SC-MINFO, foi desenvolvida pelo primeiro autor deste artigo e está hospedada em servidor da mediateca de Angola. É um sistema de gestão de equipamentos informáticos. Foi desenvolvido com HTML, (PHP, Dreamweaver, JQuery, JavaScript, db Maria) assim como metodologia ágil Scrum. A aplicação real SG-ZIP é um sistema de gestão académica que serve para gerir arquivos de planos de professores do ensino primário em Angola. Foi desenvolvido em HTML, (PHP, Dreamweaver, JQuery, JavaScript, MySQL) assim como metodologia ágil XP. Não houve a aplicação da metodologia de implantação nas aplicações SC-MINFO e SG-ZIP.

Dessa maneira, há a motivação nesse trabalho em verificar essas aplicações legadas para analisar a possibilidade de aplicação da metodologia desenvolvida. Quanto às ferramentas, foram utilizadas duas distintas nos experimentos:

- Wapiti: Ferramenta de varredura de vulnerabilidades em aplicações web (scanner), de código aberto, que executa uma análise dinâmica do site.
- DeepCode é uma solução baseada em inteligência artificial que realiza análise estática de código, buscando trechos potencialmente vulneráveis. Analisa repositórios de código e fornece recomendações e feedback instantâneo sobre questões relacionadas a bugs, vulnerabilidades e práticas ruins de codificação, adaptando-se continuamente a novos padrões de código. Contudo, as ferramentas não corrigem automaticamente o código. O fluxo de uso é o seguinte:
  1. As ferramentas identificam os problemas;
  2. O DeepCode fornece recomendações e soluções para mitigação (ex.: substituir funções inseguras, sanitizar saídas, incluir token CSRF);
  3. O programador aplica as correções no código.

Na ferramenta Wapiti, foram configurados parâmetros como URL do alvo, profundidade de varredura e módulos ativos (por exemplo, XSS, CSRF, SQL injection etc.). Foi executada a ferramenta permitindo que ela percorresse as páginas do site em busca de possíveis pontos de injeção ou requisições suspeitas. A figura 7 mostra o resultado obtido como saída do scanner Wapiti feito na aplicação teste SC-MINFO em busca de XSS e CSRF.

A adoção de ambas se justifica pela complementaridade: enquanto o Wapiti verifica a aplicação em execução, o DeepCode analisa o código-fonte. Essa

combinação aumenta a cobertura e precisão da detecção. Wapiti funciona em qualquer aplicação web acessível via HTTP/HTTPS, independentemente da linguagem utilizada (PHP, Python, Node.js, Java, etc.). DeepCode suporta diversas linguagens como JavaScript, TypeScript, Java, Python, C#, PHP, Go, entre outras.

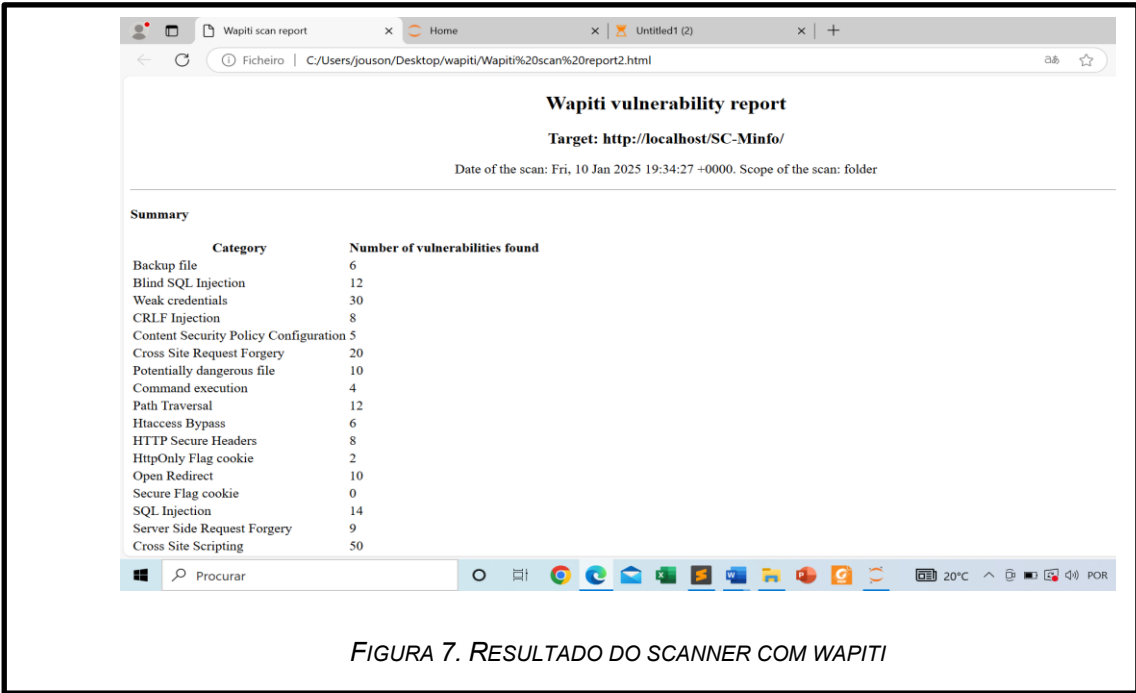


FIGURA 7. RESULTADO DO SCANNER COM WAPITI

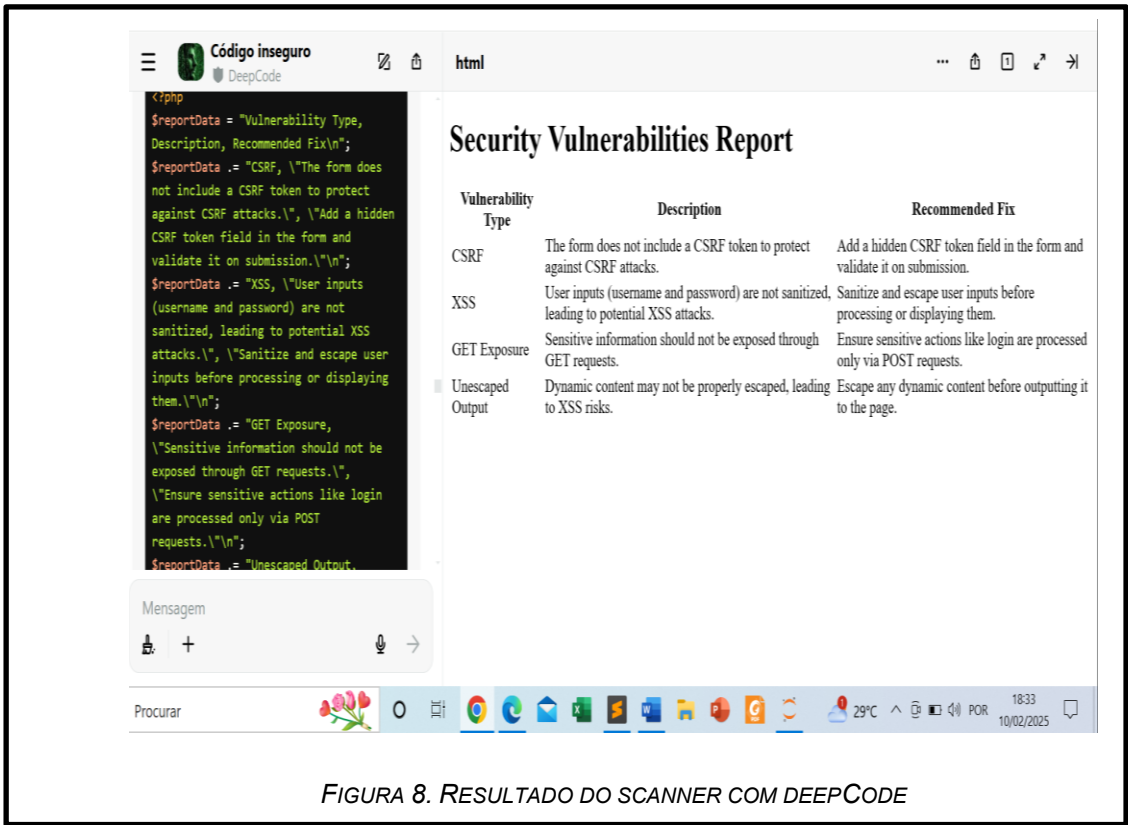


FIGURA 8. RESULTADO DO SCANNER COM DEEPCODE

No uso da ferramenta DeepCode, foi carregado o repositório de código-fonte das aplicações em um ambiente compatível. Foram também habilitadas as regras de segurança voltadas especificamente para detecção de XSS e CSRF. Foi realizada a análise estática, em que a IA do DeepCode examina o código em busca de padrões que indicam falhas ou trechos vulneráveis. A figura 8 demonstra o resultado obtido com a ferramenta DeepCode na aplicação SC-Minfo.

#### 4.3 RESULTADOS EXPERIMENTAIS

Para cada aplicação foram executadas as duas ferramentas – Wapiti e DeepCode separadamente, verificando as vulnerabilidades XSS e CSRF detectadas. No caso da Wapiti (análise dinâmica), foram verificados os alertas e foi feita a tentativa de reproduzir a exploração da falha.

No caso do DeepCode (análise estática), foi analisado se o código reportado como vulnerável realmente representava um risco de XSS ou CSRF. Mostradas as estatísticas dos dados recuperados no feedback obtido da varredura do servidor para CSRF e XSS. Começando com análise XSS, 46 detectadas e 42 resolvidas de 50 vulnerabilidades do usuário na experimentação. Este resultado representa perto de 96% das identificadas e 91% resolvidas respectivamente. A alta taxa de detecção sugere que as ferramentas de varredura automática são eficazes neste ambiente. No entanto, a taxa de resolução indica que há 9% de vulnerabilidades que não foram abordadas, o que pode ser uma área para melhoria. Treinamento adicional para administradores com habilidades médias a baixas pode ser benéfico. Após a execução dos testes, foram analisados os resultados consolidados nas tabelas 6 e 7.

As vulnerabilidades estimadas correspondem a uma análise prévia realizada pelo programador, antes da aplicação de ferramentas automatizadas como o Wapiti e o DeepCode. Essa etapa baseia-se no conhecimento técnico e na experiência prática do desenvolvedor, permitindo a identificação inicial de potenciais falhas de segurança no código.

No contexto desta pesquisa, a análise manual do programador apontou um número superior de vulnerabilidades em comparação com as ferramentas utilizadas. Contudo, essa relação pode variar em outros cenários: em determinados casos, as

ferramentas de varredura conseguem detectar vulnerabilidades que não foram percebidas na avaliação humana.

Assim, a etapa de vulnerabilidades estimadas deve ser entendida como um complemento às análises automatizadas, reforçando a importância da integração entre avaliação humana e ferramentas de varredura para alcançar maior precisão na detecção de falhas de segurança.

#### 4.3.1 CONSIDERAÇÕES SOBRE A FERRAMENTA WAPITI

Nesta subseção, é avaliada a eficácia da ferramenta de escaneamento automático para identificar vulnerabilidades no servidor onde a página da web está armazenada. Seu resultado pode ser influenciado em grande medida por vários fatores com configuração do servidor, nível de conscientização de segurança por administradores, força das ferramentas de proteção, entre outros. Nas tabelas 6 e 7 são mostradas as estatísticas dos dados recuperados no feedback obtido da varredura do servidor para CSRF e XSS. Começando com análise XSS, 46 detectadas e 42 resolvidas de 50 vulnerabilidades do usuário na experimentação. Este resultado representa perto de 96% das identificadas e 91% resolvidas respectivamente. A alta taxa de detecção sugere que as ferramentas de varredura automática são eficazes neste ambiente. No entanto, a taxa de resolução indica que há 9% de vulnerabilidades que não foram abordadas, o que pode ser uma área para

melhoria. Treinamento adicional para administradores com habilidades médias a baixas pode ser benéfico.

Tabela 6. Resultados para tipos de ataques CSRF

| Vulnerabilidades<br>CSRF |           | Ferramenta de scan |           |      |      | Ferramenta IA |          |
|--------------------------|-----------|--------------------|-----------|------|------|---------------|----------|
|                          | Estimadas | Detectado          | Resolvido | VDR  | VRR  | Detectado     | Precisão |
| 1 Ensaio                 | 4         | 4                  | 3         | 1,00 | 0,75 | 4             | 1,00     |
| 2 Ensaio                 | 8         | 8                  | 7         | 1,00 | 0,87 | 7             | 0,87     |
| 3 Ensaio                 | 8         | 7                  | 5         | 0,87 | 0,71 | 6             | 0,75     |
| Total de CSRF            | 20        | 19                 | 15        | 0,95 | 0,77 | 17            | 0,87     |

Tabela 7. Resultados para tipos de ataques XSS

| Vulnerabilidade<br>XSS |           | Ferramentas de scan |           |      |      | Ferramenta IA |          |
|------------------------|-----------|---------------------|-----------|------|------|---------------|----------|
|                        | Estimadas | Detectado           | Resolvido | VDR  | VRR  | Detectado     | Precisão |
| 1 Ensaio               | 8         | 8                   | 7         | 1,00 | 0,87 | 6             | 0,75     |
| 2 Ensaio               | 12        | 11                  | 11        | 0,91 | 1,00 | 9             | 0,75     |
| 3 Ensaio               | 30        | 27                  | 24        | 0,90 | 0,88 | 29            | 0,96     |
| Total de XSS           | 50        | 46                  | 42        | 0,96 | 0,91 | 44            | 0,82     |

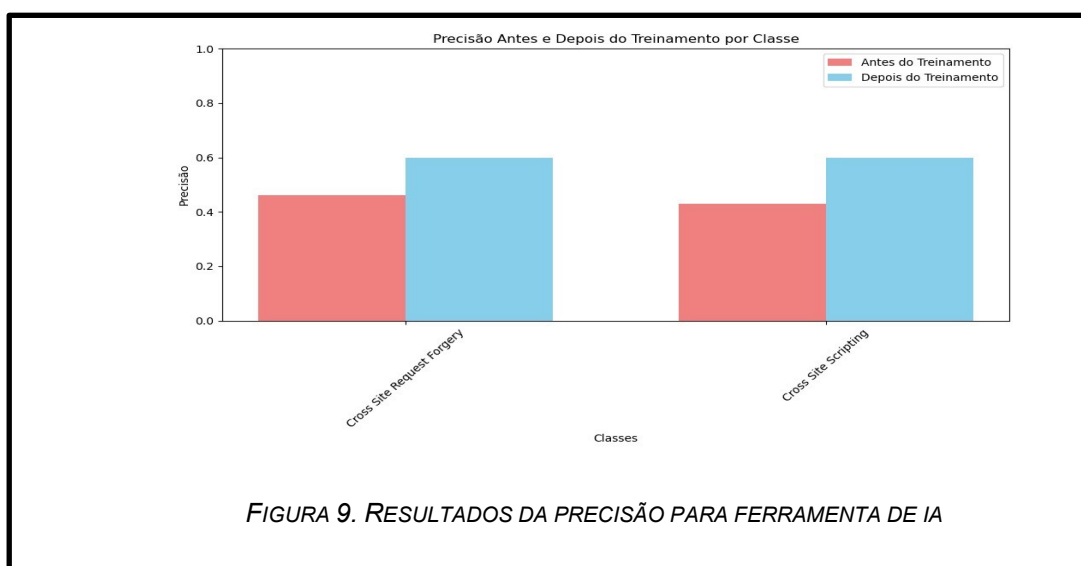
#### 4.3.2 Considerações sobre a ferramenta DeepCode

Avaliar a capacidade de ferramentas baseadas em IA de identificar padrões de código fracos e vulnerabilidades potenciais é crucial hoje em dia, devido ao seu potencial de detectar padrões nunca vistos antes e à resiliência que eles adicionam às entidades que empregam essas tecnologias avançadas. Na tabela 6, o resultado mostra que a ferramenta de IA para a detecção da análise de CSRFs encontrou 17 vulnerabilidades de 20, o que representa perto de 87% das identificadas. Além disso, para análise de XSS (tabela 7), a ferramenta de IA pode atingir 82% de precisão, o que é um bom resultado, sugerindo que perdeu apenas 18% das vulnerabilidades que representam 3 não resolvidas.

#### 4.4 MEDIÇÃO DA FERRAMENTA DE SCANNER COM REDES BAYESIANAS E FERRAMENTAS DE INTELIGÊNCIA ARTIFICIAL

Os relatórios de vulnerabilidades gerados pelo Wapiti foram utilizados como conjunto de dados de entrada para o treinamento de um modelo baseado em redes bayesianas.

Com base nesses dados, o modelo de redes bayesianas foi treinado para identificar padrões de vulnerabilidades, estabelecendo relações probabilísticas entre os fatores de risco e os tipos de ataques detectados. O treinamento permitiu que o modelo aprendesse quais características indicam maior probabilidade de uma determinada vulnerabilidade ocorrer, tornando-o capaz de prever vulnerabilidades em sistemas similares.



O gráfico da figura 9 mostra a precisão da classificação de ataques de CSRF (Cross Site Request Forgery) e XSS (Cross Site Scripting) após o treinamento com redes bayesianas.

Recomendações para a equipe:

- Monitoramento contínuo: Apesar da melhora na precisão, é crucial continuar monitorando o desempenho do modelo, pois novas variantes de ataques podem surgir e comprometer sua eficácia.
- Atualizações regulares: As redes bayesianas precisam ser atualizadas periodicamente com novos dados para manter sua capacidade de detecção. A frequência das atualizações dependerá da dinâmica dos ataques e da quantidade de dados disponíveis.

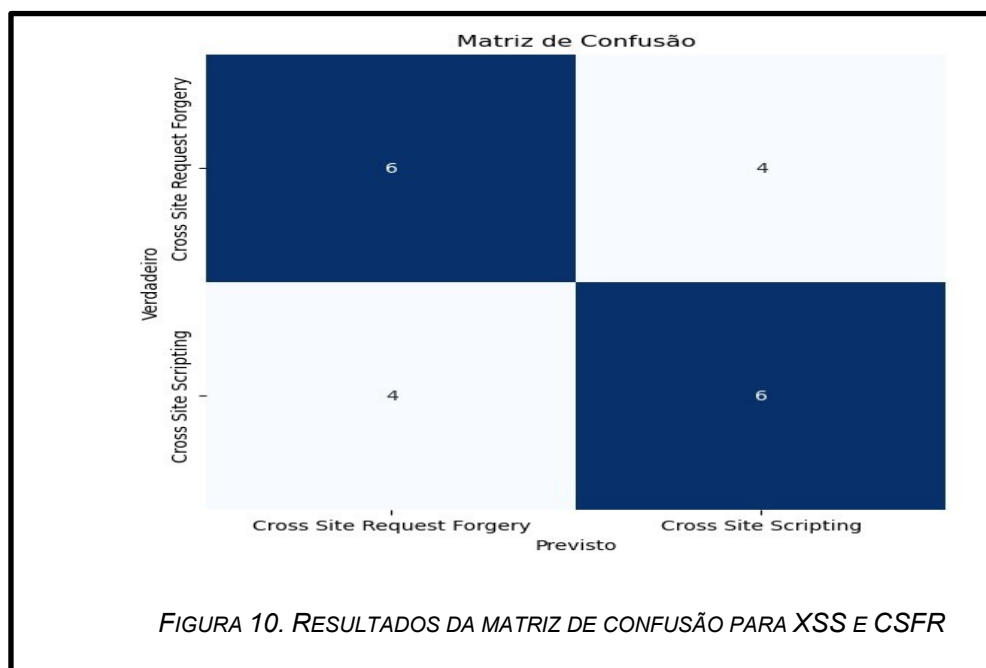
A matriz de confusão da figura 10 mostra os resultados do modelo de classificação binária que distingue entre Cross Site Request Forgery (CSRF) e Cross Site Scripting (XSS). As linhas representam as classes reais (verdadeiras) e as colunas as classes previstas pelo modelo.

Precisão: O modelo tem uma precisão boa para ambos os tipos de ataque. A precisão é calculada para cada classe individualmente (verdadeiros positivos / (verdadeiros positivos + falsos positivos)).

Para CSRF, a precisão seria  $6/(6+4) = 0.6$  ou 60%. Para XSS, também  $6/(6+4) = 0.6$  ou 60%.

Recall: O recall (sensibilidade) também é bom. O recall mede a capacidade do modelo de identificar corretamente todos os exemplos de uma determinada classe. Para CSRF, o recall seria  $6/(6+4) = 0.6$ . Para XSS, também  $6/(6+4) = 0.6$ .

Desempenho geral: O desempenho geral do modelo é bom. A acurácia  $((6+6)/(6+6+4+4))$  é 60%, o que indica que o modelo está correto em 60% das previsões. Considerando que é um problema de classificação binária, a acurácia não é uma métrica muito informativa, e seria melhor usar precisão e recall.



#### 4.5 CONSIDERAÇÕES SOBRE OS RESULTADOS OBTIDOS

Nesta análise, os resultados obtidos das ferramentas de escaneamento automático e das ferramentas de detecção de código fraco de IA são combinados para avaliar sua eficácia conjunta no ambiente de implantação de aplicativos da web seguros. Esta análise é importante porque, ao combinar ambas as abordagens, podemos observar pistas importantes sobre o desempenho futuro da metodologia proposta. Primeiramente, a soma das vulnerabilidades detectadas por ambas as ferramentas fornece uma visão geral mais completa. Se considerarmos que as ferramentas de varredura automatizadas identificaram 46 vulnerabilidades XSS e 19 CSRF e as ferramentas de IA identificaram 42 problemas de segurança XSS e 17 CSRF, poderíamos estimar que há uma sobreposição nas detecções, mas também um potencial para detectar vulnerabilidades que cada ferramenta pode não ter identificado separadamente. Por esse motivo, concluímos parcialmente que a combinação de ferramentas de varredura automática e detecção de código fraco usando IA fornece maior cobertura e eficácia na identificação e resolução de vulnerabilidades.

No final, com uma taxa de resolução combinada com precisão, onde 91% das vulnerabilidades de auto-scan e 77% das detecções automatizadas são resolvidas. Combinado com 82% e 87% de precisão para XSS e CSRF, respectivamente, pode-

se supor que a colaboração de ambas as ferramentas pode levar a uma taxa de resolução ainda maior, especialmente se os problemas mais críticos forem priorizados.

## 5 ESTUDO DE CASO E RESULTADOS

Neste capítulo será realizada a aplicação prática da metodologia proposta para auxiliar o desenvolvimento de aplicações web, visando mitigar vulnerabilidades. Para isso, será utilizado como estudo de caso a aplicação chamada *Damn Vulnerable Web Application* (DVWA), um ambiente amplamente empregado para pesquisa e testes de segurança cibernética. A escolha da DVWA se justifica por seu propósito didático, pois contém vulnerabilidades intencionais que permitem a validação de técnicas de mitigação em um ambiente controlado. O estudo de caso será conduzido seguindo as etapas da metodologia proposta, garantindo a identificação, análise e mitigação de vulnerabilidades. Os testes serão realizados de maneira sistemática, com base em um modelo estruturado de segurança, permitindo avaliar a eficácia das contramedidas aplicadas.

### 5.1 ESCOPO DO ESTUDO DE CASO

A metodologia não será aplicada na aplicação DVWA completa, mas sim em uma ou duas páginas específicas, escolhidas com base em critérios como criticidade das vulnerabilidades e relevância para aplicações web reais. As etapas do estudo incluem:

- Definição do ambiente de testes: configuração da DVWA em um ambiente controlado;
- Seleção das páginas para análise: escolha de uma ou duas páginas com vulnerabilidades relevantes;
- Identificação e análise de vulnerabilidades: levantamento das principais falhas presentes;
- Aplicação das técnicas de mitigação: implementação de medidas corretivas com base na metodologia; e,
- Avaliação dos resultados: análise da eficácia das contramedidas adotadas.

#### 5.1.1 APLICANDO A METODOLOGIA MISTA M-MADAWS NA DVWA

A implementação da Metodologia Mista para Auxiliar o Desenvolvimento de Aplicações Web Seguras (M-MADAWS) na Damn Vulnerable Web Application (DVWA) tem como objetivo validar a eficácia das práticas propostas para mitigar vulnerabilidades em aplicações web.

### 5.1.2 ETAPA 1 - DEFINIÇÃO DO MODELO DE AMEAÇAS

Nesta etapa, foi realizada uma análise detalhada do código-fonte da página login.php do aplicativo DVWA, representado na Figura 12, com o propósito de identificar possíveis vulnerabilidades que possam comprometer a integridade, confidencialidade e disponibilidade do sistema. A sequência das atividades executadas encontra-se representada no diagrama desta etapa (Figura 11).

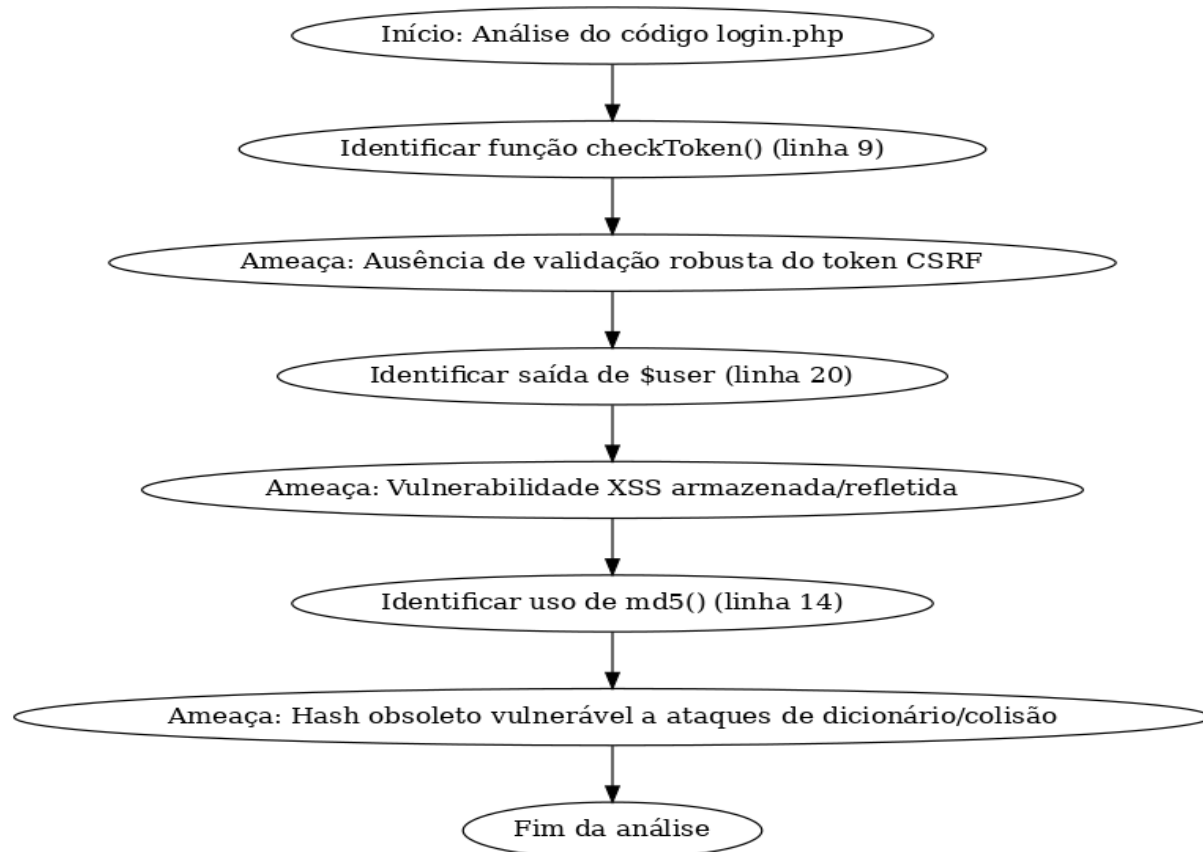


*Figura 11 – Diagrama de identificação de ameaças no código login.php da DVWA*

```

01 <?php
02 define('DVWA_WEB_PAGE_TO_ROOT', '');
03 require_once DVWA_WEB_PAGE_TO_ROOT . 'dvwa/includes/dvwaPage.inc.php';
04
05 dvwaPageStartup([]);
06 dvwaDatabaseConnect();
07
08 if (isset($_POST['Login'])) {
09     checkToken($_REQUEST['user_token'], $_SESSION['session_token'] ?? '', 'login.php');
10
11     $user = stripslashes($_POST['username']);
12     $user = mysqli_real_escape_string($GLOBALS['__mysqli_ston'], $user);
13
14     $pass = md5(stripslashes($_POST['password']));
15
16     $query = "SELECT * FROM `users` WHERE user='$user' AND password='$pass'";
17     $result = mysqli_query($GLOBALS['__mysqli_ston'], $query);
18
19     if ($result && mysqli_num_rows($result) == 1) {
20         dvwaMessagePush("You have logged in as '{$user}'");
21         dvwaLogin($user);
22         dvwaRedirect(DVWA_WEB_PAGE_TO_ROOT . 'index.php');
23     }
24
25     dvwaMessagePush('Login failed');
26     dvwaRedirect('login.php');
27 }
28
29 generateSessionToken();
30 ?>
28
29 generateSessionToken(); // Geração de token de sessão, mas precisa reforço de validade e
rotação
30 ?>

```

*FIGURA 12. CÓDIGO PHP DA PÁGINA LOGIN.PHP DA DVWA*

### 5.1.3 ETAPA 2 - DEFINIÇÃO DE REQUISITOS DE SEGURANÇA DO AMBIENTE E IMPLANTAÇÃO NO SERVIDOR

A partir das ameaças identificadas, são estabelecidos os requisitos de segurança e aplicadas as configurações necessárias no servidor, garantindo uma infraestrutura inicial protegida como mostra a figura 13.

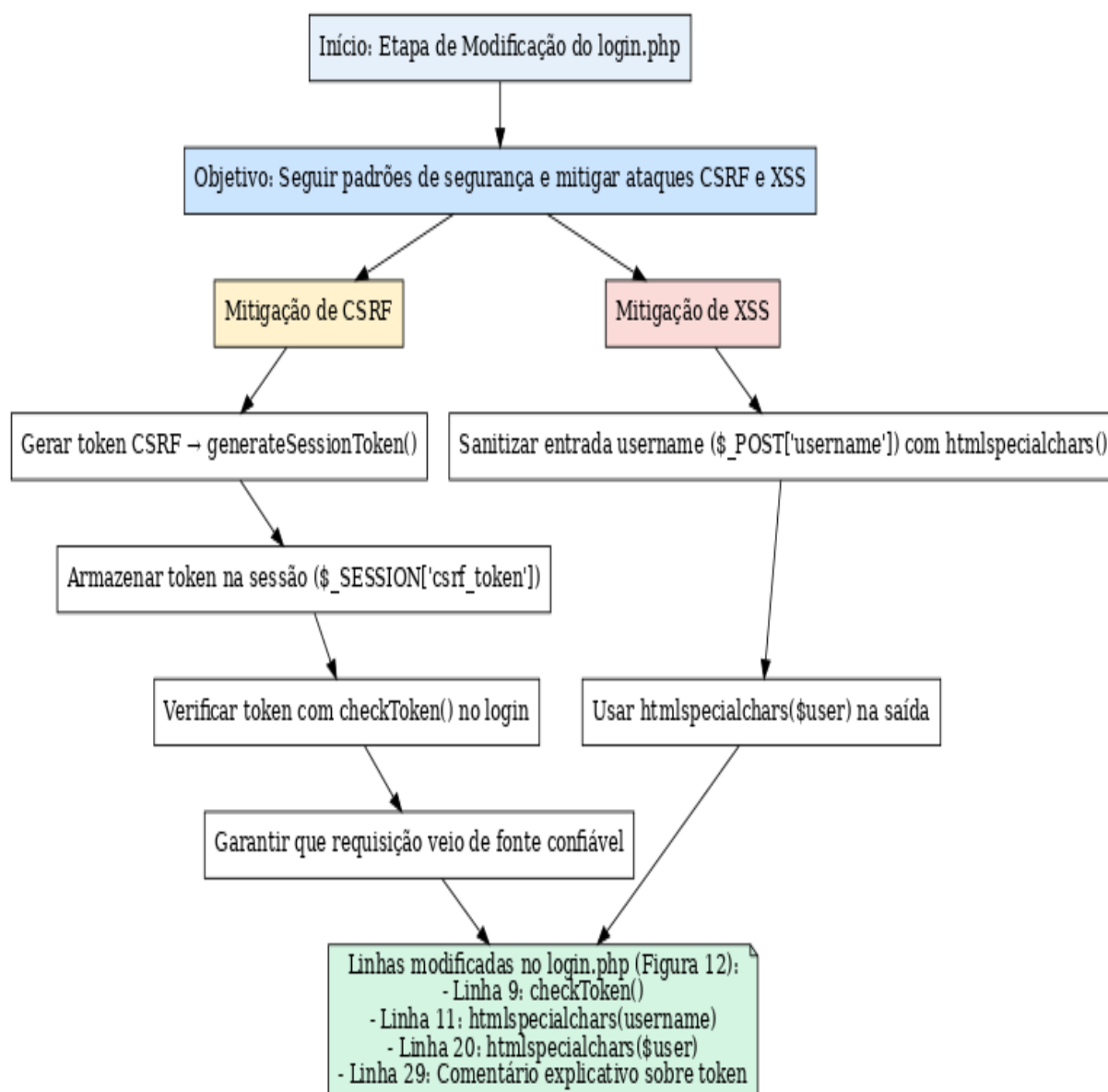
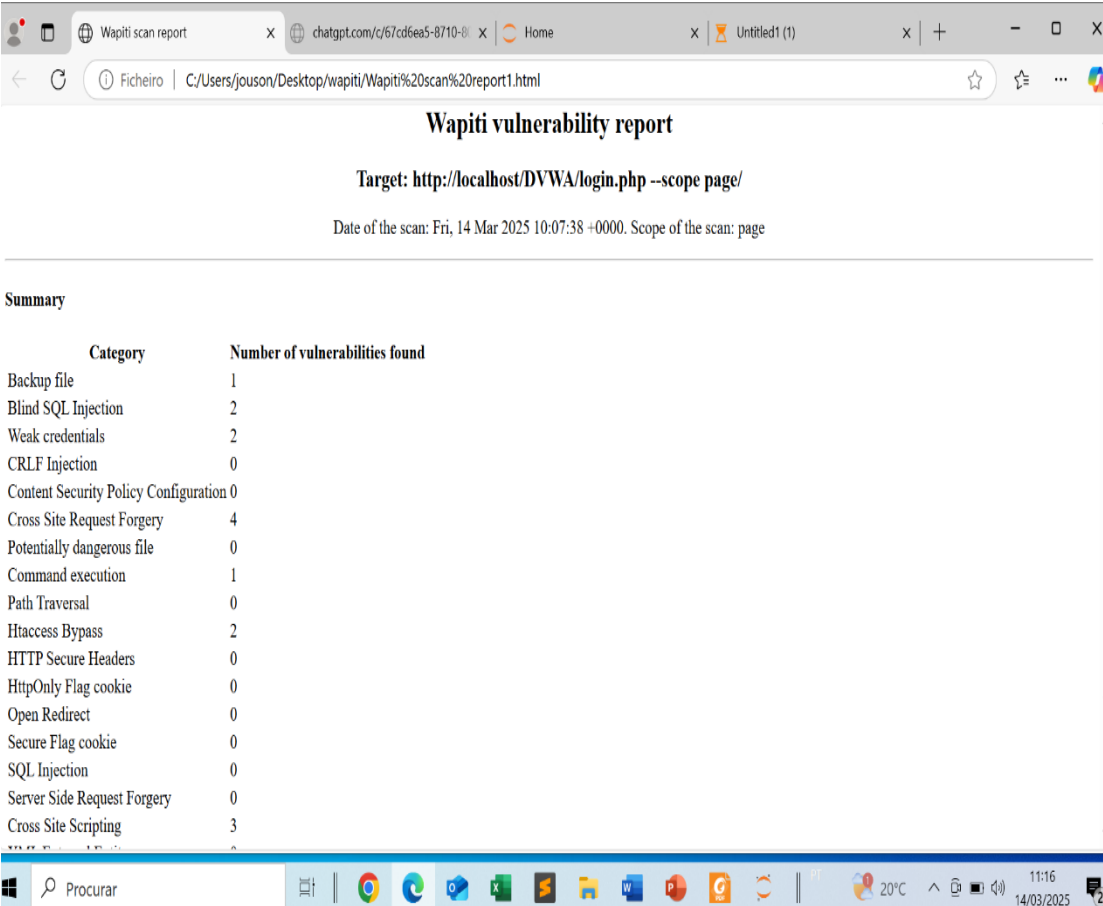


FIGURA 13. PADRÕES DE SEGURANÇA PARA MITIGAR ATAQUES CSRF E XSS

#### 5.1.4 ETAPA 3 - VARREDURA INICIAL COM A FERRAMENTA WAPITI

Após a inserção de técnicas para reduzir as vulnerabilidades, a aplicação é implantada no servidor para a realização de testes com a ferramenta Wapiti como mostra a figura 14. O comando “wapiti -u http://localhost/DVWA/login.php --scope page” foi executado e o resultado é mostrado na figura 14.



| Category                              | Number of vulnerabilities found |
|---------------------------------------|---------------------------------|
| Backup file                           | 1                               |
| Blind SQL Injection                   | 2                               |
| Weak credentials                      | 2                               |
| CRLF Injection                        | 0                               |
| Content Security Policy Configuration | 0                               |
| Cross Site Request Forgery            | 4                               |
| Potentially dangerous file            | 0                               |
| Command execution                     | 1                               |
| Path Traversal                        | 0                               |
| Htaccess Bypass                       | 2                               |
| HTTP Secure Headers                   | 0                               |
| HttpOnly Flag cookie                  | 0                               |
| Open Redirect                         | 0                               |
| Secure Flag cookie                    | 0                               |
| SQL Injection                         | 0                               |
| Server Side Request Forgery           | 0                               |
| Cross Site Scripting                  | 3                               |

FIGURA 14. RESULTADO DA VARREDURA COM A WAPITI

#### 5.1.5 ETAPA 4 - PLANO DE TESTES E ANÁLISE DE VULNERABILIDADES

Após a implantação, a aplicação é submetida a testes utilizando Wapiti para identificar possíveis vulnerabilidades.

É criado um plano de testes na tabela 8, para avaliar a segurança do sistema.

| Fase   | Atividade                                                   | Responsável               | Data Prevista | Status         |
|--------|-------------------------------------------------------------|---------------------------|---------------|----------------|
| Fase 1 | Configuração do ambiente de testes (servidor + ferramentas) | Equipe de Segurança       | 03/03/2025    | (Concluído)    |
| Fase 2 | Execução da varredura automática com Wapiti                 | Analista de Segurança     | 03/03/2025    | (Concluído)    |
| Fase 3 | Testes manuais para XSS e CSRF                              | Equipe de Testes          | DD/MM/AAAA    | Executado      |
| Fase 4 | Análise dos resultados e elaboração de relatório            | Responsável pelo projeto  | DD/MM/AAAA    | Executado      |
| Fase 5 | Aplicação das correções nas vulnerabilidades encontradas    | Equipe de Desenvolvimento | DD/MM/AAAA    | Executado      |
| Fase 6 | Reteste para validar correções                              | Equipe de Testes          | DD/MM/AAAA    | (Não iniciado) |
| Fase 7 | Aprovação final e monitoramento contínuo                    | Gestor de Segurança       | DD/MM/AAAA    | (Não iniciado) |

Tabela 8. Plano de Testes e Análise de Vulnerabilidades

### 5.1.6 ETAPA 5 - CONSULTA AO OWASP TOP 10

Após a identificação das vulnerabilidades durante a fase de testes, realiza-se uma análise comparativa com a OWASP Top 10 na tabela 9, que lista as ameaças mais críticas à segurança de aplicações web. Esse processo permite entender a relevância e o impacto das falhas encontradas, além de fornecer diretrizes para mitigar essas vulnerabilidades seguindo as melhores práticas recomendadas.

| Vulnerabilidade Encontrada        | Categoria OWASP Top 10 (2021)   | Descrição da Ameaça                                                 | Risco | Recomendação                                                                 |
|-----------------------------------|---------------------------------|---------------------------------------------------------------------|-------|------------------------------------------------------------------------------|
| Cross-Site Scripting (XSS)        | A03:2021 - Injeção              | Permite a execução de scripts maliciosos no navegador do usuário.   | Alto  | Implementar escape adequado nas saídas e usar Content Security Policy (CSP). |
| Cross-Site Request Forgery (CSRF) | Não está no OWASP Top 10 (2021) | Um atacante pode forjar requisições em nome do usuário autenticado. | Alto  | Implementar tokens CSRF e validar a origem das requisições.                  |

Tabela 9. Consulta ao OWASP Top 10

### 5.1.7 ETAPA 6 - ANÁLISE DE CÓDIGO-FONTE PARA DETECÇÃO DE VULNERABILIDADES COM DEEPCODE

A varredura com a ferramenta DeepCode consistiu na análise automatizada do código-fonte para identificar vulnerabilidades de segurança e possíveis falhas na aplicação (figura 15).

Vulnerabilidades XSS - Poe

+

poe.com/chat/38rzghmzege4jb3rgo

Gmail

YouTube

Maps

Todos os favoritos

Vulnerabilidades XSS

Assistant

Hoje

Relatório de vulnerabilidades

Support no suporte XSS - Negozio Fantasy

Summary

Resumo

Resumo

Resumo

Recomendações

Recomendações

Compartilhar

Recarregar

Aprimorar

@TopazLabs

→

Mensagem

+

→

html2

Publishar

Publishar

# Relatório de Vulnerabilidades

## Resumo

O seguinte relatório descreve as vulnerabilidades encontradas na aplicação, incluindo Cross-Site Scripting (XSS) e Cross-Site Request Forgery (CSRF).

## Vulnerabilidades Encontradas

| Tipo | Quantidade | Descrição                                                    | Recomendação                                                                                                                                                                                                                                                                                                                                                |
|------|------------|--------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XSS  | 3          | Vulnerabilidades de injeção de scripts em campos de entrada. | <ol style="list-style-type: none"><li>Implementar sanitização e codificação de dados:<pre>htmlspecialchars(\$input, ENT_QUOTES, 'UTF-8');</pre></li><li>Usar políticas de segurança de conteúdo (CSP):<pre>header("Content-Security-Policy: default-src 'self'; script-src 'none';");</pre></li></ol>                                                       |
| CSRF | 2          | Formulários sem proteção contra CSRF.                        | <ol style="list-style-type: none"><li>Adicionar tokens CSRF em todos os formulários:<pre>&lt;input type="hidden" name="csrf_token" value="&lt;?php echo \$_SESSION['csrf_token']; ?&gt;"&gt;</pre></li><li>Verificar o token CSRF no servidor:<pre>if (\$_POST['csrf_token'] !== \$_SESSION['csrf_token']) { die("CSRF token inválido."); }</pre></li></ol> |

FIGURA 15. RESULTADO DA VARREDURA COM DEEPCODE

### 5.1.8 ETAPA 7 - DESENVOLVIMENTO OU CORREÇÃO DO CÓDIGO DA APLICAÇÃO

A aplicação é desenvolvida ou ajustada para eliminar as vulnerabilidades detectadas, seguindo padrões de codificação segura (figura 16).

```
1 <?php
2 session_start();
3 define('DVWA_WEB_PAGE_TO_ROOT', '');
4 require_once DVWA_WEB_PAGE_TO_ROOT . 'dvwa/includes/dvwaPage.inc.php';
5 dvwaPageStartup(array());
6 dvwaDatabaseConnect();
7
8 if (!isset($_SESSION['csrf_token'])) {
9     $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
10 }
11
12 if (isset($_POST['Login'])) {
13     if (!isset($_POST['user_token']) || $_POST['user_token'] !== $_SESSION['csrf_token']) {
14         die("Erro: Token CSRF inválido.");
15     }
16
17     $user = htmlspecialchars($_POST['username'], ENT_QUOTES, 'UTF-8');
18     $pass = $_POST['password'];
19
20     $query = "SELECT * FROM users WHERE user = '$user'";
21     $result = mysqli_query($GLOBALS["___mysqli_ston"], $query);
22
23     if ($result->num_rows === 1) {
24         $row = $result->fetch_assoc();
25         if (password_verify($pass, $row['password'])) {
26             $_SESSION['user'] = $user;
27             dvwaMessagePush("Login bem-sucedido como '{$user}'");
28             dvwaRedirect(DVWA_WEB_PAGE_TO_ROOT . 'index.php');
29         } else {
30             dvwaMessagePush('Falha no login.');
31             dvwaRedirect('login.php');
32         }
33     } else {
34         dvwaMessagePush('Usuário não encontrado.');
35         dvwaRedirect('login.php');
36     }
37 }
38
39 $csrf_token = $_SESSION['csrf_token'];
40
?>
```

FIGURA 16. CÓDIGO PHP MELHORADO DA PAGINA LOGIN.PHP DA DVWA

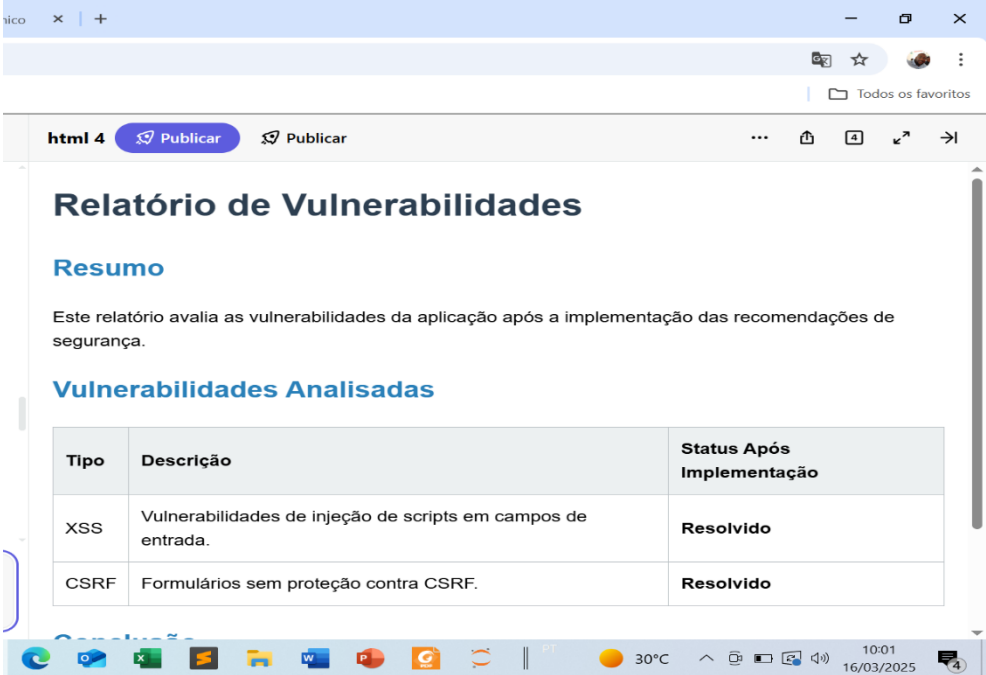
As principais melhorias aplicadas no código de login PHP/HTML da DVWA com linhas alteradas são apresentadas na tabela 10. As melhorias tem o foco na mitigação das vulnerabilidades CSRF e XSS. Cada item destaca a condição original insegura, a correção implementada e as linhas específicas que sofreram alterações.

Tabela 10. Comparação com linhas alteradas do Código HTML/PHP

### Tabela Comparativa com Linhas Alteradas no Código PHP/HTML

| Nº | Item                        | Antes (Inseguro)                                                           | Depois (Melhorado)                          | Linhas Alteradas |
|----|-----------------------------|----------------------------------------------------------------------------|---------------------------------------------|------------------|
| 1  | Token CSRF                  | Versificação básica<br>checkToken()                                        | Geracao com<br>random_bytes()               | PHP: 8-10-13-14  |
| 2  | Entrada do Usuário          | Uso de stripslashes() e mysqli_real_escape_string()                        | Uso password_verify() para verificar o suco | PHP: 17          |
| 3  | Senha não protegida         | Senha obtida diretamente do \$_POST                                        | Uso de mysqli_prepare() e bind_param()      | PHP: 18          |
| 4  | SQL Injection               | Construção montada diretamente com víveli                                  | <input type='text' name='password'> </>     | PHP: 20-22-26-27 |
| 5  | Campo de Senha no HTML      | <input type='text' name='password'>(s)                                     | <input type='password' name='password'>     | HTML 56          |
| 6  | Token refletido no HTML     | Token refletido com <= tokenField() ?>                                     | Inclusão do atributo required               | HTML 57          |
| 7  | Boas práticas no formulário | Escapado diretamente com username, password, sem validação do lado cliente | Inclusão do atributo                        | HTML 54, 56, 62  |

Após a inserção de técnicas para reduzir as vulnerabilidades com base nas recomendações da verificação realizada com a ferramenta DeepCode, são aplicadas correções no código-fonte para eliminar falhas de segurança. As sugestões incluem a adoção de boas práticas de programação, ajustes em trechos de código vulneráveis e a implementação de mecanismos de proteção adicionais, garantindo maior robustez à aplicação.



| Relatório de Vulnerabilidades                                                                               |                                                              |                           |
|-------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------|---------------------------|
| <b>Resumo</b>                                                                                               |                                                              |                           |
| Este relatório avalia as vulnerabilidades da aplicação após a implementação das recomendações de segurança. |                                                              |                           |
| <b>Vulnerabilidades Analisadas</b>                                                                          |                                                              |                           |
| Tipo                                                                                                        | Descrição                                                    | Status Após Implementação |
| XSS                                                                                                         | Vulnerabilidades de injeção de scripts em campos de entrada. | Resolvido                 |
| CSRF                                                                                                        | Formulários sem proteção contra CSRF.                        | Resolvido                 |

*FIGURA 17. RESULTADO DA VARREDURA COM A WAPITI*

### 5.1.9 ETAPA 8 - IMPLANTAÇÃO CI/CD E CONFIGURAÇÃO DO SERVIDOR

Nesta etapa, o código da aplicação é integrado e preparado para implantação utilizando pipelines de Integração Contínua (CI) e Entrega Contínua (CD) como mostra a figura 18. O objetivo é garantir que o código seja testado automaticamente e implantado de forma segura no servidor, minimizando erros e agilizando o processo de desenvolvimento.

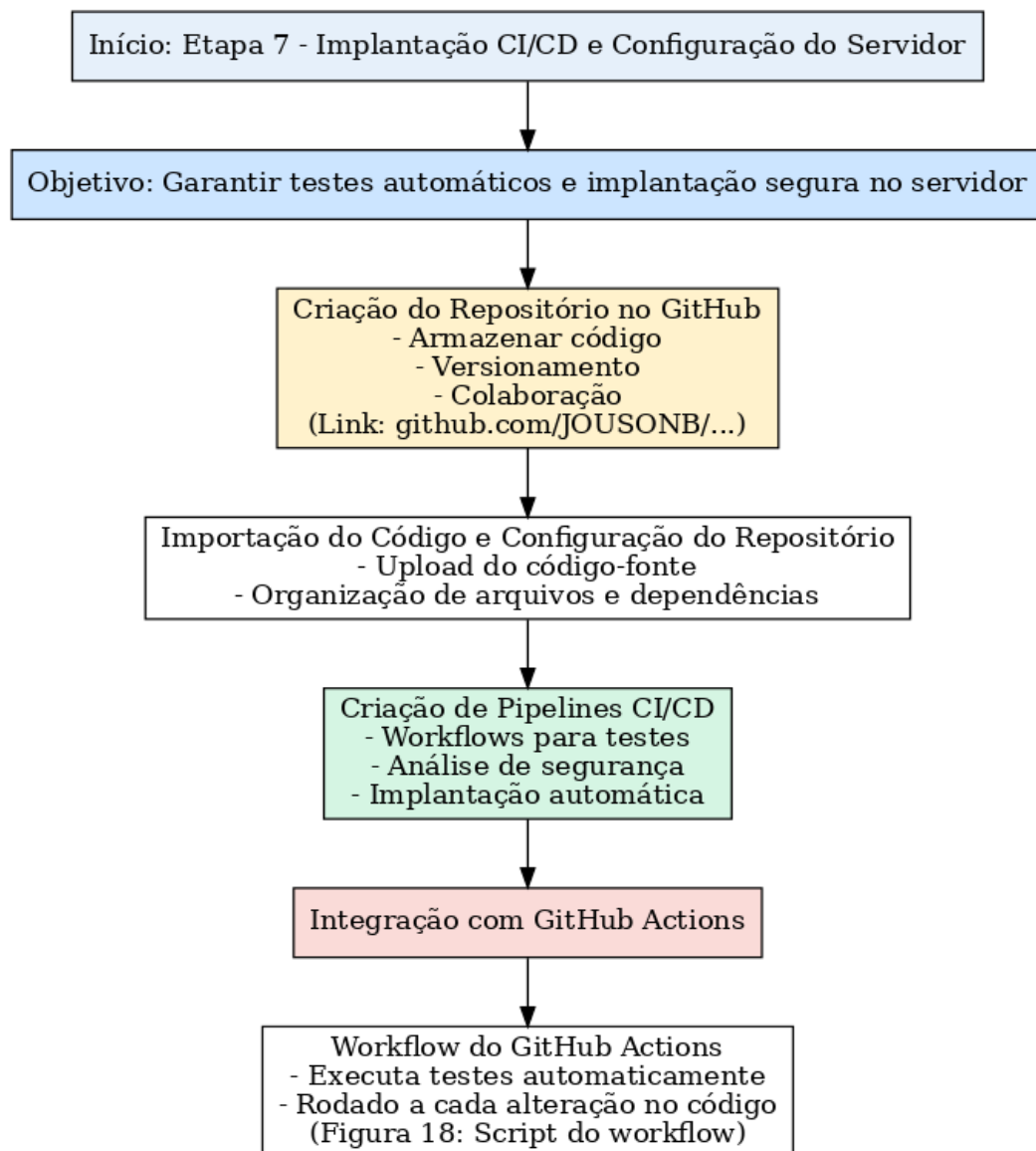


FIGURA 18 INTEGRAÇÃO COM CI/CD (GITHUB ACTIONS)

Para que essa análise seja feita sempre que houver uma nova alteração no código, adicionamos um workflow do GitHub Actions que executa esses testes automaticamente. A figura 19 apresenta o script do workflow.

```
1| name: Security Scan
2| on:
3|   push:
4|     branches:
5|       - main
6|   schedule:
7|     - cron: '0 2 * * *' Executa diariamente às 02:00 AM
8|
9| jobs:
10|   security_test:
11|     runs-on: ubuntu-latest
12|     steps:
13|       - name: Checkout do Código
14|         uses: actions/checkout@v3
15|
16|       - name: Instalar Dependências
17|         run:
18|           sudo apt-get update
19|           sudo apt-get install -y wapiti
20|
21|       - name: Executar Wapiti
22|         run: wapiti -u http://localhost:8000/login.php -m xss,csrf -o security_report.json
23|
24|       - name: Analisar Resultados
25|         run: python analisar_vulnerabilidades.py
```

*FIGURA 19. SCRIPT DO WORKFLOW GITHUB ACTIONS (MAIN.YML)*

#### 5.1.10 ETAPA 9 - MONITORAMENTO CONTÍNUO

Na fase de **IA Precision** do modelo M-MADAWS, criamos um sistema que automatiza a análise de vulnerabilidades, extraíndo dados do relatório do Wapiti e classificando as falhas detectadas (XSS e CSRF). Essa abordagem simula o uso de Inteligência Artificial, pois permite interpretar os resultados e atribuir níveis de risco automaticamente, sem intervenção manual.

Na fase **Scanning VRR** do modelo M-MADAWS, aplicamos um método de pontuação VRR (Vulnerability Risk Rating), avaliando as vulnerabilidades com base em impacto, explorabilidade e probabilidade. Isso permite detectar e classificar

ameaças usando métricas predefinidas, ajudando na priorização das correções de segurança.

## 5.2 RESULTADOS DOS EXPERIMENTOS

A tabela 11 apresenta o resultado da varredura de CSRF: o ensaio 1 detectou 4 CSRF com a wapiti e 2 CSRF com a ferramenta deepcode.

| Ensaio | Estimadas | Detectadas | Resolvidas | VDR | VRR | IA Tools | Precisão |
|--------|-----------|------------|------------|-----|-----|----------|----------|
| 1      | 4         | 4          | 4          | 1   | 1   | 2        | 1,00     |

Tabela 11. Resultado da varredura CSRF

A tabela 12 apresenta o resultado da varredura de XSS: o ensaio 1 detectou 3 XSS com a wapiti e 3 XSS com a ferramenta deepcode.

| Ensaio | Estimadas | Detectadas | Resolvidas | VDR | VRR | IA Tools) | Precisão |
|--------|-----------|------------|------------|-----|-----|-----------|----------|
| 1      | 3         | 3          | 3          | 1   | 1   | 3         | 1,00     |

Tabela 12. Resultado da varredura XSS

## 6 CONCLUSÕES

A implantação segura de aplicativos da web é um processo complexo que requer uma combinação de ferramentas automatizadas e abordagens adicionais como inteligência artificial. Esta dissertação destaca a necessidade de adotar práticas de segurança robustas e se manter atualizado com as tendências emergentes em segurança cibernética para proteger efetivamente os aplicativos da web. Implementar estratégias de segurança desde os estágios iniciais do desenvolvimento é essencial para mitigar os riscos associados a vulnerabilidades em aplicativos da web.

A pesquisa realizada demonstrou a relevância e a eficácia da metodologia **M-MADAWS**(Metodologia Mista para Auxiliar o Desenvolvimento de Aplicações Web Seguras) como uma proposta para auxiliar o desenvolvimento de aplicações web. A metodologia integra técnicas tradicionais de varredura automática com soluções baseadas em inteligência artificial, aumentando o nível de precisão e cobertura na identificação de vulnerabilidades como XSS e CSRF.

Os experimentos práticos realizados com a aplicação DVWA comprovaram a eficiência da abordagem híbrida, demonstrando alta taxa de detecção e elevado nível de precisão. Além disso, a combinação entre métricas como precisão e taxa de detecção revelou-se útil para a avaliação comparativa entre diferentes ferramentas de segurança.

Apesar dos resultados positivos, observou-se a necessidade de ajustes finos nas configurações das ferramentas e a importância do treinamento contínuo de desenvolvedores e administradores, a fim de reduzir falsos positivos e lidar com vulnerabilidades não detectadas. Portanto, conclui-se que a segurança eficaz depende não apenas da tecnologia, mas também da capacitação humana e do monitoramento contínuo.

### 6.1 CONSIDERAÇÕES SOBRE OS OBJETIVOS DA DISSERTAÇÃO

Com base nos resultados obtidos e nas análises realizadas ao longo da pesquisa, podemos afirmar que os objetivos propostos foram plenamente alcançados. O objetivo principal, que era propor uma metodologia segura para a implantação de aplicações web, com foco na mitigação das vulnerabilidades XSS e CSRF, foi concretizado por meio do desenvolvimento e validação prática da metodologia **M-MADAWS**. Os experimentos conduzidos na DVWA demonstraram a eficácia da

abordagem híbrida, integrando ferramentas tradicionais e soluções de inteligência artificial, evidenciando melhorias significativas na taxa de detecção e precisão das análises. Além disso, os objetivos específicos, como a análise de ferramentas de varredura, a aplicação prática da metodologia e a avaliação da eficácia da proposta, foram cumpridos com base em evidências empíricas e métricas técnicas. Dessa forma, a pesquisa não apenas atendeu às expectativas iniciais, como também abriu caminhos para futuras investigações na área da segurança cibernética em ambientes web.

A análise comparativa entre a proposta metodologia **M-MADAWS** e os trabalhos relacionados evidencia as principais contribuições e vantagens do método desenvolvido nesta dissertação. Diferentemente das abordagens anteriores, que frequentemente se limitam a tratar aspectos isolados da segurança em aplicações web, a metodologia M-MADAWS propõe uma metodologia abrangente que contempla todas as etapas fundamentais do processo de proteção. A proposta integra de forma estruturada a identificação de ataques, o uso e a combinação de ferramentas, estratégias de mitigação, e testes práticos, além de incorporar o uso de redes, métricas de avaliação e técnicas baseadas em inteligência artificial (IA). Esse conjunto de características, ausente de forma integrada nos demais estudos analisados, demonstra a contribuição científica da metodologia proposta, posicionando-a como uma contribuição relevante e promissora para o desenvolvimento seguro de aplicações web em cenários reais.

Foi publicado um artigo no evento CISTI 2025 - 20th Iberian Conference on Information Systems and Technologies, que foi apresentado em Junho de 2025 (José e Westphall, 2025). Este evento é classificado no Qualis CAPES como evento B1, cumprindo o requisito necessário para a defesa de dissertação de mestrado no PPGCC.

## 6.2 TRABALHOS FUTUROS

A partir das limitações observadas e das tendências atuais da área, alguns caminhos promissores para trabalhos futuros incluem:

- Aprimoramento das ferramentas de IA: Investigar o uso de modelos de IA mais avançados (como aprendizado profundo) e estratégias de atualização dinâmica

com dados recentes de ciberataques para aumentar a adaptabilidade das soluções propostas.

- Estudos Comparativos Ampliados: Conduzir estudos comparativos com outras metodologias emergentes de segurança, a fim de avaliar o desempenho da metodologia M-MADAWS frente a novos padrões e benchmarks do setor.
- Automatização de Correções: Pesquisar técnicas de auto-fix para vulnerabilidades detectadas, com geração automática de patches seguros e verificados antes da implantação.

Essas direções apontam para uma continuidade promissora da pesquisa, podendo fortalecer ainda mais o campo da segurança cibernética aplicada ao desenvolvimento web.

## REFERÊNCIAS

ANDERSON, R. Security Engineering: A Guide to Building Dependable Distributed Systems. 2. ed. Indianapolis: Wiley, 2008.

ABULGHAFAR, K.; ELMRABIT, N.; YOUSEFI, M. Enhancing Web Application Security through automated penetration testing with multiple vulnerability scanners. *Computers*, v. 12, n. 11, p. 235, 2023. DOI: 10.3390/computers12110235.

ALAZMI, S.; LEON, D. C. D. A systematic literature review on the characteristics and effectiveness of web application vulnerability scanners. 2022.

ALBAHAR, M.; ALANSARI, D.; JURCUT, A. An empirical comparison of pen-testing tools for detecting web app vulnerabilities. *Electronics*, v. 11, p. 2991, 2022. DOI: 10.3390/electronics11112991.

ALSAEDI, A. Effective and scalable black-box fuzzing approach for modern web applications. *Computer and Information Sciences*, 2022. DOI: 10.1016/j.jksuci.2022.10.006.

ALTHUNAYYAN, M.; SAXENA, N.; LI, S.; GOPE, P. Evaluation of black-box web application security scanners in detecting injection vulnerabilities. *Electronics*, v. 11, p. 2049, 2022. DOI: 10.3390/electronics11132049.

BORENSTEIN, M.; HEDGES, L. V.; HIGGINS, J. P.; ROTHSTEIN, H. R. Introduction to meta-analysis. Hoboken: John Wiley & Sons, 2011.

BROWN, M.; WILSON, L. Web Application Security: A Comparative Study of Vulnerability Scanning Tools. 2022.

CABRAL, N. F. Teste de penetração: utilização da metodologia OWASP em aplicações web. 2022. [Tese/dissertação/unpublished].

CHOWDHARY, A.; JHA, K.; ZHAO, M. Generative adversarial network (GAN)-based autonomous penetration testing for web applications. *Sensors*, v. 23, p. 8014, 2023.

FREDJ, O. B.; CHEIKHROUHO, O.; KRICHEN, M.; HAMAM, H.; DERHAB, A. An OWASP Top Ten driven survey on web application protection methods. In: GARCIA-ALFARO, J. et al. (Eds.). *Risks and Security of Internet and Systems. Lecture Notes in Computer Science*, Cham: Springer, 2021. p. 235–252. DOI: 10.1007/978-3-030-68887-5\_14.

GANDIKOTA, P. S. S. K.; VALLURI, D.; MUNDRU, S. B.; YANALA, G. K.; SUSHAINI, S. Web application security through comprehensive vulnerability assessment. *Procedia Computer Science*, v. 230, p. 168–182, 2023. DOI: 10.1016/j.procs.2023.12.072.

Gupta, C.; SINGH, R. K.; MOHAPATRA, A. K. Securing Web Applications Using Security Patterns. In: MISHRA, D. K. et al. (Eds.). *ICT for Competitive Strategies*. Boca Raton: CRC Press, 2020. p. 485–494. DOI: 10.1201/9781003052098-50.

HOFFMAN, A. Web Application Security: Exploitation and Countermeasures for Modern Web Applications. Sebastopol: O'Reilly Media, 2020.

HUSSAIN SEH, A. et al. Hybrid computational modeling for web application security assessment. *Computers, Materials & Continua*, v. 70, p. 469–489, 2022. DOI: 10.32604/cmc.2022.019593

JAMIL, A.; ASIF, K.; ASHRAF, R.; MEHMOOD, S.; MUSTAFA, G. A comprehensive study of cyber attacks & counter measures for web systems. In: ICFNDS'18 – International Conference on Future Networks and Distributed Systems, Amman: ACM, 2018. p. 1–7. DOI: 10.1145/3231053.3231116

JENSEN, F. V.; NIELSEN, T. D. *Bayesian Networks and Decision Graphs*. 2. ed. New York: Springer, 2007.

JOSÉ, JOUSON BARRETO; WESTPHALL, CARLA MERKLE. Metodologia Mista para Implantar Aplicações Web Seguras: combinando ferramentas tradicionais e de IA. In: CISTI 2025 - 20th Iberian Conference on Information Systems and Technologies. 2025. A ser publicado.

KROLL, J.; D'ORNELLAS, M. C.; FONTOURA, L. M. Desenvolvimento de sistemas de gestão da segurança da informação através da integração das normas ISO/IEC 27001:2006 e ISO/IEC 21827 (SSE-CMM). In: VI Simpósio Brasileiro de Sistemas de Informação (SBSI), Marabá, PA: Sociedade Brasileira de Computação, 2010.

LAKATOS, E. M.; MARCONI, M. de A. Fundamentos de metodologia científica: técnicas de pesquisa, planejamento e execução de pesquisa, amostragens e técnicas de elaboração, análise e interpretação de dados. 6. ed. São Paulo: Atlas, 2008.

LUI, B. S. *Web Application Security: A Beginner's Guide*. 2016.

MÁRQUEZ, J. F. R. et al. Implementation of a security strengthening system to mitigate vulnerabilities in academic web applications. *Journal of Survey in Fisheries Sciences*, v. 10, p. 5447–5456, 2023.

NEAPOLITAN, R. E. *Learning Bayesian Networks*. Upper Saddle River: Prentice Hall, 2004.

OWASP. OWASP Broken Web Applications Project. 2016. Disponível em: <https://sourceforge.net/projects/owaspbwa/>. Acesso em: 18 jun. 2025.

OWASP. OWASP Top 10 – 2020: The Ten Most Critical Web Application Security Risks. 2020. Disponível em: <https://owasp.org/www-project-top-ten/>. Acesso em: 18 jun. 2025.

OWASP FOUNDATION. OWASP Zed Attack Proxy (ZAP). 2023. Disponível em: <https://www.zaproxy.org/>. Acesso em: 20 jun. 2025.

OWASP. Top 10 OWASP. 2021. Disponível em: <https://owasp.org/www-project-top-ten/>. Acesso em: 18 jun. 2025.

PEARL, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco: Morgan Kaufmann, 1988.

PONDE, P. S.; SHIRWAIKAR, S. C.; KHARAT, V. S. Knowledge representation of security design pattern landscape using formal concept analysis. *J. Eng. Sci. Technol.*, v. 12, p. 2513–2527, 2017.

PORTSWIGGER LTD. Burp Suite: Application Security Testing Software. 2023. Disponível em: <https://portswigger.net/burp>. Acesso em: 20 jun. 2025.

RAI, R.; MISHRA, A. Application Security: Current Trends and Emerging Threats. 2022.

RAWHAT, S.; BHATEJA, A. Web Application Security: Emerging Threats and Techniques for Protection. 2020.

SALVA, S.; REGAINIA, L. A catalogue associating security patterns and attack steps to design secure applications. *Journal of Computer Security*, v. 27, p. 49–74, 2019. DOI: 10.3233/JCS-171063

SANTOS, A. Analysis of web vulnerabilities and identification techniques. *Journal of Information Systems Engineering & Management*, v. 3, n. 2, p. 1–14, 2018.

SETHI, S.; ROY, H. Web Application Security: A Study on Security Vulnerabilities and Countermeasures. 2019.

SHIREY, G. d. Glossário de segurança na Internet, versão 2. 2007.

THAQI, R.; VISHI, K.; REXHA, B. Enhancing Burp Suite with machine learning extension for vulnerability assessment of web application. *Journal of Information Security and Applications*, 2022. DOI: 10.1080/19361610.2022.2096387

TAMA, B. A.; NKENYEREYE, L.; ISLAM, S. M. R.; KWAK, K.-S. An enhanced anomaly detection in web traffic using a stack of classifier ensemble. 2020.

VIANA, R. S. Segurança no desenvolvimento de aplicações web com a qualidade dos dados. 2013.

VIERA, B. Web application security: a systematic literature review. *Computers & Security*, v. 88, p. 101650, 2019.

YADAV, S.; SINHA, D.; PALLAVI, A.; KAUSHIK, S. Automated vulnerability assessment and security testing in web applications: a review. 2023.

YADAV, S.; KUMAR, M. S. Web Application Security: Protection from Advanced Persistent Threat. 2018.

Zhang, B.; Li, J.; Ren, J.; Huang, G. Efficiency and effectiveness of web application vulnerability detection approaches: a review. *ACM Computing Surveys*, v. 54, p. 1–35, 2022. DOI: 10.1145/3474553