

Análise de Segurança de uma Plataforma Web de CTF: Um Estudo de Caso com o TreasureHunt

Lucas Andrei Cancelier Marin, Guilherme de Campos, Ricardo de la Rocha Ladeira

Instituto Federal Catarinense – Campus Blumenau – Blumenau/SC – Brasil

{lucas23marin, guilhermedecampos12}@gmail.com,
ricardo.ladeira@ifc.edu.br

Abstract. *This paper evaluates the security of the TreasureHunt CTF generator by investigating web vulnerabilities based on OWASP Top 10 guidelines and flaws in competition logic. Using ZAP and Nessus, no High or Critical risk alerts were identified, but Medium and Low-level alerts were found and confirmed through manual validation. The confirmed vulnerabilities include the absence of CSRF tokens in state-changing requests, the use of JavaScript libraries without integrity checks, and session hijacking through reuse of the PHPSESSID cookie, allowing impersonation of authenticated players without passwords. These flaws could compromise the integrity and fairness of the competition, requiring immediate mitigation.*

Resumo. *Este trabalho avalia a segurança da plataforma geradora de CTFs (Capture the Flags) TreasureHunt, investigando vulnerabilidades web segundo as diretrizes do OWASP Top 10 e falhas na lógica da competição. Utilizando as ferramentas ZAP e Nessus, não foram identificados alertas de risco Alto ou Crítico, mas alertas de níveis Médio e Baixo foram encontrados e confirmados por validação manual. As vulnerabilidades incluem a ausência de token CSRF (Cross-Site Request Forgery) em requisições que alteram estado, o uso de bibliotecas JavaScript sem verificação de integridade e o sequestro de sessão por reuso do cookie PHPSESSID, permitindo personificar jogadores autenticados sem o uso de senhas. Conclui-se que essas falhas podem comprometer a integridade e a justiça da competição, exigindo mitigações imediatas.*

1. Introdução

As competições de CTF fornecem desafios que desenvolvem habilidades técnicas [Burket *et al.* 2015; Hulin *et al.* 2017; Schreuders *et al.* 2017] e são importantes ferramentas para o ensino de Cibersegurança. Porém, essas competições apresentam desafios distintos. Muitas delas utilizam desafios estáticos, o que permite o compartilhamento de respostas (*flags*) entre participantes [Burket *et al.* 2015; Schreuders *et al.* 2017]. Além disso, é crítico garantir que o ambiente da competição esteja livre de vulnerabilidades que possam comprometer sua integridade.

Nesse contexto, este trabalho tem como objetivo avaliar a segurança da plataforma geradora de CTF TreasureHunt, sob duas frentes: a identificação de vulnerabilidades com base nas diretrizes do OWASP Top 10, e a investigação de falhas relacionadas à lógica da competição, buscando encontrar brechas que permitam aos participantes contornar as regras do jogo, como obter acesso indevido às contas ou aos exercícios de outros jogadores e manipular o placar e a pontuação dos usuários.

2. Segurança de Aplicações Web e Plataformas de CTF

Devido ao amplo acesso público, aplicações *web* são alvos frequentes de ataques [Vierino *et al.* 2026]. Nesse contexto, [Sillame *et al.* 2024] definem ativos como os recursos de valor de uma organização, enquanto vulnerabilidades representam falhas de arquitetura, código ou implementação, bem como a ausência de medidas preventivas. Os mesmos autores destacam que ameaças correspondem a qualquer situação que coloque esses ativos em risco de perda, dano ou comprometimento. Para orientar o desenvolvimento seguro, a OWASP publica o OWASP Top 10, documento que reúne as principais categorias de riscos de segurança em aplicações *web* [OWASP 2025].

Uma plataforma de CTF combina requisitos de segurança de uma aplicação *web* convencional com requisitos adicionais ligados à integridade e à confidencialidade da competição, já que equipes podem atacar a própria plataforma em vez dos desafios propostos [Matias *et al.* 2018]. Isso exige que o placar reflita fielmente as respostas corretas e resista à adulteração de registros e a falhas na lógica de pontuação [Matias *et al.* 2018], além de prevenir o compartilhamento de credenciais e *flags* entre equipes. Para contornar o problema do compartilhamento de *flags* e a colaboração não autorizada, competições podem utilizar a geração de *flags* únicas por instância de desafio [Burket *et al.* 2015; Chothia & Novakovic 2015; Ladeira *et al.* 2020].

2.1 Trabalhos Relacionados

[Burket *et al.* 2015] propuseram a geração automática de problemas no PicoCTF, atribuindo *flags* únicas por instância de desafio para detectar a troca de respostas entre equipes. A técnica identifica submissões copiadas por divergência entre a *flag* enviada e a instância atribuída, mas o trabalho trata apenas da integridade da pontuação.

[Matias *et al.* 2018] propuseram o NIZKCTF, uma plataforma de CTF baseada em provas de conhecimento zero, onde cada equipe deve submeter uma prova que concluiu o desafio, pois as *flags* não são armazenadas em um banco de dados. Como as provas são públicas, todos podem verificar se a outra equipe de fato resolveu o desafio.

[Al-Abwah 2025] desenvolveu o 0xZone, uma plataforma de CTF via Docker voltada a ambientes educacionais, que utiliza como base a plataforma de código aberto CTFd (Al-Abwah, comunicação pessoal, 17 de setembro de 2026). O sistema alia a segurança de aplicações a um mecanismo *anti-cheat*, mitigando o compartilhamento de respostas por meio da criação de instâncias de arquivos exclusivas por equipe e realizando monitoramento contra vulnerabilidades *web* e ferramentas automatizadas.

2.2 TreasureHunt

TreasureHunt é um gerador automático de competições de CTF no formato *Jeopardy!*, desenvolvido para o ensino da área [Ladeira *et al.* 2020]. Sua arquitetura conta com o gerador de competição, utilizado pelo organizador para criar as instâncias dos exercícios a partir de um rol de técnicas disponíveis e iniciar a competição, e o sistema *web*, usado pelos jogadores para autenticação, *download* dos problemas e submissão das respostas.

Cada jogador recebe um identificador numérico e uma senha para acessar a plataforma, que contém um arquivo com suas instâncias dos desafios. Todos recebem desafios contendo os mesmos problemas, mas formados por instâncias distintas e *flags*

aleatórias. Quando encontradas, as *flags* devem ser submetidas via sistema (Figura 1). Os jogadores são ranqueados por acertos e pontuação, com desempate pelo horário da última submissão correta [Ladeira *et al.* 2020].

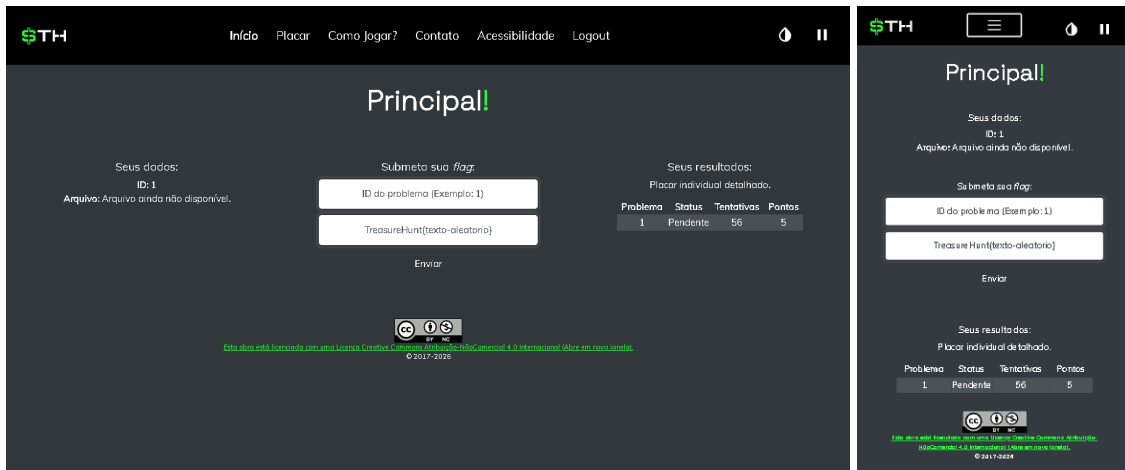


Figura 1. Versões *desktop* e *mobile* da página principal do jogador.

3. Avaliação de Segurança

3.1 Ambiente Experimental e Objeto de Estudo

O TreasureHunt foi instalado em uma máquina virtual com Parrot OS Home 7.3, servidor Apache versão 2.4.68 e Sistema de Gerenciamento de Banco de Dados (SGBD) MariaDB versão 11.8.8. Para o *back-end*, a aplicação utiliza PHP versão 8.4.24. Cabe destacar que o SGBD e o servidor Apache foram mantidos com suas configurações de segurança padrão de instalação.

Os testes foram realizados sobre uma versão de homologação mais recente que a disponível no repositório oficial do projeto no GitHub¹, que contém a documentação da ferramenta em seu README e o código-fonte utilizado como referência para a análise. A versão de homologação incorpora atualizações ainda não publicadas no repositório.

A superfície de ataque da aplicação engloba as páginas de *login*, a página inicial do jogo e as páginas de erros 403 e 404. Assim, os testes manuais se concentraram nos componentes onde as ferramentas automatizadas apontavam vulnerabilidades. Como a configuração do servidor *web* depende do ambiente de hospedagem, os alertas específicos do ambiente não foram considerados falhas da aplicação.

3.2 Procedimento de Avaliação

A avaliação foi dividida em duas etapas. Na primeira, foram utilizadas as ferramentas OWASP ZAP versão 2.17.0 e Tenable Nessus versão 10.12.4. No Nessus, executou-se o *Web Application Scan* completo. Já no ZAP, combinaram-se os *scanners* automatizados padrão com o *Manual Explore*, que identifica vulnerabilidades em segundo plano durante a navegação manual. Posteriormente os alertas foram classificados segundo o OWASP Top 10. Em seguida, a partir dos relatórios das ferramentas, foram realizados

¹<https://github.com/TreasureHuntGame/TreasureHunt>

testes manuais de validação para confirmar se os alertas eram de fato exploráveis. A validação consistiu em verificar a existência das condições reportadas e seus efeitos sobre as propriedades de segurança da aplicação.

Na segunda etapa, as falhas foram exploradas para avaliar se era possível comprometer a dinâmica da competição, manipulando o placar ou acessando arquivos de outros jogadores. Para isso, a segurança da competição foi testada quanto ao controle de acesso aos recursos de outros jogadores, à alteração de identificadores de usuário e à tentativa de alteração dos dados armazenados, das páginas e da pontuação.

3.3 Registro e Classificação dos Achados

Considerou-se vulnerabilidade cada falha técnica confirmada por testes manuais, além de qualquer brecha lógica que violasse as regras do CTF, como a manipulação do placar. O ZAP identifica o CWE (*Common Weakness Enumeration*) dos alertas, permitindo mapeá-los para o OWASP Top 10. Já os alertas do Nessus foram mapeados manualmente, pois ele não disponibiliza relatórios com CWE nas versões gratuitas.

Para a classificação de risco o ZAP utiliza um sistema próprio de classificação, enquanto a ferramenta Nessus adota o CVSS (*Common Vulnerability Scoring System*).

O registro dos achados seguiu abordagens diferentes para cada ferramenta. Os do ZAP foram registrados via exportação automática dos relatórios, reunindo descrição, risco e evidência de cada alerta. Já os do Nessus foram registrados manualmente, pois a edição utilizada não permitia a exportação de relatórios.

4. Resultados e Discussão

As análises do Nessus e do ZAP não identificaram alertas na severidade máxima de cada *scanner* (Crítico e Alto no Nessus, e Alto no ZAP). Contudo, as ferramentas apresentaram achados complementares devido aos seus escopos distintos. O Nessus, voltado à infraestrutura de serviços, concentrou seus alertas nas faixas Baixa e Informativa (Tabela 1), destacando falhas na gestão de credenciais, como a transmissão de senhas em texto claro. Já o ZAP, focado no nível de aplicação *web*, identificou vulnerabilidades nas categorias Média, Baixa e Informativa (Tabela 2), incluindo a ausência de *tokens* anti-CSRF, *cookies* de sessão sem atributos de segurança e o carregamento de bibliotecas JavaScript externas sem o atributo *integrity*.

Quanto às categorias do OWASP Top 10, ambas as ferramentas identificaram falhas de *Security Misconfiguration* (A02). Além disso, o ZAP reportou alertas em *Broken Access Control* (A01) e *Software or Data Integrity Failures* (A08).

Tabela 1. Distribuição dos alertas gerados pela ferramenta Nessus no TreasureHunt.

Nível de Risco dos Alertas Reportados					
Informativo	Baixo	Médio	Alto	Crítico	Total
9	1	0	0	0	10

Tabela 2. Distribuição dos alertas gerados pela ferramenta ZAP no TreasureHunt.

Nível de Risco dos Alertas Reportados				
Informativo	Baixo	Médio	Alto	Total
2	3	4	0	9

Os quantitativos apresentados nas tabelas representam os alertas gerados pelas ferramentas, o que não corresponde necessariamente ao total de vulnerabilidades. A confirmação dos achados foi realizada posteriormente por meio de validação manual.

As vulnerabilidades confirmadas manualmente incluem: (i) a má configuração do *cookie* PHPSESSID, associada à *Security Misconfiguration* (A02), cuja ausência de proteção adequada permitiu o sequestro de sessão e a falsificação de identidade de outros jogadores autenticados sem o conhecimento de suas senhas, caracterizando uma falha de *Authentication Failures* (A07); (ii) a ausência de *token* CSRF em requisições de alteração de estado, permitindo requisições indevidas de terceiros, associada a *Broken Access Control* (A01); e (iii) carregamento de bibliotecas JavaScript de terceiros, como jQuery, sem verificação de integridade por meio de *Subresource Integrity*, falha associada a *Software or Data Integrity Failures* (A08).

Embora falhas como o sequestro de sessão via *cookie* PHPSESSID e a ausência de *tokens* CSRF não sejam exclusivas de plataformas de CTF, mas sim comuns em aplicações *web*, a sua ocorrência em um contexto competitivo afeta a integridade da disputa. Tais vulnerabilidades permitem que um participante visualize desafios de terceiros (confidencialidade comprometida), submeta *flags* em nome alheio (integridade) ou provoque bloqueios temporários (disponibilidade) por tentativas excessivas de força bruta. Do mesmo modo, a ausência de *tokens* CSRF pode gerar requisições não autorizadas de formulários ao *back-end*. O banco de dados do TreasureHunt associa o ID do jogador ao endereço IP utilizado em cada submissão de *flag*. Esses registros podem auxiliar na investigação de possíveis usos indevidos de contas, embora não possam comprovar, isoladamente, a ocorrência de sequestro de sessão.

Com base nas falhas identificadas durante as análises, foram iniciadas as correções na plataforma TreasureHunt, as quais seguem em andamento. Como resultado dessa etapa, o *token* CSRF já foi implementado nos formulários internos. Em relação ao sequestro de sessão, foi desenvolvida uma verificação que vincula o *cookie* PHPSESSID ao endereço IP do usuário, solução que se encontra em fase de testes.

5. Conclusão e Trabalhos Futuros

Este trabalho avaliou a segurança da plataforma TreasureHunt sob a identificação de vulnerabilidades técnicas com base no OWASP Top 10 e a análise de falhas na lógica da competição. Por meio das ferramentas ZAP e Nessus, não foram identificados alertas de riscos máximos, mas a aplicação apresenta fragilidades que requerem mitigação, com destaque para o sequestro de sessão via *cookie*.

Esses achados confirmam que a segunda frente do estudo também foi atendida: as vulnerabilidades técnicas identificadas, se exploradas, podem comprometer a integridade da competição, pois permitem que um participante mal-intencionado influencie negativamente na experiência de outro. Dessa forma, o trabalho evidencia que a segurança de plataformas de CTF exige atenção à camada convencional de aplicações *web* e às particularidades da dinâmica competitiva.

Na comparação entre as ferramentas, o ZAP gerou alertas mais relevantes ao escopo do estudo. O Nessus, por sua vez, concentrou-se em aspectos da infraestrutura de hospedagem, cujos achados foram desconsiderados nos resultados finais.

Como trabalhos futuros, pretende-se continuar a avaliação ofensiva do TreasureHunt por meio de análise de código-fonte e da integração de novas ferramentas de *pentest*, bem como prosseguir com a correção das vulnerabilidades encontradas. Além disso, prevê-se o desenvolvimento de um *script* para monitoramento de falhas em execuções locais e a proposição de uma metodologia sistemática para a avaliação de segurança de plataformas de geração de desafios para CTFs.

Agradecimentos

Os autores declaram que o Gemini 3.8 Flash foi utilizado para revisão gramatical e padronização das referências bibliográficas no padrão adotado pela SBC.

Referências

- Al-Abwah, A. A. T. (2025). 0xZone: A Secure Cybersecurity Challenge Platform. Bachelor's thesis, An-Najah National University.
- Burket, J., Chapman, P., Becker, T., Ganas, C., & Brumley, D. (2015). Automatic problem generation for Capture-the-Flag competitions. USENIX 3GSE '15.
- Chothia, T., & Novakovic, C. (2015). An Offline Capture The Flag-Style Virtual Machine and an Assessment of its Value for Cybersecurity Education. USENIX 3GSE '15.
- Hulin, P., Davis, A., Sridhar, R., Fasano, A., Gallagher, C., Sedlacek, A., Leek, T., & Dolan-Gavitt, B. (2017). AutoCTF: Creating Diverse Pwnables via Automated Bug Injection. USENIX WOOT '17.
- Ladeira, R. R., Obelheiro, R. R., Custódio, R., & Martins, V. (2020). TreasureHunt: um gerador automático de competições de Segurança Computacional. RSC, 9(2).
- Matias, P., Barbosa, P., Cardoso, T. N., Campos, D. M., & Aranha, D. F. (2018). Nizkctf: A noninteractive zero-knowledge capture-the-flag platform. IEEE Security & Privacy, 16(6), 42-51.
- OWASP Foundation. (2025). OWASP Top 10:2025. <https://top10.owasp.org/2025/>
- Schreuders, Z. C., Shaw, T., Shan-A-Khuda, M., Ravichandran, G., Keighley, J., & Ordean, M. (2017). Security Scenario Generator (SecGen): A Framework for Generating Randomly Vulnerable Rich-scenario VMs for Learning Computer Security and Hosting CTF Events. USENIX ASE '17.
- Sllame, A. M., Tomia, T. E. & Rahuma, R. M. (2024). A holistic approach for cyber security vulnerability assessment based on open source tools: Nikto, Acunitx, ZAP, Nessus and enhanced with AI-powered tool "ImmuniWeb".
- Vierino, F. T., Wahanani, H. E., & Junaidi, A. (2026). Evaluating Web Application Security Using OWASP Top 10 and NIST SP 800-115. Bit-Tech, 8(3), 3754-3764.