UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Leonardo Frangello Franzese

Um Estudo das Vulnerabilidades do OWASP Top 10 na plataforma Moodle

Uberlândia, Brasil

2023

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Leonardo Frangello Franzese

Um Estudo das Vulnerabilidades do OWASP Top 10 na plataforma Moodle

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Orientador: Silvio Ereno Quincozes

Coorientador: Vagner Ereno Quincozes

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2023

Leonardo Frangello Franzese

Um Estudo das Vulnerabilidades do OWASP Top 10 na plataforma Moodle

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 19 de junho de 2023:

Silvio Ereno Quincozes
Orientador

Rodrigo Sanches Miani

Ivan da Silva Sendin

Uberlândia, Brasil
2023

Agradecimentos

Ao Prof. Dr. Silvio Ereno Quincozes e ao seu irmão Vagner Ereno Quincozes, pela ajuda, incentivo e paciência durante toda essa jornada.

Aos professores e funcionários da Faculdade de Computação (FACOM) da Universidade Federal de Uberlândia por contribuírem com minha formação.

Aos meus amigos que me apoiaram e me ajudaram durante toda essa trajetória.

Aos meus familiares, especialmente meus pais, irmã e à minha namorada por todo incentivo e apoio durante toda a minha formação.

"Não existe segurança perfeita, apenas graus de incerteza" - Gideon J. Gartner.

Resumo

A plataforma *Moodle* consiste em uma aplicação Web usada por mais de 316 milhões de usuários, os quais confiam suas informações pessoais e acadêmicas às instituições que disponibilizam materiais e aplicam avaliações por meio dessa plataforma. Em um cenário onde a segurança cibernética é uma preocupação em nível mundial, com um aumento de 32% de ataques em 2022, cabe o questionamento: A plataforma *Moodle* está segura? Para responder essa pergunta, este trabalho apresenta um estudo de caso calcado nas principais vulnerabilidades que atingem as aplicações Web atuais, de acordo com a lista "Top 10 OWASP", de 2021. Tal análise, baseada na ferramenta Zed Attack Proxy (ZAP), identificou 894 alertas de potenciais vulnerabilidades para as quais este trabalho (i) conduz uma análise de risco de exploração por atacantes, (ii) apresenta as devidas contramedidas, e (iii) aponta as principais informações que podem ser monitoradas para detecção de ataques cibernéticos.

Palavras-chave: Detecção de Ameaças, *OWASP Top 10*, *Moodle*, Vulnerabilidades, Ataques Cibernéticos, Segurança da Informação.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AR	Arquitetura de Referência
AWS	Amazon Web Services
CD	Continuous Delivery
CI	Continuous Integration
CSV	Comma-Separated Values
CSP	Content Security Policy
CWE	Common Weakness Enumeration
CSRF	Cross-Site Request Forgery
HSTS	HTTP Strict Transport Security
HTML	Hypertext Markup Language
HTTPS	Hypertext Transfer Protocol Secure
HTTP	Hypertext Transfer Protocol
IDOR	Insecure Direct Object Reference
IDS	Intrusion detection system
IP	Internet Protocol
OS	Operating System
OWASP	Open Web Application Security Project
PDF	Portable Document Format
PHP	PHP: Hypertext Preprocessor
RCE	Remote Code Execution
SSL	Secure Sockets Layer
SQL	Structured Query Language
SSRF	Server-Side Request Forgery

TLS	Transport Layer Security
URL	Uniform Resource Locator
VPN	Virtual Private Network
VLE	Virtual Learning Environment
W3AF	Web Application Attack and Audit Framework
XSS	Cross-Site Scripting
XML	eXtensible Markup Language
ZAP	Zed Attack Proxy

Sumário

1 Introdução

O *Moodle*, uma plataforma de ensino de código aberto amplamente utilizada em instituições educacionais ao redor do mundo, é uma escolha popular para a criação e oferta de cursos. Com mais de 316 milhões de usuários em todo o mundo e mais de 41 milhões de cursos disponíveis em 42 idiomas, o *Moodle* oferece uma variedade de recursos e ferramentas para apoiar o processo de aprendizagem (DOUGIAMAS, 2022). Sua natureza de código aberto permite que instituições personalizem e adaptem a plataforma de acordo com suas necessidades específicas, tornando-a flexível e escalável para diferentes contextos educacionais. Além disso, o *Moodle* está em constante evolução, com uma comunidade ativa de desenvolvedores que contribuem com melhorias e novas funcionalidades.

Em um cenário onde a segurança cibernética é uma preocupação em nível mundial, com um aumento de 32% de ataques em 2022 (ABEINFO, 2022), cabe o questionamento: A plataforma *Moodle* está segura? Para garantir a segurança de uma plataforma amplamente utilizada como o *Moodle*, é fundamental adotar medidas e práticas de segurança robustas. Nesse sentido, o Projeto Aberto de Segurança em Aplicações Web, do inglês, *Open Web Application Security Project* (OWASP) (OWASP, 2022a) desempenha um papel crucial. O OWASP é uma comunidade internacional sem fins lucrativos que se dedica a produzir artigos, metodologias, documentações, ferramentas e tecnologias no campo da segurança de aplicativos Web (OWASP, 2022a).

Através do *OWASP Top 10*, uma lista regularmente atualizada das dez principais vulnerabilidades, os desenvolvedores são conscientizados sobre os riscos mais críticos em aplicações Web (OWASP, 2022a). O OWASP *Zed Attack Proxy* (ZAP), uma ferramenta desenvolvida pelo projeto OWASP, é especificamente projetada para realizar a varredura de vulnerabilidades em aplicações Web, com base nas vulnerabilidades identificadas no *OWASP Top 10*. Portanto, ao utilizar essa ferramenta, é possível identificar as principais potenciais vulnerabilidades a fim de mitigar potenciais ameaças cibernéticas e fortalecer a segurança do *Moodle*.

1.1 Justificativa

A segurança cibernética é uma preocupação de nível mundial. Os dados recentes demonstram crescimento na quantidade de ocorrências de ataques cibernéticos. No 2° trimestre de 2022, houve um aumento de 32% nos ataques cibernéticos globais em comparação com o 2° trimestre de 2021, onde ocorreu uma média de 1200 ataques semanais por organização. Já no Brasil, no mesmo período, os ataques aumentaram em 46% comparado ao período do 2° trimestre de 2021 (ABEINFO, 2022). Dentre os alvos desses

ataques, destacam-se aqueles direcionados às aplicações Web. De acordo com o relatório da Kaspersky (KASPERSKY, 2021) durante o ano de 2021, 15,45% dos computadores de usuários da internet global sofreram algum tipo de ataque.

Atualmente, os estudos encontrados na literatura sobre a segurança do *Moodle* não estão alinhados com as principais vulnerabilidades atuais nem abordam contramedidas efetivas para proteger a plataforma. Embora existam esforços direcionados ao estudo da segurança no *Moodle* (CUERVO; ALDANA; LÓPEZ, 2016; HERNANDEZ; CHAVEZ, 2008) (MUDIYANSELAGE; PAN, 2020), esses trabalhos geralmente se baseiam em versões anteriores da plataforma, o que pode ser considerado desatualizado, já que o *Moodle* passa por atualizações constantes. Além disso, a maioria desses estudos não fornecem detalhes sobre as técnicas utilizadas para explorar as vulnerabilidades nem abordam a detecção de ataques que possam explorar essas vulnerabilidades. Também não há uma análise dos *logs* gerados por esses ataques, o que limita a compreensão dos padrões de ataque e a implementação de medidas de segurança adequadas.

Por outro lado, os estudos que consideram a lista do *OWASP Top 10* também têm suas limitações. Embora existam pesquisas que abordam as principais vulnerabilidades listadas pelo OWASP (PRIYAWATI; ROKHMAH; UTOMO, 2022; SAMPAIO, 2021; FREDJ et al., 2020), a maioria desses estudos utiliza versões desatualizadas dessa lista, como as divulgadas em 2017 ou 2013. No entanto, em setembro de 2021, uma versão atualizada foi publicada, com alterações significativas, como o agrupamento de algumas vulnerabilidades e o surgimento de outras. Além disso, esses estudos geralmente não exploram as ferramentas e técnicas práticas utilizadas pelos atacantes para explorar as vulnerabilidades discutidas. Dessa forma, há uma lacuna de estudos atualizados sobre os grupos de vulnerabilidades mais comuns em aplicações Web, incluindo o *Moodle*, e seus impactos nos sistemas atuais. É fundamental preencher essa lacuna para fortalecer a segurança do *Moodle* e proteger os usuários contra ameaças cibernéticas.

1.2 Objetivos

Este trabalho tem como objetivo preencher as lacunas identificadas anteriormente no contexto da segurança do *Moodle*. O foco será a investigação das principais vulnerabilidades encontradas nessa plataforma, com base na versão mais recente do *OWASP Top 10*. Em particular, o trabalho conduzirá uma análise do risco de exploração dessas vulnerabilidades baseada em ferramentas de testes de penetração, identificando as possíveis formas de exploração e os impactos potenciais. Dessa forma, serão propostas as devidas contramedidas para mitigar as vulnerabilidades em potencial e proteger o *Moodle* contra ataques cibernéticos. Por fim, o trabalho apontará as principais informações que podem ser monitoradas visando alertar e prevenir possíveis ataques ao ambiente do *Moodle*. Os

objetivos específicos deste trabalho são pontuadas a seguir:

- Realizar um estudo atualizado das vulnerabilidades do *OWASP Top 10* (versão de 2021) no contexto da plataforma *Moodle*;
- Executar uma varredura de vulnerabilidades na plataforma *Moodle* usando a ferramenta OWASP ZAP e discutir os resultados.
- Apresentar um estudo das causas raízes das vulnerabilidades reportadas pelo OWASP ZAP, analisando suas implicações.
- Recomendar técnicas específicas de mitigação para fortalecer a segurança do *Moodle* e mitigar as vulnerabilidades identificadas.
- Realizar uma análise aprofundada das principais fontes de informações e recursos relevantes para a detecção de ações maliciosas no ambiente do *Moodle*.

Com esses objetivos, pretende-se fornecer informações relevantes e práticas para as instituições que adotam a plataforma *Moodle*, bem como seus desenvolvedores e administradores, a fim de garantir a integridade, confidencialidade e disponibilidade dos sistemas e dados no contexto específico dessa plataforma educacional.

O restante deste trabalho está organizado da seguinte maneira: No Capítulo 2, será apresentado o referencial teórico necessário para o entendimento do trabalho. No Capítulo 3, será detalhado o processo de preparação do ambiente para a realização dos experimentos. No Capítulo 4, serão realizadas três etapas de experimentos e análises. A primeira etapa consistirá na análise das vulnerabilidades em um ambiente em nuvem com uma instalação padrão do *Moodle*. Na segunda etapa, serão indicadas as contramedidas adequadas para as vulnerabilidades identificadas. Por fim, na terceira etapa, serão estudadas as informações que podem ser monitoradas para a detecção de ataques que explorem essas vulnerabilidades. Por fim, no Capítulo 5, serão apresentadas as conclusões obtidas ao longo do trabalho.

2 Referencial Teórico

Neste capítulo serão apresentados os principais conceitos fundamentais, os trabalhos correlatos e as tecnologias aplicadas durante este trabalho.

2.1 Conceitos Fundamentais

Nesta seção, serão abordadas as definições essenciais relacionadas as propriedades da segurança da informação, vulnerabilidades, ferramentas que permitem a sua detecção (*scanners*) e os potenciais ataques que podem ser executados a partir de sua exploração.

2.1.1 Propriedades da Segurança da Informação

A segurança da Informação possui algumas propriedades importantes. Segundo o autor (STALLINGS, 2008) as propriedades são: confidencialidade, integridade, disponibilidade, anti-reprodução, autenticidade, auditoria, tempestividade, entre outras. A seguir será mostrado as definições para a tríade principal da segurança da informação.

2.1.1.1 Confidencialidade

Segundo Fortinet (2023) a confidencialidade é a propriedade que irá envolver esforços para que os dados sejam mantidos em segredo ou privados. Para conseguir isso, é necessário garantir que os dados só sejam acessados por indivíduos, sistemas ou processos autorizados. Ou seja, caso um usuário malicioso consiga acessar as informações sensíveis de um outro usuário, a confidencialidade terá sido quebrada, pois uma pessoa não autorizada obteve acesso ao dado.

2.1.1.2 Integridade

Segundo Fortinet (2023) a integridade é a propriedade que envolve garantir que seus dados sejam confiáveis livres de qualquer adulteração. Então, a integridade só é mantida apenas se os seus dados forem autênticos, precisos e confiáveis. Ou seja, caso um usuário malicioso consiga adulterar algo de um outro usuário (*e.g.* um documento) a propriedade terá sido violada.

2.1.1.3 Disponibilidade

Segundo Fortinet (2023) a disponibilidade é a propriedade que garante que sistemas, redes e aplicativos estejam funcionando como deveriam e quando deveriam. Ou seja,

caso haja uma queda de energia e um sistema caia, a sua propriedade de disponibilidade terá sido violada, pois o sistema não estará disponível.

2.1.2 Vulnerabilidades

Dentro do escopo deste trabalho, que consiste em estudar as vulnerabilidades do *OWASP Top 10* na plataforma *Moodle*, é importante entender o conceito de vulnerabilidades. Segundo o autor Humayun (HUMAYUN et al., 2020), vulnerabilidades são falhas em um sistema ou em seu projeto que permitem a um invasor executar comandos maliciosos, acessar dados de forma não autorizada e/ou realizar diversos tipos de ataques de negação de serviço.

No contexto do *Moodle*, plataforma que está sendo analisada neste estudo, é relevante destacar alguns exemplos de vulnerabilidades, conforme mencionado pelo Pedra (2022). Esses exemplos incluem vulnerabilidades de rede, softwares desatualizados e a ausência de uma política de segurança da informação bem estruturada.

Ao compreender essas definições e exemplos de vulnerabilidades, poderemos explorar e analisar os pontos específicos relacionados ao OWASP Top 10 no ambiente do *Moodle*, fornecendo assim *insights* e recomendações para a proteção efetiva contra essas vulnerabilidades.

2.1.3 Ferramentas de *Scan*

De acordo com o "Guia do Hacker Brasileiro" (ASSUNÇÃO, 2002), os *scanners* de vulnerabilidades são ferramentas poderosas que têm a capacidade de detectar erros em sistemas em questão de segundos, além de fornecer *insights* sobre como explorar esses erros para realizar um ataque cibernético.

Neste trabalho de conclusão de curso, é adotado o *scanner* de vulnerabilidades OWASP ZAP (OWASP, 2022b), uma ferramenta reconhecida e amplamente utilizada na área de segurança da informação.

Além de detectar vulnerabilidades em aplicações *Web*, o OWASP ZAP também fornece informações valiosas sobre como esses erros podem ser explorados para a realização de um ataque cibernético, contribuindo assim para a análise e mitigação de riscos de segurança no ambiente do *Moodle* (MUDIYANSELAGE; PAN, 2020).

2.1.4 Ataques

Para um melhor entendimento dentro do escopo deste trabalho, que se concentra no estudo das vulnerabilidades do OWASP Top 10 na plataforma *Moodle*, é importante contextualizar a relação entre vulnerabilidades e ataques. Conforme mencionado pelo

autor Humayun (HUMAYUN et al., 2020), ataques são ações tomadas para danificar um sistema ou perturbar suas operações rotineiras, explorando vulnerabilidades por meio de várias ferramentas e técnicas.

No contexto específico do *Moodle*, as vulnerabilidades identificadas na lista *OWASP Top 10* podem resultar em diversos tipos de ataques. Um dos objetivos de um atacante ao direcionar ataques ao *Moodle* é obter as credenciais de outros estudantes ou instrutores, a fim de se apropriar de suas contas de usuário. Isso poderia permitir ao atacante violar a confidencialidade, integridade e disponibilidade de dados protegidos dentro do *Moodle*. Por exemplo, um atacante poderia roubar uma tarefa concluída de um estudante-alvo ou excluir uma tarefa enviada para sabotar seu progresso acadêmico. Além disso, os atacantes podem tentar roubar materiais didáticos proprietários ou comprometer credenciais de administradores para fins maliciosos (MUDIYANSELAGE; PAN, 2020).

Dessa forma, é fundamental compreender e mitigar essas vulnerabilidades para garantir a segurança e proteção adequadas do ambiente do *Moodle* utilizado por instituições de ensino. Isso ajudará a evitar prejuízos financeiros e o acesso não autorizado a dados confidenciais que podem comprometer a integridade e a confiabilidade do sistema.

2.2 OWASP Top 10

Nesta seção, serão apresentadas as dez principais categorias de vulnerabilidades, de acordo com a última versão do relatório *OWASP Top 10* (OWASP, 2022a), o qual representa um amplo consenso sobre os riscos de segurança mais críticos para aplicações Web. Tais vulnerabilidades estão numeradas conforme o *ranking* do *OWASP Top 10*. Por exemplo, a primeira delas é a melhor ranqueada (*i.e.,* oferece risco maior) e a última é a pior ranqueada (*i.e.,* oferece risco menor). Ademais, são apresentados exemplos de falhas baseadas no conceito de Enumeração de Fraquezas em Comum, do inglês, *Common Weakness Enumeration* (CWE), que é um sistema feito para categorizar as falhas de segurança de software. Tal sistema considera defeitos de implementação que podem gerar vulnerabilidades (MITRE, 2022).

1. **Broken Access Control:** Esta vulnerabilidade é crítica em sistemas Web em razão do impacto negativo em termos financeiros e de danos a reputação da empresa (HASSAN et al., 2018). Uma das formas mais comuns de exploração dessa vulnerabilidade consiste na modificação de parâmetros da *Uniform Resource Locator* (URL) de modo a conseguir o acesso não autorizado a uma página, isto é, escalação horizontal de privilégios. A escala de privilégio vertical de usuários comuns para a execução de ações de um administrador também é uma variação dessa vulnerabilidade. Outra forma de explorar essa vulnerabilidade pode se dar pela ausência de controle de acesso às páginas com recursos ou funções que deveriam ser de exclusividade de um determi-

nado usuário autenticado. Como exemplo, pode-se citar a redefinição de senha de usuário de terceiros através da modificação do nome de usuário na requisição *Hyper Text Transfer Protocol* (HTTP). Alguns dos CWEs relacionados a essa categoria de vulnerabilidade consistem em *Cross-Site Request Forgery* (CSRF), *Insecure Direct Object Reference* (IDOR), *Open Redirect* e *Path Traversal* (OWASP, 2022a).

2. **Cryptographic Failures:** A vulnerabilidade de Falha Criptográfica, do inglês, Cryptographic Failures, foi incluída na versão de 2017 como *Sensitive Data Exposure*. De acordo com OWASP (OWASP, 2022a), de forma geral, qualquer falha que resulte em uma exposição de dados sensíveis ou críticos de uma pessoa pode ser considerado uma falha criptográfica. Alguns dos CWEs relacionados a essa categoria de vulnerabilidade consistem em *Codificação fraca para a senha* (CWE-261), *Transmissão de texto simples de informações confidenciais* (CWE-319), *Entropia insuficiente* (CWE-331), *Uso de um algoritmo criptográfico fraco ou arriscado* (CWE-327) e *Etapa criptográfica necessária ausente* (CWE-325).

3. **Injection:** De acordo com (DEEPA; THILAGAM, 2016), as vulnerabilidades relacionadas à categoria *Injection* ocorrem quando um invasor é capaz de manipular os valores dos parâmetros de entrada do usuário. Um exemplo bastante comum consiste na manipulação de partes de consultas *Structured Query Language* (SQL) com a finalidade de manipular o banco de dados. Como consequência desse tipo de ataque, pode ocorrer o roubo de dados, perda de dados, comprometimento do banco de dados, negação de serviço, comprometimento do sistema e desfiguração (*defacing*) do site que causa prejuízos financeiros e à imagem da empresa. Os casos mais comuns de injection ocorrem pelas seguintes falhas: *SQL/NoSQL Injection*, *Cross-Site-Script* (XSS), *Hypertext Markup Language (HTML) Injection* e Injeção de comando do sistema operacional, do inglês, *OS Command Injection*.

4. **Insecure Design:** Esta categoria está diretamente relacionada ao fato de os desenvolvedores de uma aplicação não utilizarem (i) um padrão de projeto seguro; (ii) uma modelagem de ameaças; ou (iii) arquiteturas de referência (AR) (CLOUTIER et al., 2010), que tornam menores as possibilidades da aplicação ter falhas de segurança. O *design* inseguro engloba diversos riscos quando os desenvolvedores não se preocupam em seguir as práticas de segurança recomendadas. Normalmente, esse tipo de vulnerabilidade ocorre quando as equipes responsáveis pela aplicação não conseguem prever e/ou avaliar as ameaças durante a fase do *design* de código. Um software estará sujeito a essa categoria de vulnerabilidade quando não respeitar um gerenciamento de requisitos e recursos, um *design* seguro ou ciclo de vida de desenvolvimento seguro. Alguns dos CWE's relacionados a essa categoria são *atribuição de privilégio incorreta* (CWE-266), *gerenciamento de privilégios inade-*

quado (CWE-269), *violação do limite de confiança* (CWE-501), além de *credenciais insuficientemente protegidas* (CWE-522).

5. **Security Misconfiguration:** Em contraste com as categorias anteriores, esta categoria compreende vulnerabilidades introduzidas baseadas em erros de configuração da aplicação. Dentre os erros de configuração mais comuns estão cabeçalhos HTTP mal configurados, contas com *login* e senha padrão, mensagens de erros que fornecem algum tipo de informação sensível sobre o sistema ou possuir recursos desnecessários ativados ou instalados. Os principais impactos dessa categoria podem variar desde a possibilidade de um usuário conseguir acesso não autorizado às informações do sistema até o comprometimento total do sistema. Os CWE's mais relacionados consistem em *armazenamento de informações confidenciais em texto plano em um cookie* (CWE-315), *armazenamento de senha em arquivos de configuração* (CWE-260), *cookie sensível em sessão Hypertext Transfer Protocol Secure (HTTPS) sem atributo "seguro"* (CWE-614) e *página de erro personalizada ausente* (CWE-756).

6. **Vulnerable and Outdated Components:** Este tipo de vulnerabilidade ocorre quando os desenvolvedores utilizam algum componente que tem alguma vulnerabilidade conhecida ou que não recebe mais atualizações do fornecedor. Ocorre principalmente devido as empresas, em sua maioria, priorizar a atualização somente daqueles componentes principais ou que possuem alguma vulnerabilidade mais crítica. Além disso, empresas de grande porte e com código complexo frequentemente enfrentam maiores desafios no processo de registrar todas as versões dos componentes, softwares ou tecnologias que estão sendo utilizados dentro da aplicação. O impacto de um componente com vulnerabilidades conhecidas ou que não possui mais suporte do fabricante se estende além das aplicações Web. Um exemplo disso consiste nos ataques de *ransomware*, que costumam explorar vulnerabilidades já conhecidas nos sistemas. Ainda, explorando uma vulnerabilidade conhecida, o atacante pode realizar um Remote Code Execution (RCE). O principal CWE que está relacionado a essa categoria é o *uso de componentes de terceiros não mantidos* (CWE-1104).

7. **Identification and Authentication Failures:** Falhas de identificação e autenticação ocorrem quando as funções relacionadas à identidade, autenticação ou gerenciamento de sessão de um usuário não são implementadas corretamente ou não são adequadamente protegidas. Esse tipo de vulnerabilidade permite que um atacante utilize métodos como a força bruta, que faz testes automatizados para verificar credenciais de acessos dentro de uma página de *login*. Assim, o ataque poderá ocasionar o comprometimento de *tokens* de sessão, comprometimento de senhas, que possibilitam a personificação de usuários e escalada de privilégios.

   Aplicações que não possuem um sistema para bloquear várias tentativas de *login* são um exemplo dessa vulnerabilidade (OWASP, 2022b). Isso possibilita que um invasor

efetue múltiplas combinações de *logins* e senhas de maneira automatizada de modo a encontrar combinações válidas e, consequentemente, descobrir os mesmos. Alguns dos CWEs relacionados a essa categoria são: *autenticação imprópria* (CWE-1104), *requisitos de senha fracos* (CWE-521), *expiração de sessão insuficiente* (CWE-613).

8. **Software and Data Integrity Failures:** Um dos principais fatores que leva às falhas de integridade no software e nos dados consiste na dificuldade de atualização de arquiteturas complexas. Ademais, desenvolvedores frequentemente usam *plug-ins*, módulos ou bibliotecas de repositórios públicos que podem ou não ser confiáveis. Portanto, devido a essa alta complexidade, falhas de integridade de software e dados ocorrem quando os dados críticos e atualizações de softwares são adicionados ao pipeline de entrega sem a verificação de potenciais riscos oferecidos para o sistema da empresa. A consequência de um pipeline de *Continuous Integration/Continuous Delivery* (CI/CD) inseguro pode acabar por introduzir um potencial acesso não autorizado, código malicioso ou comprometimento do sistema. Um caso famoso ligado a essa categoria foi o incidente da SolarWinds (OLADIMEJI; KERNER, 2022) onde por vários meses a empresa distribuiu uma atualização maliciosa altamente direcionada para mais de 18.000 organizações, das quais cerca de 100 organizações foram afetadas. Esse caso é uma das violações mais abrangentes e significativas da história. Alguns dos CWEs ligados à essa categoria são: *verificação insuficiente de autenticidade de dados* (CWE-345), *download do código sem verificação de integridade* (CWE-494) e *falta suporte para verificação de integridade* (CWE-353).

9. **Insufficient Logging and Monitoring Failures:** Ela ocorre quando uma empresa não segue práticas adequadas para registrar e monitorar os *logs* do sistema ou aplicação, *e.g.*, não registrar a detecção de falhas de *login*. Os CWEs relacionados são *neutralização de saída imprópria para logs* (CWE-117), *omissão de informações relevantes para a segurança* (CWE-223), *inserção de informações confidenciais no arquivo de log* (CWE-532) e *registro insuficiente* (CWE-778).

10. **Server-Side Request Forgery (SSRF):** Permite que um atacante use o próprio servidor para fazer requisições na rede interna dentro da infraestrutura de uma organização. Assim, ele consegue burlar sistemas de segurança para impedir que usuários comuns tenham acesso a partes restritas do sistema, como um painel de administrador. Isso ocorre porque o atacante faz a falsificação da requisição se passando pelo servidor. Para Jabiyev (JABIYEV et al., 2021) os ataques de Server-Side Request Forgery (SSRF) podem explorar os serviços internos de diferentes maneiras. Por exemplo, fazendo com que eles enviem *e-mails* de *spam* para executar comandos do sistema remotamente em servidores que executam aplicações Web. O impacto causado por esse ataque quando bem sucedido pode incluir leitura de arquivos internos, acesso ao armazenamento de metadados em ambientes da Amazon Web

Services (AWS), mapeamento da rede interna, exposição de dados confidenciais e comprometimento de arquivos internos.

2.3 Materiais e Métodos

Nesta seção são detalhados os materiais e métodos empregados no estudo de caso apresentado neste trabalho. Em suma, a metodologia adotada (ilustrada na Figura 1) compreende a execução da ferramenta OWASP *Zed Attack Proxy* (ZAP) (OWASP, 2022b), a qual envia requisições para a plataforma *Moodle* (DOUGIAMAS, 2022) (passo 1) e recebe respostas. Tais requisições podem incluir, por exemplo, tentativa de exploração de vulnerabilidades conhecidas. Por meio da análise das resposta recebidas (passo 2), a ferramenta gera um relatório de alertas de potenciais vulnerabilidades (passo 3).

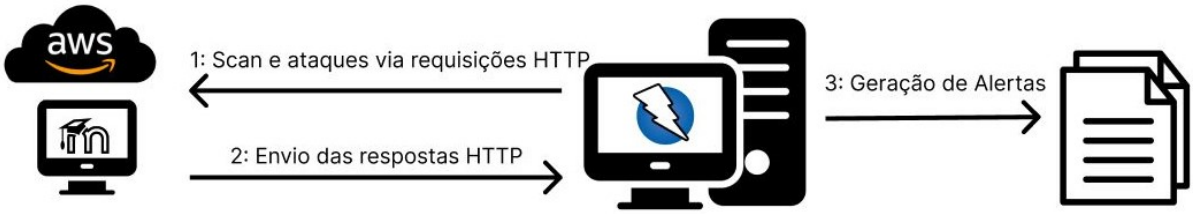

Figura 1 – Cenário de experimentação adotado.

A seguir são apresentados mais detalhes sobre tais componentes bem como o processo de estudo a partir do relatório obtido por meio do OWASP ZAP.

Além da funcionalidade de *scan* automático (ou varredura automática), o OWASP ZAP também pode ser utilizado de forma manual. Quando a função de *scan* automático é utilizado, o OWASP ZAP realiza diversas requisições para a aplicação em questão e, através das respostas das requisições, ele mapeia as vulnerabilidades encontradas fornecendo diversas informações úteis para a análise das mesmas. Já a utilização do modo de exploração manual permitirá que o OWASP ZAP seja utilizado como um *proxy* no navegador escolhido e interceptará todo o tráfego que passa pela interface de rede. Através de tal interceptação, é possível que o usuário manipule os parâmetros de requisições Web antes de enviá-las para o destino.

Os principais benefícios de se usar essa ferramenta consiste nas diversas funcionalidades gratuitas oferecidas (*e.g.,* Varredura automatizada de aplicações Web, *Web Spidering* e *Unthrottled Intruder*), em contraste com seu principal concorrente *Burp Suite* que apenas possui esses recursos na versão *premium*. Portanto, a ferramenta OWASP ZAP será empregada neste trabalho com a finalidade de efetuar análises a partir da plataforma *Moodle* (DOUGIAMAS, 2022).

Por fim, o cenário adotado para a realização do experimento consiste na instalação do OWASP ZAP na versão 2.12.0 e do *Moodle* na versão 4.1, lançada em 28 de novembro

de 2022. A instalação do OWASP ZAP se deu por meio de uma máquina virtual com o sistema operacional Kali Linux na versão 2022.3. A plataforma *Moodle* foi instalada em uma máquina virtual hospedada no serviço Amazon AWS, com sistema operacional Ubuntu 22.04 e com o servidor Apache na versão 2.4.52.

2.4 Trabalhos Correlatos

Atualmente, a literatura contêm trabalhos que utilizam a lista *OWASP Top 10* como referência para o estudo de segurança na Web em diversos contextos, inclusive em aplicações de ensino e aprendizagem. Nesta seção, tais trabalhos são discutidos. A Tabela 1 sumariza os principais trabalhos relacionados.

O trabalho de Priyawati, Rokhmah e Utomo (2022) propõe um *pentest* do tipo *grey box* em uma aplicação Web para determinar as vulnerabilidades presentes. Os autores analisam os resultados obtidos e sugerem melhorias na segurança do sistema para os desenvolvedores. No entanto, a versão da lista *OWASP Top 10* adotada é de 2017. Outra limitação desse trabalho consiste na ausência de validações acerca das vulnerabilidades encontradas pela ferramenta utilizada. Por fim, não há quaisquer experimentos ou provas de conceito que visem demonstrar de forma prática a exploração das vulnerabilidades discutidas.

Em Sampaio (2021), é apresentada uma análise das principais vulnerabilidades da OWASP em aplicações Web. Para cada vulnerabilidade apresentada, o autor executa experimentos práticos e discute métodos de mitigação para cada uma das categorias do *OWASP Top 10*. Como prova de conceito, as vulnerabilidades são implantadas em laboratórios fornecidos pelas instituições Portswigger e Acunetix para fins de estudo. Porém, a versão utilizada da OWASP pelos autores está desatualizada (2017). Adicionalmente, uma limitação importante nesse trabalho consiste no fato de que o estudo não está calcado em uma aplicação que é utilizada na prática.

O trabalho Monteverde (2014) tem como objetivo identificar e analisar as principais vulnerabilidades Web baseadas na OWASP. O autor apresenta (i) descrição de cada vulnerabilidade, (ii) demonstração de como ela ocorre em um ambiente e (iii) metodologias para mitigá-la. Dentre as ferramentas utilizadas para detectar vulnerabilidades destacam-se a *Web Application Attack and Audit Framework* (W3AF), SQLmap, Nmap e Burp Suite. Diversas categorias de sites Web foram testados, incluindo o *Moodle*. Por fim, para cada vulnerabilidade encontrada o autor propõe uma prova de conceito demonstrando os passos seguidos para encontrá-la. Entretanto, o autor utiliza a versão da lista OWASP de 2013. Ademais, não foi utilizado o OWASP ZAP para realizar o *scan* de vulnerabilidades (ou varredura de vulnerabilidades).

O trabalho Ahamed et al. (2022) tem como objetivo investigar as vulnerabilidades

Tabela 1 – Sumário de trabalhos relacionados.

Ref.	**Escopo**	**Metodologia**	**Ferramentas**	**Aplicação**	**Contramedidas**
(HERNANDEZ; CHAVEZ, 2008)	Vulnerabilidades em Geral	Análise Manual	Várias	Moodle 1.8.6	Contempla
(MONTEVERDE, 2014)	*OWASP Top 10* (2013)	Scan Automático	Várias	Aplicações Web Diversas	Não Contempla
(SILVA et al., 2014)	*OWASP Top 10* (2013)	Teste de Penetração	Várias	Moodle 2.7.1	Contempla
(CUERVO; ALDANA; LÓPEZ, 2016)	Vulnerabilidades em Geral	Scan Automático	Várias	Dokeos VLEs, Moodle 2.4.5	Não Contempla
(SAMPAIO, 2021)	*OWASP Top 10* (2017)	Exploração Manual	-	Portswigger, Acunetix	Contempla
(AHAMED et al., 2022)	*OWASP Top 10* (2017)	Scan Automático	Várias	Aplicações Web Governamentais	Não Contempla
(PRIYAWATI; ROKHMAH; UTOMO, 2022)	*OWASP Top 10* (2021)	Gray-box testing	OWASP ZAP	Aplicação Web	Não Contempla
(MUDIYANSELAGE; PAN, 2020)	*OWASP Top 10* (2013)	Análise de código fonte, Teste de Penetração, Scan Automático	RIPS	Moodle 2.6	Não Contempla
Este trabalho	***OWASP Top 10* (2021)**	**Scan Automático, Análise Técnica**	**OWASP ZAP**	**Moodle 4.1**	**Contempla**

dos sites governamentais de Bangladesh para determinar se existe uma correlação entre popularidade e número de vulnerabilidades encontradas. Para tanto, os autores adotam as ferramentas OWASP ZAP, BurpSuite e NetSpeaker. Complementarmente, a eficiência de tais ferramentas são comparadas para verificar qual detectou mais vulnerabilidades nas aplicações. Contudo, os autores consideram apenas a versão desatualizada da OWASP, publicada em 2017. Outra limitação é que os testes não envolveram a plataforma *Moodle*.

Os autores Silva et al. (2014) estudam a segurança da plataforma *Moodle*. O principal objetivo do trabalho consiste em realizar um teste de penetração utilizando como referência as vulnerabilidades da *OWASP Top 10* de 2013. As ferramentas Nikto, *Metasploit Framework*, W3AF, *Spidering*, *WebScarab* e *OWASP ZAP* são adotadas para execução de testes. Como resultados, os autores indicam métodos para mitigar as vulnerabilidades encontradas. Embora os autores apresentem um estudo interessante sobre segurança Web, as versões tanto da OWASP (2013) quanto do *Moodle* (v. 2.7.1) estão desatualizadas. Atualmente, a versão mais recente do *Moodle* é a 4.1.

O trabalho Cuervo, Aldana e López (2016) tem por objetivo pesquisar regras e padrões de seguranças com ênfase em ferramentas de segurança e análise de vulnerabilidades focadas em aplicações de *Virtual Learning Environment* (VLE). O estudo foi feito utilizando as plataformas open-source *Dokeos VLEs* e *Moodle*. Entretanto, os autores apenas citam as ferramentas utilizadas (W3AF, SQLMap e OWASP ZAP), sem, entretanto, tornar claro seu uso e como os testes foram realizados a fim de encontrar as vulnerabilidades relatadas. Além disso, fazem uma descrição superficial sobre cada vulnerabilidade, deixando de citar como elas funcionam e como ocorrem. Somado a isso, a versão utilizada das plataformas estão desatualizadas. O trabalho não está baseado no *ranking* de vulnerabilidades Web publicado pela OWASP.

O trabalho Hernandez e Chavez (2008) tem por objetivo discutir os serviços de segurança na plataforma *Moodle*. Os autores apresentam ataques para a descoberta de vulnerabilidades. Além disso, algumas propostas de mitigação são apresentadas. Entretanto, não á adotada lista vigente do *OWASP Top 10*. Inclusive, o trabalho não deixa claro a ferramenta utilizada para execução do testes. Por fim, a versão do *Moodle* utilizada já está ultrapassada.

O trabalho Mudiyanselage e Pan (2020) tem por objetivo encontrar uma solução simples e eficaz para realizar testes de segurança no *Moodle*. Os autores, em primeiro momento, discutem sobre as vulnerabilidades do Top 10 da OWASP. Então, explicam a metodologia que utilizada para realizar os testes, a qual consiste na análise de código fonte, realização de *pentest* no *Moodle* e, posteriormente, a identificação de vulnerabilidades utilizando *scanners*. Ao fim da metodologia os autores discutem sobre o risco de cada vulnerabilidade encontrada por eles e seus respectivos impactos. Entretanto, o trabalho não aborda metodologias para mitigar ou corrigir essas vulnerabilidades. Ademais, as

versões utilizadas do *OWASP Top 10* e do *Moodle* estão desatualizadas.

Os trabalhos relacionados apresentam algumas limitações importantes. Em geral, observa-se o uso de versões desatualizadas da lista OWASP Top 10, o que pode resultar em uma análise incompleta ou não abrangente das vulnerabilidades mais recentes. Além disso, alguns estudos não apresentam validações adequadas das vulnerabilidades encontradas, comprometendo a confiabilidade dos resultados. Outras limitações incluem a falta de experimentos práticos ou provas de conceito para demonstrar a exploração das vulnerabilidades discutidas, bem como a ausência de testes específicos no ambiente do *Moodle*, que é o foco deste trabalho. Portanto, há espaço para melhorias e para uma abordagem mais aprofundada e atualizada na identificação e mitigação de vulnerabilidades no contexto específico do *Moodle*.

3 Desenvolvimento

Este capítulo visa mostrar e explicar os métodos utilizados para a realização dos experimentos demonstrados no capítulo 4. Primeiramente, descreve-se o processo de instalação do *Moodle*, abordando a configuração do servidor Apache, do banco de dados MySQL e do PHP. Em seguida, detalha-se o uso do OWASP ZAP para realizar um *scan* de vulnerabilidades no *Moodle*, descrevendo as etapas de: (i)*spidering*, (ii) análise e (iii) ataque ativo realizadas pela ferramenta. Além disso, discutiremos o processo de auditoria e monitoramento de comportamento, abordando o uso das ferramentas THC Hydra, WFuzz, GoBuster e OWASP ZAP, bem como a geração de *logs* de tráfego comum. Ao final desta seção, discutiremos sobre a detecção de ações maliciosas e a correlação entre as informações presentes nos *logs* para identificar possíveis ataques.

3.1 Instalação do Moodle

Para instalar o *Moodle*, primeiramente é necessário instalar o servidor e o banco de dados onde o mesmo será hospedado. Então, para realizar a instalação do servidor Apache, utilizou-se o comando:

```
apt-get install apache2
```

E para realizar a instalação do banco de dados MySQL, utilizou-se os comandos:

```
sudo apt install mysql-server
sudo mysql\_secure\_installation
```

O segundo comando serve para realizar o procedimento de definir usuário e senha para o MySQL tornando-o mais seguro. O *Moodle* também possui o pré-requisito de instalar o *PHP5* no qual a instalação se deu pelo comando:

```
apt-get install php5 libapache2-mod-php5
```

Com todos os pré-requisitos instalados na máquina, pode se dar início ao processo de instalação do *Moodle* no servidor Web local. Primeiro, realizou-se o *download* do arquivo de instalação no site oficial[1]. Em seguida, moveu-se o arquivo baixado na etapa anterior para o diretório */www/var* do computador. Após isso, a instalação se da por meio de uma interface gráfica apresentada no navegador Web através da seguinte URL:

[1] Disponível em: <https://moodle.com/>

http://localhost/moodle para finalizar a instalação do *Moodle*, seguiu-se o passo a passo que é mostrado na interface gráfica.

3.2 Relatório do OWASP ZAP

Após a configuração do cenário, utilizou-se o OWASP ZAP para realizar um *scan* de vulnerabilidades (ou varredura de vulnerabilidades) no *Moodle*. Para utilizar o OWASP ZAP, primeiro inicia-se sua interface gráfica e após isso seleciona-se o tipo de *scan* que deseja realizar. Há duas opções de *scan* disponíveis: *Automated Scan* e *Manual Explore*. Após selecionar o tipo de *scan* desejado, é necessário configurá-lo para realizar o ataque ao alvo desejado. Neste trabalho a opção utilizada para a realização dos testes foi a de *scan* automático, no qual é divida em várias etapas que serão detalhadas a seguir:

- A primeira etapa é chamada de *spidering*, no qual o OWASP ZAP realiza diversas requisições HTTP para a aplicação Web com intuito de coletar informações e criar um mapa. Esse mapa é necessário para a identificação de pontos de entrada onde possíveis vulnerabilidades podem ser identificadas e exploradas.
- Durante a segunda etapa inicia-se o OWASP ZAP realiza uma análise das informações coletadas a fim de detectar possíveis vulnerabilidades nas *URLs* que foram encontradas durante a etapa de *spidering*.
- A terceira etapa é conhecida como ataque ativo (ou *Active Scan*) no qual o OWASP ZAP irá realizar diversos ataques para possivelmente identificar vulnerabilidades que não foram encontradas nas etapas anteriores, realizando assim testes mais profundos nas *URLs* encontradas.
- Por fim, ao final da etapa de ataque ativo, o usuário pode realizar uma verificação manual para verificar se há alguma falha que não foi reportada.

Após todas as etapas serem concluídas o OWASP ZAP irá gerar um relatório com informações detalhadas sobre cada vulnerabilidade ou alerta encontrado, juntamente com algumas recomendações para corrigi-las. O relatório pode ser gerado nos formatos *Portable Document Format* (PDF), HTML, *eXtensible Markup Language* (XML) ou *Comma Separated Values* (CSV).

3.3 Auditoria e Monitoramento de Comportamento

Nesta etapa, a metodologia empregada envolveu o uso de ferramentas capazes de executar ataques, com o objetivo de gerar registros de *logs* para a posterior análise. Tal análise tem como objetivo identificar as fontes de informações e as características

representativas para cada perfil de ataque. Após gerar os registros de *logs* dos ataques, pretende-se estudá-los no Capítulo 4, a fim de extrair informações valiosas para a detecção de intrusões, do inglês, *Intrusion Detection Systems* (IDSs) (QUINCOZES et al., 2021a).

A análise das fontes de informações e características relevantes é fundamental para o desenvolvimento de IDSs capazes de identificar e alertar sobre diferentes tipos de ataques. Ao compreender as características distintivas de cada perfil de ataque, é possível construir modelos de detecção mais eficazes e melhorar a capacidade de resposta a incidentes de segurança. A seguir, na Seção 3.3.1 serão apresentadas as ferramentas e metodologias usadas para a geração de comportamento malicioso. Em seguida, na Seção 3.3.3, serão analisadas as informações e suas respectivas fontes que podem ser extraídas e monitoradas para a detecção dos comportamentos maliciosos executados a partir dessas ferramentas.

3.3.1 Geração de Comportamentos Maliciosos

As ferramentas listadas a seguir foram usadas com o intuito de explorar potenciais vulnerabilidades no *Moodle* e, dessa forma, produzir registros de comportamentos maliciosos para a posterior análise. Além dessas ferramentas, a seguir é descrita a geração de comportamento típico de um usuário legítimo, isto é, sem tentativas ataques.

3.3.1.1 THC Hydra

O THC Hydra é uma ferramenta usada para realizar ataques de força bruta. Neste trabalho, o objetivo era conduzir um ataque de força bruta de duas maneiras diferentes. Na primeira abordagem, o ataque foi direcionado ao usuário, assumindo que a senha já era conhecida. Na segunda abordagem, foi realizado o oposto, ou seja, o ataque foi focado na senha, assumindo que o usuário era conhecido. O comando a ser utilizado com o THC Hydra é o seguinte:

```
hydra -L -wordlist de usuarios- -p senhafixa
ec2-54-227-153-250.compute-1.amazonaws.com
http-post-form "/moodle/login/index.php:username="USER"
&password=senhafixa:Mensagem de erro."
```

Em resumo, o THC Hydra será utilizado para executar um ataque de força bruta, testando uma lista de usuários com uma senha fixa no formulário de *login* do *Moodle*. Essa abordagem visa explorar possíveis combinações de usuários com a senha especificada para tentar obter acesso não autorizado ao sistema.

3.3.1.2 WFuzz

O WFuzz é uma ferramenta utilizada em testes de penetração e exploração de vulnerabilidades em aplicações Web. Neste trabalho o objetivo era realizar um ataque de

dicionário para descobrir possíveis arquivos sensíveis no *Moodle*. O comando a ser utilizado com o WFuzz é o seguinte:

```
wfuzz -t 30 -z file /usr/share/seclists/Discovery/Web-Content/apache.txt
--hl 0 http://ec2-54-227-153-250.compute-1.amazonaws.com/moodle/FUZZ
```

Em resumo, o WFuzz será utilizado para executar um ataque de dicionário, testando uma lista de nomes de arquivos conhecidos na URL do *Moodle*. Essa abordagem visa encontrar arquivos que contenham informações sensíveis.

3.3.1.3 GoBuster

O GoBuster é uma ferramenta utilizada para enumeração de diretórios e subdomínios de uma aplicação Web. Neste trabalho o objetivo era realizar uma enumeração dos diretórios existentes na aplicação para assim conseguir identificar quais recursos estão disponíveis. O comando a ser utilizado com o GoBuster é o seguinte:

```
gobuster dir -u http://ec2-54-227-153-250.compute-1.amazonaws.com/moodle/
-w /usr/share/seclists/Discovery/Web-Content/directory-list-lowercase-2.3
-big.txt -t 20 -e -no-error -r
```

Em resumo, o GoBuster será utilizado para executar um ataque de dicionário, testando uma lista com nomes de diretórios conhecidos na URL do *Moodle*. Essa abordagem visa encontrar diretórios ou recursos ocultos que contenham informações sensíveis.

3.3.1.4 OWASP ZAP

Os passos utilizados para gerar os *logs* utilizando o OWASP ZAP foram semelhantes aos descritos na seção 3.2. No entanto, nesta ocasião, o objetivo principal foi gerar os *logs* que conterão informações relevantes para a detecção de intrusões e análise de segurança.

Ao executar o OWASP ZAP, foram realizadas as mesmas etapas descritas anteriormente, sem a necessidade de gerar um relatório detalhado novamente. Os *logs* resultantes serão analisados em busca de informações valiosas que possam indicar atividades suspeitas ou tentativas de intrusão no sistema.

3.3.2 Comportamento Legítimo

Esses *logs* são gerados quando um usuário comum que utiliza o *Moodle*, ou seja, quando o mesmo faz *login* em sua conta, entra nos cursos, responde ou cria tópicos de discussão, entre outras atividades típicas de um usuário do *Moodle*.

3.3.3 Auditoria e Monitoramento

A fim de permitir a auditoria de registros para o monitoramento comportamental de usuários legítimos e maliciosos no *Moodle*, foram identificadas três fontes de informações relevantes que possuíam características representativas para a detecção dos ataques executados pelas ferramentas empregadas neste trabalho, as quais são listadas a seguir:

- **Fonte 1: Registros de ações de usuários no *Moodle***: essa informação é armazenada pela própria plataforma *Moodle* na tabela `mdl_logstore_standard_log`, dentro de seu banco de dados. Dentre as informações contidas nessa fonte de registros estão várias atividades e eventos que ocorrem no sistema. Esta tabela irá registrar qualquer informação de interação do usuário com a plataforma, como por exemplo o *login*, acesso aos cursos, interações com atividades (*e.g.,* envio de trabalhos, realização de tarefas, entre outros).

- **Fonte 2: Registros de Acesso ao *Apache***: esse tipo de informação é extraída do arquivo de *log* de acessos do *Apache* (`access.log`). Tal arquivo registra todas as solicitações que o servidor *Web Apache* (*i.e.,* que hospeda o *Moodle*) recebe. Para cada solicitação, o *log* de acesso pode conter detalhes como o endereço *Internet Protocol* (IP) do cliente, data e hora da solicitação, método HTTP (*e.g.,* `GET`, `POST`), o caminho do recurso solicitado, o código de status HTTP retornado pelo servidor (*e.g.,* 200, 404, 500), o tamanho da resposta em *bytes*, o agente do usuário (*e.g., browser*, *robot*) e o referenciador (*i.e.,* a página da qual o usuário veio).

- **Fonte 3: Registros de Erros no *Apache***: esse tipo de informação é extraída do arquivo de *log* de erros do *Apache* (`error.log`). Nesse arquivo, são registrados detalhes sobre erros que ocorrem enquanto o servidor Web *Apache* está processando solicitações. Isso pode incluir erros de programação em *scripts* CGI ou PHP, problemas com arquivos *.htaccess*, problemas de configuração do Apache e muitos outros tipos de erros. O *log* de erro também pode ser configurado para mostrar avisos e mensagens de informação. Normalmente, cada entrada no *log* de erros inclui a data e hora do erro e uma descrição do mesmo.

A fim de melhor compreender os *logs* que foram produzidos a partir da execução das ferramentas detalhadas na Seção 3.3.1, antes de executar tais ferramentas, se fez necessária a exclusão de todo e qualquer *log* antigo. Para os *logs* do Apache, utilizou-se o comando `sudo truncate -s 0 *.log`, enquanto para a tabela de *logs* no MySQL, utilizou-se o comando `DELETE FROM mdl_logstore_standard_log;`.

Por fim, cabe ressaltar que a implementação de um IDS está além do escopo deste trabalho, portanto, esta seção se limita a apresentar e discutir as informações relevantes que devem ser monitoradas para a detecção dos ataques executados.

4 Experimentos e Análises

Este capítulo está dividido em três seções, a primeira irá contemplar a etapa de análise e teste do *Moodle*, a segunda etapa irá contemplar a demonstração de como foi realizada as correções para as vulnerabilidades e alertas encontrados e por fim a última etapa contempla uma análise dos *logs* gerados na Seção 3.

4.1 Etapa 1 - Teste com o *Moodle*

A Tabela 2 sumariza os alertas reportados pela ferramenta OWASP ZAP. Em particular, foram identificados 894 alertas que são divididos de acordo com seu risco: *Informativo*, *baixo*, *médio* e *alto*. Embora existam dez vulnerabilidades no *OWASP Top 10*, os alertas reportados pelo OWASP ZAP para a plataforma *Moodle* encontram-se entre as cinco primeiras vulnerabilidades (A01, A02, A03, A04 e A05, conforme Seção 2).

Tabela 2 – Alertas reportados pela ferramenta OWASP ZAP na plataforma *Moodle*.

Nome da Vulnerabilidade	**Risco**	**Categoria no *OWASP Top 10***	**Alertas**
SQL Injection	Alto	A03 - Injection	5
Missing Anti-CSRF tokens	Médio	A05 - Security Misconfiguration	56
CSP Header Not Set	Médio	A05 - Security Misconfiguration	37
HTTP to HTTPS (Insecure Transition in Form Post)	Médio	A02 - Cryptographic Failures	4
Hidden File Found	Médio	A05 - Security Misconfiguration	1
Missing Anti-Clickjacking Header	Médio	A05 - Security Misconfiguration	11
.httaccess Information Leak	Médio	A05 - Security Misconfiguration	2
Big Redirect Detected[1]	Baixo	A04 - Insecure Design	12
Cookie No HTTPOnly Flag	Baixo	A05 - Security Misconfiguration	5
Cookie without SameSite Attribute	Baixo	A01 - Broken Access Control	5
Disclosure of Date and Time - Unix	Baixo	A01 - Broken Access Control	150
Server Leaks Version Information[2]	Baixo	A05 - Security Misconfiguration	68
Strict-Transport-Security Header Not Set	Baixo	A05 - Security Misconfiguration	11
X-Content-Type-Option Header Missing	Baixo	A05 - Security Misconfiguration	65
Information Disclosure (Suspicious Comments)	Informativo	A01 - Broken Access Control	89
Information Disclosure (Sensitive Information in URL)	Informativo	A01 - Broken Access Control	2
Modern Web Application	Informativo	Nenhum	55
Re-examine Cache-control Directives	Informativo	Nenhum	8
User Agent Fuzzer	Informativo	Nenhum	240
User Controllable HTML Element Attribute (Potential XSS)	Informativo	A03 - Injection	68

Note que muitos dos alertas reportados são originados da falta de configurações adicionais além da instalação padrão do *Moodle*. O objetivo desta análise consiste em revelar o quão vulnerável pode ser uma instalação do *Moodle* quando realizada por um

usuário que não tem o perfil de especialista em segurança da informação, e portanto, não implementa medidas adicionais além daquelas que a própria plataforma oferece.

A seguir, os alertas reportados pela ferramenta OWASP ZAP são apresentados e discutidos. Como contribuição adicional, são descritos os principais ataques que podem ser realizados através da exploração de tais vulnerabilidades, bem como as devidas contramedidas recomendadas para cada situação.

4.1.1 Vulnerabilidades de Risco Alto

Dentre as vulnerabilidades reportadas e categorizadas pela ferramenta OWASP ZAP (OWASP, 2022b), apenas uma categoria foi classificada como tendo alto nível de risco. Foram gerados 5 alertas para a possibilidade de ataques de injeção de comandos através da vulnerabilidade *SQL Injection*. Essa vulnerabilidade pertence à categoria *Injection*, a qual ocupa a terceira posição no *OWASP Top 10*, conforme a Seção 2.

As falhas de SQL Injection são introduzidas quando os desenvolvedores de software criam consultas de banco de dados dinâmicas construídas com concatenação de *strings* que inclui entrada fornecida pelo usuário. Assim, uma potencial forma de ataque à plataforma *Moodle* pode se dar manipulando o valor do parâmetro *logintoken* ao enviar uma requisição *POST* a partir da página inicial */moodle/login/index.php*. De acordo com o relatório da ferramenta, os resultados da página foram manipulados com sucesso através de uma tentativa de ataque automatizado que usa as condições booleanas denotadas nas Equações 4.1 e 4.2.

$$[NeM7GmXnkNHvlEGtgNg5SIQ2rFw0EtCe' AND '1' = '1' - -] \tag{4.1}$$

$$[NeM7GmXnkNHvlEGtgNg5SIQ2rFw0EtCe' AND '1' = '2' - -] \tag{4.2}$$

Como principais contramedidas para a contenção de ataques de injeção, como o *SQL Injection*, recomenda-se verificar todos os dados no lado do servidor. Não se deve assumir que as informações provindas da aplicação cliente são sempre confiáveis, mesmo que hajam validações implementadas na aplicação cliente. Ademais, o uso de funções como o `mysqli_real_escape_string` ou procedimentos (*procedures*) podem ser usados como formas de mitigação da vulnerabilidade apresentada. Por fim, a adoção do princípio de privilégio mínimo é fortemente recomendada como forma de garantir que um atacante que explore a vulnerabilidade de *SQL Injection* esteja limitado às permissões de seu grupo de usuários (CHEATSHEETS-SERIES, 2022).

4.1.2 Vulnerabilidades de Risco Médio

As vulnerabilidades reportadas pelo OWASP ZAP como risco médio constituem variações das vulnerabilidades *Cryptographic Failures* e *Security Misconfiguration*, as quais ocupam a segunda e quinta posição no *OWASP Top 10*, respectivamente. Foram gerados 111 alertas para seis tipos subcategorias dessas vulnerabilidades, conforme segue.

A vulnerabilidade *Cross-Site Request Forgery* (CSRF) consiste na falsificação de solicitação entre sites. No contexto da plataforma *Moodle*, foram reportados 56 alertas devido à ausência de *tokens* que são tipicamente usados para prevenir ataques que exploram essa vulnerabilidade. Um *token anti-CSRF* deve conter um valor aleatório seguro, como um *nonce*, por exemplo. Recomenda-se que o *token* seja exclusivo por sessão do usuário. Ademais, o mesmo deve ser representado por um valor de tamanho grande e aleatório para dificultar a sua descoberta. Ao explorar essa vulnerabilidade, um atacante pode forçar que a vítima envie solicitações HTTP para um destino sem seu conhecimento ou intenção. Nas fases de arquitetura e design, as contramedidas cabíveis incluem a geração e verificação de *nonces* exclusivos para formulários. Já na fase de implementação, uma contramedida que pode ser adotada consiste na verificação do cabeçalho HTTP *Referer* de modo a confirmar se a solicitação se originou de uma página esperada. Note que a segunda solução pode ser ineficiente em casos em que usuários ou *proxies* legítimos desabilitam o envio deste cabeçalho por razões de privacidade (OWASP, 2022c).

Os 37 alertas de *Content Security Policy* (CSP) *Header Not Set* ocorreram devido a falta de configuração deste cabeçalho HTTP. É importante observar que tal configuração não faz parte de um procedimento previsto durante a instalação padrão da plataforma *Moodle*. Portanto, instalações da plataforma que não contemplarem tais medidas adicionais podem ficar vulneráveis. Um atacante pode explorar tais vulnerabilidades para executar ataques de injeção de dados, por exemplo. Como forma de mitigação, deve-se assegurar que o servidor Web esteja devidamente configurado para suportar a definição deste cabeçalho pelo cliente. As definições de cabeçalhos CSP podem variar de acordo com o navegador e suas versões (OWASP, 2022c):

- *X-WebKit-CSP*: Google Chrome (versão 14+) e Safari (versão 6+);
- *X-Content-Security-Policy*: Firefox (versão 4+) e Internet Explorer (versão 10+);
- *X-Content-Security-Policy*: Google Chrome (versão 25+), Firefox (versão 23 ou superior) e Safari (versão 7+).

Os 4 alertas *HTTP to HTTPS Insecure Transition in Form Post* revelam que a plataforma *Moodle* não foi instalada em um ambiente seguro que contemple o uso do protocolo *Hyper Text Transfer Protocol Secure* (HTTPS). Por conta disso a aplicação fica vulnerável à ataques *Man-In-The-Middle* (MITM), possibilitando que um invasor consiga interceptar a troca de dados entre duas partes e roubar suas informações. Como forma de mitigação, recomenda-se fortemente a utilização do protocolo HTTPS ao invés do HTTP.

O alerta de *Hidden File Found* revela que há um arquivo potencialmente confidencial que pode estar exposto a usuários não autorizados. Isso pode permitir que usuários mal-intencionados possam obter informações administrativas, obter conhecimento da

configuração da aplicação e conseguir credenciais de acesso, possibilitando que ele consiga expandir os métodos de ataques que ele pode fazer dentro da aplicação. Uma outra possibilidade consiste em realizar engenharia social usando as informações obtidas. Portanto, para mitigar essa possibilidade de ataque, recomenda-se fortemente desativar da produção qualquer componente que seja desnecessário. Se o componente for necessário, é importante garantir que o seu acesso seja restrito a usuários autenticados.

Os 11 alertas gerados por *Missing Anti-Clickjacking Header* ocorreram devido à instalação da plataforma*Moodle* não contemplar a configuração da política de segurança de conteúdo com diretiva '*frame-ancestors*' ou *X-Frame-Options*. Por conta disso, um atacante pode executar o ataque de *Clickjacking*. Neste ataque, o agente malicioso consegue infectar os botões da aplicação Web fazendo com que eles executem ações maliciosas ao serem clicados pelo usuário. Para mitigar essa vulnerabilidade, é necessário certificar-se que os cabeçalhos *Content-Security-Policy* e *X-Frame-Options* estejam configurados para todas as páginas Webs utilizadas pela aplicação.

Os 2 alertas de *.httaccess Information Leak* foram gerados devido ao *scan* do ZAP conseguir acessar o arquivo *httaccess* da aplicação. Esse arquivo em questão pode ser utilizado para realizar alterações nas configurações do Servidor Apache. Portanto, um usuário mal-intencionado, ao conseguir acesso a este arquivo, pode realizar a ativação e a desativação de funcionalidades que o Servidor Apache pode oferecer. Portanto, a forma de se mitigar essa vulnerabilidade, é desativando qualquer tipo de acesso a este arquivo.

4.1.3 Vulnerabilidades de Risco Baixo

As vulnerabilidades reportadas pelo OWASP ZAP como risco baixo constituem variações das categorias de *Security Misconfiguration*, *Broken Access Control* e *Insecure Design*. No total, foram gerados 316 alertas para sete tipos de subcategorias dessas vulnerabilidades, conforme detalhado a seguir.

Os 5 alertas de *Cookie No HTTPOnly* foram gerados devido a ausência do sinalizador HTTPOnly. Este sinalizador consiste em um sinalizador adicional incluído no cabeçalho de resposta *HTTP Set-Cookie*. Uma vez que esse cabeçalho não é devidamente configurado por usuários que executam a instalação do *Moodle*, a aplicação fica sujeita a determinados tipos de ataques. O principal ataque ocasionado pela falta desse sinalizador é o acesso de *cookies* armazenados no lado do cliente. Esse ataque pode ser viabilizado por meio de *scripts*. Um exemplo de falha que pode ser explorada para revelar o *cookie* de um usuário para terceiros é o *Cross-Site Scripting*. Portanto, uma forma de mitigar tal acesso de *cookies* por terceiros compreende a inclusão do sinalizador *HTTPOnly* no cabeçalho de resposta HTTP (OWASP, 2022c).

Os 5 alertas de *Cookie without SameSite Attribute* foram gerados devido à ausência

do atributo *SameSite* no cabeçalho HTTP *Set-Cookie*. Tal atributo tem por finalidade evitar que um *cookie* seja enviado como resultado de uma solicitação entre sites. Dessa forma, um tipo de ataque que pode explorar essa falta de configuração consiste no CSRF, já mencionado anteriormente. Então, para mitigar a possibilidade de um atacante falsificar solicitações entre sites através do envio de *cookies*, deve-se habilitar o atributo *SameSite* no cabeçalho de resposta (OWASP, 2022c).

Os 12 alertas de *Big Redirect Detected (Potential Sensitive Information Leak)* ocorreram devido ao tamanho da resposta recebida pelo servidor no momento em que o mesmo efetua redirecionamentos, o qual foi considerado longo pela ferramenta OWASP ZAP. O potencial risco dessa vulnerabilidade consiste nos casos em que a resposta do redirecionamento contém informações confidenciais ou dados pessoais. Ademais, com base em tais informações, novos ataques podem ser explorados. Portanto, o jeito mais eficaz de mitigar essa vulnerabilidade é configurar o servidor para que o mesmo não exiba conteúdos confidenciais ou privados no redirecionamento (OWASP, 2022c).

Os 150 alertas de *Disclosure of Date and Time - Unix* ocorreram devido ao envio das informações de data e hora do servidor na resposta HTTP. O principal problema do vazamento dessa informação consiste em facilitar a um atacante a obtenção de informações internas para a criação de padrões de exploração. Para mitigar esse tipo de vazamento de informação, deve-se verificar se o uso dessa informação por parte de um atacante pode ser agregada com outras informações para o mapeamento de padrões por atacantes. Em caso positivo, deve-se remover o campo *timestamp* do cabeçalho resposta (OWASP, 2022c).

Os 68 alertas de *Server Leaks Version Information* ocorreram devido ao vazamento da versão do servidor *Web* na resposta HTTP. Esse tipo de informação pode ser usada pelo atacante para identificar outras vulnerabilidades que são conhecidas naquela versão em que o servidor se encontra. Como forma de mitigação, pode-se remover o campo *"Server"* do cabeçalho de resposta HTTP.

Os 11 alertas gerados por *Strict-Transport-Security Header Not Set* ocorreram devido a configuração deste cabeçalho HTTP não ser um procedimento que faz parte da instalação padrão da plataforma *Moodle*. O HTTP *Strict-Transport-Security* (HSTS) consiste em um aprimoramento de segurança opcional redireciona solicitações HTTP não seguras para HTTPS. A falta da utilização desse cabeçalho possibilita que ataques como *Man-In-The-Middle* ocorra. Portanto, como forma de mitigação, deve-se realizar a configuração deste cabeçalho no servidor Web.

Os 65 alertas de *X-Content-Type-Option Header Missing* ocorreram devido ao procedimento de instalação da plataforma *Moodle* não contemplar a configuração do cabeçalho *X-Content-Type-Option*, o qual faz parte do mecanismo *Anti-Mime-Sniffing*. Essa falta de configuração pode fazer com que um usuário mal-intencionado consiga disfarçar um arquivo, fazendo com que se pareça com outro, possibilitando por exemplo, ataques do

tipo XSS. Portanto, como forma de mitigação recomenda-se a configuração do cabeçalho *X-Content-Type-Options* como *nosniff* para todas as páginas da Web.

4.1.4 Motivos Informativos

Os 89 alertas de *Suspicious Comments* foram gerados devido às respostas conterem comentários suspeitos. Isso contribui com um atacante à medida que pode fornecer informações sobre o funcionamento do sistema. Ademais, um atacante pode conseguir credenciais de acesso, caso elas estejam disponíveis nos comentários. Um método para mitigar essa vulnerabilidade consiste em remover quaisquer comentários desnecessários ou que possam explanar informações adicionais sobre o funcionamento do sistema.

Os 2 alertas de *Information Disclosure - Sensitive Information in URL* foram emitidos devido a suspeita de vazamento de informações confidenciais na URL: </moodle/login/index.php?testsession=2>. Incluir informações confidenciais em uma URL aumenta o risco de que informações relevantes para um ataque sejam capturadas por um invasor. No entanto, a mitigação deste risco é simples, basta não transmitir informações sensíveis na URL, tradicionalmente feitas usando o método `GET`.

Os 55 alertas do *Modern Web Applications* foram gerados como um informativo de que a aplicação *Moodle* contém recursos modernos e pode ser melhor explorada por ferramentas tais como o *Ajax Spider*. Portanto, uma vez que isso não representa uma vulnerabilidade propriamente dita, nenhuma ação é necessária como contramedida.

Os 8 alertas de *Re-examine Cache-control Directives* foram gerados devido a ausência ou má configuração de cabeçalhos de controle de cache. Isso permite que navegadores e *proxies* acessem e alterem o conteúdo que encontra-se em cache. Com isso, um atacante pode explorar tal vulnerabilidade para executar o envenenamento de *cache*, manipulando, por exemplo, o Sistema de Nomes de Domínio, do inglês, *Domain Name System* (DNS). Como forma de mitigação, recomenda-se definir o cabeçalho HTTP de controle de cache como `no-cache`, `no-store`, `must-revalidate`. Caso haja a necessidade do armazenamento de recursos em *cache*, recomenda-se a definição das diretivas `public`, `max-age`, `immutable` para mitigar e proteger a aplicação de potenciais ataques.

Os 240 alertas para *User Agent Fuzzer* foram gerados devido ao recebimento de diferentes respostas ao modificar o campo *User-Agent* na requisição. Essa potencial vulnerabilidade demonstra que a aplicação pode estar sujeiras a falhas oriundas de modificações no campo *User-Agent* da requisição. Para mitigar essa vulnerabilidade, deve-se configurar as URLs para fornecer uma resposta independentemente do *User-Agent*.

Os 68 alertas de *User Controllabe HTML Element Attribute (Potencial XSS)* gerados devido ao OWASP ZAP identificar pontos onde determinados atributos HTML podem ser controlados por um usuário através do método POST. Por conta disso, um usuário

malicioso pode utilizar dessa vulnerabilidade para realizar a execução de ataques XSS podendo por exemplo, roubar os *cookies* de sessão de um usuário, outra forma é a realização de ataques de *HTML Injection*, onde o invasor pode realizar uma desfiguração (*deface*) na aplicação. Uma forma de mitigar essa vulnerabilidade é validar todas as entradas do usuário e higienizar a saída antes de gravar-la em atributos HTML.

4.2 Etapa 2 - Correção das Vulnerabilidades

Esta seção tem por objetivo mostrar as configurações realizadas de modo a corrigir e mitigar os alertas mostrados na Seção 4.1.

4.2.1 Vulnerabilidades de Risco Alto

De modo a verificar se a aplicação *Moodle* é vulnerável à ataques do tipo SQL Injection, foram empregadas as ferramentas para testes automáticos e testes manuais, a saber o *SQL Map* e *Burp Suite*, respectivamente. Para realizar a validação manual, as configurações do *Burp Suite* foram ajustadas para interceptar a solicitação HTTP que havia gerado o alerta de *SQL Injection* no *OWASP ZAP*. Durante o teste, o parâmetro "*logintoken*", que havia sido reportado na requisição testada pelo OWASP ZAP, foi testado com os valores sugeridos pela ferramenta. No entanto, não houve confirmação de uma injeção de SQL bem-sucedida. Para realizar a validação automática, foram inseridos diversos *payloads* de injeção de SQL no parâmetro "*logintoken*". No entanto, o SQL Map não foi capaz de explorar com sucesso nenhuma dessas injeções. Isso sugere que a plataforma *Moodle* estava protegida contra vulnerabilidades de injeção de SQL. Dessa forma, verificou-se que o alerta de *SQL Injection* se trata de um falso positivo.

4.2.2 Vulnerabilidades de Risco Médio

4.2.2.1 Missing Anti-CSRF tokens

Atualmente o *Moodle* implementa o uso de um *token* de sessão para mitigar os riscos dos ataques de CSRF, de acordo com a documentação de *CSRF Prevention Cheat Seet* da própria OWASP, tal medida se apresenta como aceitável para mitigar esses ataques.

4.2.2.2 CSP Header Not Set

Foi observado que de fato não havia o cabeçalho CSP nas respostas HTTP, então foram realizadas duas tentativas de configuração. A primeira tentativa foi utilizar o *plug-in* disponibilizado pelo próprio *Moodle*, que pode ser baixado e instalado pela parte administrativa do site. A segunda maneira é a configuração do cabeçalho dentro do arquivo de configuração do Apache (*apache2.conf)* adicionando a seguinte linha para configurá-lo:

```
Header set Content-Security-Policy "default-src 'self';"
```

4.2.2.3 HTTP to HTTPS Insecure Transition in Form Post

De modo a corrigir este alerta, configurou-se um certificado digital de modo a permitir a implementação e uso do protocolo HTTPS, o qual consiste em uma implementação de segurança que utiliza o protocolo Transporte Layer Security/ Secure Sockets Layer (TLS/SSL).

4.2.2.4 Hidden File Found

Neste alerta, ao entrar na URL que o OWASP ZAP acusou de ter um arquivo oculto, verificamos que de fato o arquivo oculto *composer.lock* estava exposto para todos os usuários. Então a medida tomada para resolver esse alerta foi criar a seguinte linha no arquivo de configuração do apache:

```
<Files *.lock>
Require all denied
</Files>
```

Com esta linha de configuração, todos os acessos aos arquivos terminados com *.lock* serão negados sendo redirecionados a página *403 forbidden*.

4.2.2.5 Missing Anti-Clickjacking Header

Embora o cabeçalho para a prevenção de ataques de clickjacking não estivesse configurado, os testes feitos para a realização deste tipo de ataque nos mostraram que não é possível abrir o *Moodle* em um *iframe*. Entretanto, para realizar a remoção deste alerta, é necessário adicionar a seguinte linha no arquivo de configuração do servidor:

```
Header set X-Frame-Options:''sameorigin''
```

4.2.2.6 .htaccess Information Leak

Neste alerta, foi observado que o arquivo *.htaccess* estava exposto a todos os usuários na URL identificada pelo OWASP ZAP. Ao contrário da configuração feita para a mitigação do *composer.lock* para o *.htaccess* a configuração que melhor resolveu este alerta foi a seguinte:

```
RedirectMatch 404 /\\.htaccess
```

Com isso, qualquer usuário que tentar acessar um arquivo *.htaccess* será redirecionado a uma página de erro 404.

4.2.3 Vulnerabilidades de Risco Baixo

4.2.3.1 Big Redirect Detected

Este alerta foi causado pelo tamanho da resposta HTTP vindo do servidor Web. Isso indica que o servidor está respondendo com um redirecionamento, porém, adicionando outras informações no corpo da resposta. Para remover este alerta, as respostas HTTP foram analisadas na tentativa de identificar falhas ou informações sensíveis dos usuários. Também, investigou-se os códigos fontes relacionados. Não foram identificadas informações sensíveis sendo expostas.

4.2.3.2 Cookie without SameSite Attribute e Cookie No HTTPOnly Flag

Esse alerta indica que o *cookie* não possui os atributos de SameSite e No HTTPOnly definidos. Para corrigir o alerta, é necessário adicionar a seguinte linha no arquivo de configuração do servidor:

```
Set-Cookie: HttpOnly; SameSite=Strict
```

4.2.3.3 Disclosure of Date and Time

Verificando esse alerta, foi investigado que grandes sites deixam visível o dia e a hora como resposta HTTP, com isso, chegou-se a conclusão de que essa resposta HTTP não gera um risco de segurança para a aplicação.

4.2.3.4 Server Leaks Version Information

Verificando as respostas de requisição HTTP, verificamos que o cabeçalho Server estava retornando a versão do servidor, para solucionar este problema, foi adicionada a seguinte linha no arquivo de configuração do servidor:

```
ServerSignature Off
ServerTokens Prod
```

4.2.3.5 Strict-Transport-Security Header Not Set

Foi observado que de fato não havia o cabeçalho Strict-Transport-Security nas respostas HTTP, então, foi realizada a configuração do cabeçalho no arquivo de configuração do apache para realizar a remoção deste alerta. A configuração foi feita adicionando a seguinte linha no arquivo de configuração do servidor:

```
Header always set "Strict-Transport-Security"
"max-age=31536000" env=HTTPS
```

4.2.4 Motivos Informativos

4.2.4.1 Information Disclosure - Suspicious Comments

O aviso "*Suspicious Comments*"que busca identificar a presença de comentários que possam indicar informações sensíveis sobre a aplicação. No código fornecido, não foram encontrados comentários claros que possam indicar problemas de segurança. Na suspeita de um falso positivo, é possível ignorar o alerta ou revisar as configurações do OWASP ZAP para desativar especificamente esse alerta.

4.2.4.2 Information Disclosure - Sensitive Information in URL

Esse alerta indica que o *testsession* está sendo passado pela URL e que pode ser um conteúdo sensível. Esse parâmetro apenas segue para identificar o *id* do usuário que está realizando o *login* na plataforma. Entretanto, ao tentar realizar a troca desse valor para um outro, a própria plataforma já lida com isso, pedindo para o usuário realizar o *logout* e realizar o *login* novamente para acessar o outro id.

4.2.4.3 Modern Web Application

Nenhuma medida precisa ser tomada em relação a esse alerta, pois ele só serve para informar que o site está usando uma aplicação Web moderna.

4.2.4.4 Re-examine Cache-Control Directives

As diretivas de Cache-control ajudam a controlar como o conteúdo é armazenado em cache e atualizado pelos navegadores e outros clientes da Web. Algumas das diretivas comuns incluem "no-cache", "no-store", "max-age"e "must-revalidate". Ao usar essas diretivas, você pode garantir que o conteúdo seja atualizado regularmente e não seja armazenado em cache por muito tempo. No entanto, a diretiva que define o tempo máximo já está definida:

```
Header set Cache-Control "max-age=3600, public"
```

Contemplando assim um falso positivo.

4.2.4.5 User Agent Fuzzer

Este alerta indica que estamos utilizando o OWASP ZAP, no qual modifica o cabeçalho do usuário (User-Agent). Dessa forma não há uma correção específica para o alerta visto que ele não indica uma falha de segurança real. Uma maneira de reduzir a probabilidade dos alertas é usar um cabeçalho de usuário padrão na aplicação, ao invés de permitir que os usuários personalizem seu próprio cabeçalho de usuário. Também é

possível limitar o número de solicitações por segundo de um único endereço IP para impedir ataque de força bruta.

4.2.4.6 User Controllable HTML Element Attribute

Este alerta, foi gerado por conta do parâmetro *lang* que ocorre na URL do site. Esse parâmetro no entanto serve para modificar o idioma do site, mudando assim o atributo HTML do site. Para verificar se haveria um risco potencial desse parâmetro ocasionar uma falha, alguns testes foram feitos para tentar reproduzir um ataque de XSS. Os testes feitos incluíram dois tipos de *scripts*:

```
?lang=\textbf{<script>alert(1)</script}
?lang=\textbf{\%27"</script><script>alert(document.cookie)</script>}
```

Em ambos os testes, não houve um efeito de *trigger* dos alertas, então, pode-se concluir que, embora o usuário consiga "controlar"o atributo HTML do site, ele não apresenta riscos para a segurança do site e dos outros usuários.

4.3 Etapa 3 - Detecção de Ameaças

A análise dos registros gerados, ou seja, os *logs*, evidenciou que certos ataques resultam na criação de *logs* nas três diferentes fontes mencionadas na Seção 3.3.3. Por outro lado, outros ataques resultam em registros apenas no *Apache*, o que aponta que a plataforma *Moodle* não implementa mecanismos adequados para a auditoria desses eventos potencialmente maliciosos.

Não obstante, foi observado que nem todos os *logs* relacionados a atividades maliciosas são registrados no *logs* de erros do *Apache* (`error.log`). Da mesma forma, em geral, a análise isolada do arquivo que registra os *logs* de acesso do *Apache* (`access.log`) não fornece informações suficientemente significativas sobre a ocorrência de um ataque. Contudo, quando se consideram ambos os arquivos em conjunto com os *logs* do *Moodle*, é possível obter uma percepção mais aprofundada e precisa do que realmente está acontecendo.

Durante a análise executada neste trabalho, tomou-se a decisão de não levar em consideração o campo *User-Agent* da requisição. A motivação por trás disso é que as ferramentas de ataque geralmente inserem seu próprio nome neste campo, o que seria um indicador explícito da ação maliciosa. Por outro lado, elas também permitem a personalização deste campo, o que torna difícil rastrear a origem do ataque com base apenas nele. Além disso, muitos *firewalls*, sistemas de detecção de intrusão e sistemas de prevenção de intrusão utilizam listas negras, ou seja, *blacklists*, e geram um alerta ou bloqueiam o

tráfego assim que recebem uma requisição contendo o nome de uma ferramenta de ataque conhecida. Logo, a ênfase na análise dos *logs* deve ser a interação entre eles e não a confiança única em elementos facilmente manipuláveis, como o *User-Agent*.

4.3.1 Analise dos Logs de *Bruteforce* no login

Algumas abordagens podem ser utilizadas para a identificação de tentativas de ataques de força bruta em uma aplicação Web. No caso deste trabalho, utilizou-se a abordagem de correlação de eventos entre as três fontes mencionadas na Seção 3.3.3.

Analisando as Figuras 2, 3 e 4, torna-se evidente que o endereço IP `117.106.230.166` está executando uma série de tentativas de login sem sucesso em um intervalo de tempo bastante curto. O campo *Epoch*, presente nas figuras exemplificativas dos *logs* de erro (*error.log*, Figura 3) e do *Moodle* (Figura 4), demonstra que foram realizadas seis solicitações em um intervalo de tempo potencialmente medido em milissegundos. Tal comportamento é atípico para um usuário humano.

Outra peculiaridade observada é a mudança constante do nome de usuário a cada tentativa de login. Tal comportamento poderia ser considerado normal caso os nomes de usuários fossem similares ou seguissem um padrão, porém os nomes testados exibem variações substanciais. Esse tipo de atividade reforça a hipótese de um ataque de força bruta.

IP	Requisição	Epoch	Código HTTP
177.106.230.166	GET /moodle/login/index.php HTTP/1.0	1685030756	200
177.106.230.166	GET /moodle/login/index.php HTTP/1.0	1685030756	200
177.106.230.166	GET /moodle/login/index.php HTTP/1.0	1685030756	200
177.106.230.166	GET /moodle/login/index.php HTTP/1.0	1685030756	200
177.106.230.166	GET /moodle/login/index.php HTTP/1.0	1685030756	200
177.106.230.166	GET /moodle/login/index.php HTTP/1.0	1685030756	200
177.106.230.166	GET /moodle/login/index.php HTTP/1.0	1685030756	200
177.106.230.166	GET /moodle/login/index.php HTTP/1.0	1685030756	200

Figura 2 – Registro no access.log de tentativa de Acesso por Força Bruta.

Client-ID	URL	Error message	Epoch
[client 177.106.230.166]	http://ec2-54-227-153-250.compute-1.amazonaws.com/moodle	Invalid Login Token: robert	1685030756
[client 177.106.230.166]	http://ec2-54-227-153-250.compute-1.amazonaws.com/moodle	Invalid Login Token: null	1685030756
[client 177.106.230.166]	http://ec2-54-227-153-250.compute-1.amazonaws.com/moodle	Invalid Login Token: michael	1685030756
[client 177.106.230.166]	http://ec2-54-227-153-250.compute-1.amazonaws.com/moodle	Invalid Login Token: david	1685030756
[client 177.106.230.166]	http://ec2-54-227-153-250.compute-1.amazonaws.com/moodle	Invalid Login Token: mike	1685030756
[client 177.106.230.166]	http://ec2-54-227-153-250.compute-1.amazonaws.com/moodle	Invalid Login Token: 2000	1685030756
[client 177.106.230.166]	http://ec2-54-227-153-250.compute-1.amazonaws.com/moodle	Invalid Login Token: info	1685030757
[client 177.106.230.166]	http://ec2-54-227-153-250.compute-1.amazonaws.com/moodle	Invalid Login Token: dave	1685030757

Figura 3 – Registro no error.log de tentativa de Acesso por Força Bruta.

177.106.230.166	coreeventuser_login_failed	user_login	{username":"david"}	1685030756	failed
177.106.230.166	coreeventuser_login_failed	user_login	{username":"null"	1685030756	failed
177.106.230.166	coreeventuser_login_failed	user_login	{username":"mike"	1685030756	failed
177.106.230.166	coreeventuser_login_failed	user_login	{username":"robert"	1685030756	failed
177.106.230.166	coreeventuser_login_failed	user_login	{username":"michael"	1685030756	failed
177.106.230.166	coreeventuser_login_failed	user_login	{username":"2000"	1685030756	failed
177.106.230.166	coreeventuser_login_failed	user_login	{username":"dave"	1685030757	failed
177.106.230.166	coreeventuser_login_failed	user_login	{username":"richard"	1685030757	failed

Figura 4 – Registro no Moodle de tentativa de Acesso por Força Bruta.

4.3.2 Analise dos Logs de Listagem de Diretórios

Uma maneira de se identificar uma tentativa de listagem de diretórios através da análise manual é verificando as requisições feitas ao servidor utilizando o arquivo que registra os *logs* de acesso do *Apache* (`access.log`) mencionado na Seção 3.3.3, procurando por tentativas de acessos a diretórios incomuns ou suspeitos, assim como os respectivos códigos HTTP para essas tentativas.

Nas Figuras 5 e 6 podemos ver os *logs* de uma possível listagem de diretórios através do arquivo de acesso do *Apache* (`access.log`) e dos *logs* de erro do *Apache* (`error.log`). Observando a Figura 5 foi destacado em vermelho e em azul claro duas requisições no qual uma delas acabou por gerar um *log* de erro. A diferença entre esses dois, se da pelo fato de que a requisição destacada em vermelho tenta acessar uma página chamada *privacy* e a requisição destacada em azul claro tenta acessar um diretório chamado *privacy*. No primeiro caso o Apache interpretou como uma tentativa normal de acesso, resultando em um redirecionamento (código HTTP 301) e não registrando-o no arquivo de *log* de erro. Já no segundo caso ocorreu um código HTTP 403 *Forbidden*, no qual o Apache interpretou como um erro, registrando essa requisição no arquivo de *log* de erro *error.log*.

Através dos *logs* de acesso *access.log* é possível observar uma quantidade muito alta de requisições para diferentes diretórios vindo de um mesmo endereço de IP, isso também é um indício de que o usuário está fazendo uma listagem de diretório para ganhar informações sobre a aplicação, essa listagem de diretório juntamente com a listagem de arquivos são as primeiras etapas para a realização de um possível ataque.

Uma observação que pode ser feita é que a tentativa de acesso a um diretório resultar em um código HTTP 403 *Forbidden* significando que o servidor está configurado para evitar listagem de diretórios, o que acaba por aumentar a segurança do site.

4.3.3 Analise dos Logs de Listagem de Arquivos

A análise dos *logs* da listagem de arquivos parte do mesmo princípio que a análise dos *logs* da listagem de diretórios. Primeiramente identificam-se os *logs* relevantes no arquivo que registra os *logs* de acesso no *Apache* (`access.log` mencionado na Seção 3.3.3,

IP	Requisição	Código HTTP	Tamanho	Referer	Epoch
189.15.108.199	GET /moodle/privacy HTTP/1.1	301	503	no	1684954562
189.15.108.199	GET /moodle/search HTTP/1.1	301	501	no	1684954562
189.15.108.199	GET /moodle/11 HTTP/1.1	404	341	no	1684954562
189.15.108.199	GET /moodle/logo HTTP/1.1	404	341	no	1684954562
189.15.108.199	GET /moodle/blog HTTP/1.1	301	497	no	1684954562
189.15.108.199	GET /moodle/privacy/ HTTP/1.1	403	344	http://ec2no54nc	1684954562
189.15.108.199	GET /moodle/new HTTP/1.1	404	341	no	1684954563
189.15.108.199	GET /moodle/search/ HTTP/1.1	200	7808	http://ec2no54nc	1684954562
189.15.108.199	GET /moodle/10 HTTP/1.1	404	341	no	1684954563
189.15.108.199	GET /moodle/cginobin HTTP/1.1	404	341	no	1684954563
189.15.108.199	GET /moodle/faq HTTP/1.1	404	341	no	1684954563
189.15.108.199	GET /moodle/blog/ HTTP/1.1	303	2083	http://ec2no54nc	1684954563
189.15.108.199	GET /moodle/rss HTTP/1.1	301	495	no	1684954563

Figura 5 – Registro no access.log de tentativa de listagem de diretório.

ClientIPPorta	Error message	Epoch
[client 189.15.108.199:27587]	AH01276: Cannot serve directory /var/www/html/moodle/privacy/: No matching Dire	1684954562
[client 189.15.108.199:27585]	AH01276: Cannot serve directory /var/www/html/moodle/rss/: No matching Director	1684954563
[client 189.15.108.199:60270]	AH01276: Cannot serve directory /var/www/html/moodle/media/: No matching Direc	1684954564
[client 189.15.108.199:27585]	AH01276: Cannot serve directory /var/www/html/moodle/report/: No matching Direc	1684954574
[client 189.15.108.199:27587]	AH01276: Cannot serve directory /var/www/html/moodle/local/: No matching Directc	1684954575
[client 189.15.108.199:60272]	AH01276: Cannot serve directory /var/www/html/moodle/pix/: No matching Director	1684954575
[client 189.15.108.199:27591]	AH01276: Cannot serve directory /var/www/html/moodle/install/: No matching Direc	1684954584
[client 189.15.108.199:27593]	AH01276: Cannot serve directory /var/www/html/moodle/portfolio/: No matching Dir	1684954586
[client 189.15.108.199:60278]	AH01276: Cannot serve directory /var/www/html/moodle/cache/: No matching Direc	1684954594

Figura 6 – Registro no error.log de tentativa listagem de diretórios.

após isso passa-se a identificar os acessos incomuns e por fim analisar o código HTTP para cada um desses *logs* identificados.

No exemplo das Figuras 7 e 8 pode-se notar que o mesmo endereço IP está tentando uma série de acesso a diferentes arquivos da aplicação Web com uma diferença de milissegundos entre cada requisição, juntamente a essa quantidade alta de requisições pode-se notar a tentativa de acesso a arquivos que não possuem sentido ao contexto da aplicação do *Moodle* (*e.g router.php* e *backend.php*). Com base nas informações apontadas, pode-se afirmar que esse IP está fazendo uma tentativa de *listagem* de arquivos para coletar mais informações sobre a aplicação Web.

4.3.4 Análise dos Logs do OWASP ZAP

É necessário proceder com maior prudência, uma vez que os primeiros *logs* sugerem que um usuário comum está utilizando a plataforma, entretanto, assim como nas outras análises, muitas requisições foram feitas com uma diferença de milissegundos. Ademais, é importante considerar a seleção de atributos relevantes para maximizar as chances de detectar-se corretamente as ameaças (QUINCOZES et al., 2021b). Porém a análise só foi

IP	Requisição	Código HTTP	Tamanho	Epoch
189.15.108.199	GET /moodle/default.php HTTP/1.1	404.0	341.0	1684958740
189.15.108.199	GET /moodle/invocactf.php HTTP/1.1	404.0	341.0	1684958740
189.15.108.199	GET /moodle/frontend.php HTTP/1.1	404.0	341.0	1684958740
189.15.108.199	GET /moodle/view.php HTTP/1.1	404.0	341.0	1684958740
189.15.108.199	GET /moodle/controller.php HTTP/1.1	404.0	341.0	1684958740
189.15.108.199	GET /moodle/router.php HTTP/1.1	404.0	341.0	1684958740
189.15.108.199	GET /moodle/backend.php HTTP/1.1	404.0	341.0	1684958740
189.15.108.199	GET /moodle/colorConfig.ini.php HTTP/1.1	404.0	341.0	1684958740
189.15.108.199	GET /moodle/helper.php HTTP/1.1	404.0	341.0	1684958740
189.15.108.199	GET /moodle/config.php HTTP/1.1	200.0	373.0	1684958740
189.15.108.199	GET /moodle/view.html.php HTTP/1.1	404.0	341.0	1684958740
189.15.108.199	GET /moodle/addadminuser.php HTTP/1.1	404.0	341.0	1684958740
189.15.108.199	GET /moodle/menu.php HTTP/1.1	404.0	341.0	1684958740
189.15.108.199	GET /moodle/search.php HTTP/1.1	404.0	341.0	1684958740
189.15.108.199	GET /moodle/adminlogin.php HTTP/1.1	404.0	341.0	1684958740
189.15.108.199	GET /moodle/user.php HTTP/1.1	404.0	341.0	1684958740
189.15.108.199	GET /moodle/admin.php HTTP/1.1	404.0	341.0	1684958740

Figura 7 – Registro no access.log de tentativa de listagem de arquivos

ClientIP-Porta	Error message	Epoch
[client 189.15.108.199:62264]	script '/var/www/html/moodle/default.php' not found or unable to stat	1684958740
[client 189.15.108.199:28525]	script '/var/www/html/moodle/invocactf.php' not found or unable to stat	1684958740
[client 189.15.108.199:62266]	script '/var/www/html/moodle/frontend.php' not found or unable to stat	1684958740
[client 189.15.108.199:28527]	script '/var/www/html/moodle/view.php' not found or unable to stat	1684958740
[client 189.15.108.199:28531]	script '/var/www/html/moodle/controller.php' not found or unable to stat	1684958740
[client 189.15.108.199:62268]	script '/var/www/html/moodle/router.php' not found or unable to stat	1684958740
[client 189.15.108.199:62272]	script '/var/www/html/moodle/backend.php' not found or unable to stat	1684958740
[client 189.15.108.199:62270]	script '/var/www/html/moodle/colorConfig.ini.php' not found or unable to stat	1684958740
[client 189.15.108.199:28533]	script '/var/www/html/moodle/helper.php' not found or unable to stat	1684958740
[client 189.15.108.199:28525]	script '/var/www/html/moodle/view.html.php' not found or unable to stat	1684958740
[client 189.15.108.199:62268]	script '/var/www/html/moodle/addadminuser.php' not found or unable to stat	1684958740
[client 189.15.108.199:28527]	script '/var/www/html/moodle/menu.php' not found or unable to stat	1684958740
[client 189.15.108.199:62266]	script '/var/www/html/moodle/search.php' not found or unable to stat	1684958740
[client 189.15.108.199:28529]	script '/var/www/html/moodle/adminlogin.php' not found or unable to stat	1684958740
[client 189.15.108.199:62270]	script '/var/www/html/moodle/user.php' not found or unable to stat	1684958740
[client 189.15.108.199:28531]	script '/var/www/html/moodle/admin.php' not found or unable to stat	1684958740

Figura 8 – Registro no error.log de tentativa de listagem de arquivos

possível correlacionando os três arquivos de *logs* (*i.e.,* `access.log` (Figura 9), `error.log` (Figura 10) e os *logs* do *Moodle* (Figura 11)) mencionados na Seção 3.3.3. Isso ocorre porque no `access.log` apenas aparecem requisições *POST* para a página de *login* da plataforma, com isso não da para se ter uma ideia se foi uma tentativa comum ou se foi uma tentativa maliciosa, entretanto correlacionando os arquivos e através do *timestamp,* foi possível notar que o usuário estava manipulando os parâmetros da requisição *POST* inserindo comandos para tentar acessar arquivos do servidor, como pode ser visto na

última linha das Figuras 10 e 11.

IP	Requisição	Código HTTP	Epoch	I
177.106.230.166	POST /moodle/login/index.php HTTP/1.1	200	1685033978	I
177.106.230.166	POST /moodle/login/signup.php HTTP/1.1	404	1685033978	I
177.106.230.166	POST /moodle/login/index.php HTTP/1.1	303	1685033978	I
177.106.230.166	POST /moodle/login/signup.php HTTP/1.1	404	1685033978	I
177.106.230.166	POST /moodle/login/index.php HTTP/1.1	200	1685033978	I
177.106.230.166	POST /moodle/login/signup.php HTTP/1.1	404	1685033978	I
177.106.230.166	POST /moodle/login/index.php HTTP/1.1	303	1685033979	I
177.106.230.166	POST /moodle/login/signup.php HTTP/1.1	404	1685033979	I
177.106.230.166	POST /moodle/login/index.php HTTP/1.1	200	1685033979	I
177.106.230.166	POST /moodle/login/signup.php HTTP/1.1	404	1685033979	I
177.106.230.166	POST /moodle/login/index.php HTTP/1.1	303	1685033979	I
177.106.230.166	POST /moodle/login/signup.php HTTP/1.1	404	1685033979	I
177.106.230.166	POST /moodle/login/index.php HTTP/1.1	200	1685033979	I
177.106.230.166	POST /moodle/login/signup.php HTTP/1.1	404	1685033979	I
177.106.230.166	POST /moodle/login/index.php HTTP/1.1	303	1685033979	I
177.106.230.166	POST /moodle/login/signup.php HTTP/1.1	404	1685033979	I
177.106.230.166	POST /moodle/login/index.php HTTP/1.1	200	1685033979	I
177.106.230.166	POST /moodle/login/signup.php HTTP/1.1	404	1685033980	I
177.106.230.166	POST /moodle/login/index.php HTTP/1.1	303	1685033980	I

Figura 9 – Registro de access.log

ClientIP-Porta	Error message	Epoch
[client 177.106.230.166:50908]	Invalid Login Token: zap	1685033978
[client 177.106.230.166:50908]	Invalid Login Token: zap	1685033978
[client 177.106.230.166:50908]	Invalid Login Token: zap	1685033978
[client 177.106.230.166:50908]	Invalid Login Token: zap	1685033978
[client 177.106.230.166:50908]	Invalid Login Token: zap	1685033979
[client 177.106.230.166:50908]	Invalid Login Token: zap	1685033979
[client 177.106.230.166:50908]	Invalid Login Token: zap	1685033979
[client 177.106.230.166:50908]	Invalid Login Token: zap	1685033979
[client 177.106.230.166:50908]	Invalid Login Token: zap	1685033979
[client 177.106.230.166:50908]	Invalid Login Token: cat /etc/passwd	1685033979

Figura 10 – Registro de error.log

4.3.5 Considerações Finais da Detecção de Ameaças

É importante ressaltar que as análises de listagem de diretórios, listagem de arquivos e dos *logs* do OWASP ZAP baseiam-se também em uma análise de contexto que irá variar para cada aplicação Web. Com isso, para identificar esses tipos de ataques, é

177.106.230.166	\\core\\event\\user_login_failed	failed	user_login	{\username\":\"zap\"	1685033978
177.106.230.166	\\core\\event\\user_login_failed	failed	user_login	{\username\":\"zap\"	1685033978
177.106.230.166	\\core\\event\\user_login_failed	failed	user_login	{\username\":\"zap\"	1685033978
177.106.230.166	\\core\\event\\user_login_failed	failed	user_login	{\username\":\"zap\"	1685033978
177.106.230.166	\\core\\event\\user_login_failed	failed	user_login	{\username\":\"zap\"	1685033979
177.106.230.166	\\core\\event\\user_login_failed	failed	user_login	{\username\":\"zap\"	1685033979
177.106.230.166	\\core\\event\\user_login_failed	failed	user_login	{\username\":\"zap\"	1685033979
177.106.230.166	\\core\\event\\user_login_failed	failed	user_login	{\username\":\"zap\"	1685033979
177.106.230.166	\\core\\event\\user_login_failed	failed	user_login	{\username\":\"zap\"	1685033979
177.106.230.166	\\core\\event\\user_login_failed	failed	user_login	{\username\":\"cat \\/etc\\/passwd\"	1685033979

Figura 11 – Registro dos *logs* do *Moodle*

necessário compreender se a tentativa de acesso a um arquivo ou a um diretório faz sentido para a aplicação.

Uma atenção também deve ser dada ao endereço de IP que está enviando as requisições, pois com o uso de redes virtuais privadas, do inglês *Virtual Private Network* (VPN) pode ser que as diversas requisições maliciosas venham de diferentes endereços IP, por conta disso uma análise mais cautelosa deve ser feita.

Outro ponto importante a se destacar é que a análise manual de *logs* deve ser reservada para situações em que haja dúvidas sobre determinado comportamento. Atualmente, muitos sistemas de detecção de intrusão são capazes de realizar essa análise de forma automática, o que permite identificar e gerar alertas sobre possíveis ataques de maneira mais eficiente.

Esses sistemas automatizados de detecção de intrusão utilizam algoritmos avançados e modelos de *machine learning* para analisar os *logs* em busca de padrões e anomalias. Ao comparar os registros de atividades com perfis de comportamento normal, essas ferramentas conseguem identificar indícios de atividades maliciosas de forma ágil e precisa.

Ao automatizar a análise de *logs*, é possível reduzir significativamente o tempo necessário para detectar possíveis ameaças e responder a elas de forma proativa. Além disso, a análise automática dos *logs* pode ajudar a minimizar a carga de trabalho dos analistas de segurança, permitindo que eles se concentrem em investigações mais complexas e tarefas estratégicas.

No entanto, é importante ressaltar que a análise manual ainda tem seu valor em certos cenários. Em casos de incidentes graves ou suspeitas complexas, os analistas de segurança podem precisar examinar os *logs* de forma mais detalhada, realizando uma análise forense para compreender a extensão do incidente e tomar medidas adequadas.

5 Conclusões

Este trabalho apresentou um estudo de caso acerca das potenciais vulnerabilidades da lista *OWASP Top 10* presente no ambiente virtual de ensino e aprendizagem *Moodle* por meio da ferramenta OWASP ZAP. A plataforma *Moodle* é utilizada por mais de 316 milhões de usuários e com o aumento de ataques cibernéticos é de extrema importância verificar o nível de segurança da plataforma ao ser instalada por um usuário comum.

Portanto esse estudo mostrou que a plataforma *Moodle*, ao ser instalada direto do repositório está sujeita a diversas vulnerabilidades, através das quais um invasor pode se aproveitar para efetuar ataques na aplicação. Tal análise foi realizada utilizando o *scan* automático da ferramenta OWASP ZAP, o qual gerou um total de 894 alertas divididos entre 20 vulnerabilidades como visto na tabela 2.

As recomendações apresentadas neste trabalho visam proteger o *Moodle* contra possíveis ataques, garantindo a segurança dos dados e informações dos usuários. É importante ressaltar que a segurança em ambientes virtuais de aprendizagem é um tema cada vez mais relevante, especialmente em um contexto em que o ensino remoto se tornou uma realidade para muitas instituições educacionais.

Ademais, partindo da premissa que ataques podem acontecer e sua detecção é fundamental, apresentamos uma análise dos *logs* de acesso do *Moodle*, que podem ser utilizados para monitorar possíveis tentativas de ataques cibernéticos e prevenir vulnerabilidades em potencial. Ao examinar esses *logs*, conseguimos identificar padrões de acesso suspeitos, permitindo a adoção de medidas preventivas para garantir a segurança da plataforma.

Por fim, espera-se que este estudo contribua para a conscientização sobre a importância da segurança em ambientes virtuais de aprendizagem e incentive a adoção de medidas preventivas para garantir a proteção dos dados e informações dos usuários, incluindo a análise regular dos *logs* de acesso do *Moodle* como parte integrante das estratégias de segurança cibernética.

Como trabalhos futuros, pretende-se utilizar outras ferramentas de *scan* de vulnerabilidades (*e.g., OpenVAS*, *Nessus*, *Nikto*, entre outras) para comparar os resultados obtidos utilizando o ZAP e assim conseguir verificar qual possui maior eficiência para detectar vulnerabilidades na aplicação e assim chegar em uma configuração de segurança do *Moodle* que seja eficiente na mitigação e prevenção de potenciais vulnerabilidades. Pretende-se também utilizar os *logs* gerados pelas ferramentas como um *dataset* de entrada para *Machine Learning* e assim desenvolver um *plug-in* para a plataforma que realize essas detecções e gere alertas para os administradores.

Referências

ABEINFO. **Ataques cibernéticos no Brasil aumentaram 46% no segundo trimestre de 2022.** 2022. <https://abeinfobrasil.com.br/ataques-ciberneticos-no-brasil-aumentaram-46-no-segundo-trimestre-de-2022>. Citado na página 12.

AHAMED, A.; SADMAN, N.; KHAN, T. A.; HANNAN, M. I.; SADIA, F.; HASAN, M. Automated testing: Testing top 10 owasp vulnerabilities of government web applications in bangladesh. In: ICSEA. **The Seventeenth International Conference on Software Engineering Advances**. Bangladesh: IARIA, 2022. p. 1–7. Citado 2 vezes nas páginas 22 e 23.

ASSUNÇÃO, M. **Guia do Hacker Brasileiro**. Marcos Flávio Araújo Assunção, 2002. ISBN 9788575020838. Disponível em: <https://books.google.com.br/books?id=s2ZZAwAAQBAJ>. Citado na página 16.

CHEATSHEETS-SERIES. **OWASP Cheat Sheet Series: SQL Injection Prevention**. 2022. Disponível em: <https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html>. Acesso em: 9 dez. 2022. Citado na página 32.

CLOUTIER, R.; MULLER, G.; VERMA, D.; NILCHIANI, R.; HOLE, E.; BONE, M. The concept of reference architectures. **Systems Engineering**, Wiley Online Library, v. 13, n. 1, p. 14–27, 2010. Citado na página 18.

CUERVO, M. C.; ALDANA, A. A.; LÓPEZ, A. B. Security evaluation model for virtual learning environments. In: IEEE. **2016 XI Latin American Conference on Learning Objects and Technology (LACLO)**. San Carlos, Costa Rica: IEEE, 2016. p. 1–6. Citado 3 vezes nas páginas 13, 23 e 24.

DEEPA, G.; THILAGAM, P. S. Securing web applications from injection and logic vulnerabilities: Approaches and challenges. **Information and Software Technology**, Elsevier, v. 74, p. 160–180, 2016. Citado na página 18.

DOUGIAMAS, M. **Aprendizado on-line com o LMS mais popular do mundo - Moodle**. 2022. Disponível em: <https://moodle.com/pt/>. Acesso em: 25 set. 2022. Citado 2 vezes nas páginas 12 e 21.

FORTINET. **CIA Triad**. 2023. <https://www.fortinet.com/resources/cyberglossary/cia-triad#:~:text=The%20three%20letters%20in%20%22CIA,and%20methods%20for%20creating%20solutions.> Acesso em: 29 jul. 2023. Citado na página 15.

FREDJ, O. B.; CHEIKHROUHOU, O.; KRICHEN, M.; HAMAM, H.; DERHAB, A. An owasp top ten driven survey on web application protection methods. In: SPRINGER. **International Conference on Risks and Security of Internet and Systems**. New York, NY, USA: Springer, Cham, 2020. p. 235–252. Citado na página 13.

HASSAN, M.; ALI, M.; BHUIYAN, T.; SHARIF, M.; BISWAS, S. Quantitative assessment on broken access control vulnerability in web applications. In: **International**

Conference on Cyber Security and Computer Science 2018. Safranbolu, Turkey: ICONCS, 2018. Citado na página 17.

HERNANDEZ, J. C. G.; CHAVEZ, M. A. L. Moodle security vulnerabilities. In: IEEE. **2008 5th International Conference on Electrical Engineering, Computing Science and Automatic Control**. Mexico City, Mexico: IEEE, 2008. p. 352–357. Citado 3 vezes nas páginas 13, 23 e 24.

HUMAYUN, M.; NIAZI, M.; JHANJHI, N.; ALSHAYEB, M.; MAHMOOD, S. Cyber security threats and vulnerabilities: a systematic mapping study. **Arabian Journal for Science and Engineering**, Springer, v. 45, n. 4, p. 3171–3189, 2020. Citado 2 vezes nas páginas 16 e 17.

JABIYEV, B.; MIRZAEI, O.; KHARRAZ, A.; KIRDA, E. Preventing server-side request forgery attacks. In: **Proceedings of the 36th Annual ACM Symposium on Applied Computing**. New York, NY, United States: Association for Computing Machinery, 2021. p. 1626–1635. Citado na página 20.

KASPERSKY. **Kaspersky Security Bulletin 2021 Statistics**. 2021. <https://go.kaspersky.com/rs/802-IJN-240/images/KSB_statistics_2021_eng.pdf>. Citado na página 13.

MITRE. **Common Weakness Enumeration (CWE): Overview**. 2022. Disponível em: <https://cwe.mitre.org/about/index.html>. Acesso em: 25 nov. 2022. Citado na página 17.

MONTEVERDE, W. A. **Estudo e análise de vulnerabilidades web**. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2014. Citado 2 vezes nas páginas 22 e 23.

MUDIYANSELAGE, A. K.; PAN, L. Security test moodle: a penetration testing case study. **International Journal of Computers and Applications**, Taylor & Francis, v. 42, n. 4, p. 372–382, 2020. Citado 5 vezes nas páginas 13, 16, 17, 23 e 24.

OLADIMEJI, S.; KERNER, S. M. **SolarWinds hack explained: Everything you need to know**. 2022. <https://www.techtarget.com/whatis/feature/SolarWinds-hack-explained-Everything-you-need-to-know>. Citado na página 20.

OWASP. **OWASP Top 10:2021**. 2022. Disponível em: <https://owasp.org/www-project-top-ten/>. Acesso em: 25 set. 2022. Citado 3 vezes nas páginas 12, 17 e 18.

_____. **OWASP Zed Attack Proxy (ZAP)**. 2022. Disponível em: <https://www.zaproxy.org/>. Acesso em: 25 set. 2022. Citado 4 vezes nas páginas 16, 19, 21 e 32.

_____. **OWASP Zed Attack Proxy (ZAP) - Alerts Documentation**. 2022. Disponível em: <https://www.zaproxy.org/docs/alerts/>. Acesso em: 25 set. 2022. Citado 3 vezes nas páginas 33, 34 e 35.

PEDRA, D. **O que é uma ameaça em segurança da informação? Como calcular o seu impacto?** 2022. Disponível em: <https://www.siteware.com.br/seguranca/o-que-e-uma-ameaca-em-seguranca-da-informacao/#:~:text=J%C3%A1%20a%20vulnerabilidade%20na%20seguran%C3%A7a,amea%C3%A7as%20%C3%A0%20seguran%C3%A7a%20da%20informa%C3%A7%C3%A3o.>. Acesso em: 23 jan. 2023. Citado na página 16.

PRIYAWATI, D.; ROKHMAH, S.; UTOMO, I. C. Website vulnerability testing and analysis of website application using owasp. **International Journal of Computer and Information System (IJCIS)**, v. 3, n. 3, p. 142–147, 2022. Citado 3 vezes nas páginas 13, 22 e 23.

QUINCOZES, S. E.; ALBUQUERQUE, C.; PASSOS, D.; MOSSÉ, D. A survey on intrusion detection and prevention systems in digital substations. **Computer Networks**, Elsevier, v. 184, p. 107679, 2021. Citado na página 28.

QUINCOZES, S. E.; MOSSÉ, D.; PASSOS, D.; ALBUQUERQUE, C.; OCHI, L. S.; SANTOS, V. F. dos. On the performance of grasp-based feature selection for cps intrusion detection. **IEEE Transactions on Network and Service Management**, IEEE, v. 19, n. 1, p. 614–626, 2021. Citado na página 44.

SAMPAIO, F. F. **Uma análise prática das principais vulnerabilidades em aplicações web baseado no top 10 OWASP**. 61 f. Tese (Bacharelado) — Universidade Federal do Paraná (UFC), Paraná, 2021. Citado 3 vezes nas páginas 13, 22 e 23.

SILVA, R. R. T.; LIMA, R. W. de; LEITE, C. R. M.; SILVA, R. R. T. da. Investigação de segurança no moodle. **RENOTE**, v. 12, n. 2, p. 61, 2014. Citado 2 vezes nas páginas 23 e 24.

STALLINGS, W. **Criptografia e segurança de redes. São Paulo SP**. [S.l.]: Pearson Prentice Hall, 2008. Citado na página 15.

Anexos

ANEXO A – Artigo fruto deste trabalho, o qual foi submetido ao SBSeg 2023.

A Plataforma Moodle está Segura? Uma Análise das Vulnerabilidades do Top 10 da OWASP

1

Abstract. *The Moodle platform consists of a web application with more than 316 million users, who entrust their personal and academic information to institutions that provide materials and apply assessments through this platform. In a scenario where cybersecurity is a worldwide concern, with an increase of 32% in attacks in 2022, the following question arises: Is the Moodle platform a secure platform? To answer this question, this paper presents a case study based on the main vulnerabilities that affect current web applications, according to the "Top 10 OWASP" list of 2021. Our investigation using the Zed Attack Proxy (ZAP) tool identified 894 alerts of potential vulnerabilities. For each of them, we present the proper countermeasures.*

Resumo. *A plataforma Moodle consiste em uma aplicação web usada por mais de 316 milhões de usuários, os quais confiam suas informações pessoais e acadêmicas às instituições que disponibilizam materiais e aplicam avaliações por meio dessa plataforma. Em um cenário onde a segurança cibernética é uma preocupação em nível mundial, com um aumento de 32% de ataques em 2022, cabe o questionamento: A plataforma Moodle está segura? Para responder essa pergunta, este trabalho apresenta um estudo de caso calcado nas principais vulnerabilidades que atingem as aplicações web atuais, de acordo com a lista "Top 10 OWASP", de 2021. Tal análise, baseada na ferramenta Zed Attack Proxy (ZAP), identificou 894 alertas de potenciais vulnerabilidades para as quais se propõem as devidas contramedidas.*

1. Introdução

O Projeto Aberto de Segurança em Aplicações Web, do inglês, *Open Web Application Security Project* (OWASP) [OWASP 2022a] consiste em uma comunidade internacional sem fins lucrativos que tem por finalidade a produção de artigos, metodologias, documentações, ferramentas e tecnologias no campo de segurança de aplicativos web. Tal organização disponibiliza regularmente uma lista das dez principais vulnerabilidades, denominada *OWASP Top 10* [OWASP 2022a]. O *OWASP Top 10* serve para conscientizar os desenvolvedores sobre os riscos mais críticos em uma aplicação web [Santos et al. 2019].

A segurança cibernética é uma preocupação de nível mundial. Os dados recentes demonstram crescimento na quantidade de ocorrências de ataques cibernéticos. No 2° trimestre de 2022, houve um aumento de 32% nos ataques cibernéticos globais comparado ao 2° trimestre de 2021, onde ocorreu uma média de 1200 ataques semanais por organização. Já no Brasil, no mesmo período, os ataques aumentaram em 46% comparado ao período do 2° trimestre de 2021 [Schendes 2022]. Dentre os alvos desses ataques, destacam-se aqueles direcionados às aplicações web. De acordo com o relatório

da [Kaspersky 2021] durante o ano de 2021, 15,45% dos computadores de usuários da internet global sofreram algum tipo de ataque.

Atualmente na literatura há trabalhos que apresentam esforços direcionados ao estudo das principais vulnerabilidades do *OWASP Top 10* [Priyawati et al. 2022] [Sampaio 2021] [Fredj et al. 2020]. No entanto, a maioria desses estudos concentram-se em versões anteriores, tal como o *OWASP Top 10* divulgado em 2017 —- que pode ser considerado desatualizado, uma vez que em setembro de 2021 foi publicada uma relação atualizada das principais vulnerabilidades. A versão do *OWASP Top 10* de 2021 apresenta, por exemplo, o agrupamento de algumas vulnerabilidades e o aparecimento de outras. Ademais, os trabalhos cuja temática é o *OWASP Top 10* não abordam o estudo das ferramentas e implementações práticas utilizadas por atacantes para explorar as vulnerabilidades discutidas. Portanto, faltam estudos atualizados acerca dos grupos de vulnerabilidades que mais tem ocorrido nas aplicações web, bem como seus impactos nos sistemas atuais mormente as plataformas virtuais educacionais as quais tiveram o seu uso intensificado durante a pandemia de COVID-19.

A seguir, são sumarizados contribuições e estrutura deste trabalho. Na Seção 2, apresenta-se um levantamento de aplicações web existentes na literatura seguido de uma investigação das principais vulnerabilidades presentes na plataforma Moodle. Para a investigação, utilizou-se como base a lista das dez principais vulnerabilidades apontadas pela última versão do *ranking OWASP Top 10* de 2021 [OWASP 2022a]. As principais contribuições deste trabalho consistem em (i) analisar os alertas gerados pela ferramenta OWASP ZAP, uma ferramenta de *scan* de vulnerabilidades proposta pela própria organização OWASP, e (ii) recomendar contramedidas para as vulnerabilidades em potencial encontradas. Com base em tal análise, é desenvolvido um cenário de experimentações, descrito na Seção 3. Na Seção 4, é apresentado um estudo de caso que emprega a ferramenta supracitada na avaliação do ambiente virtual de ensino e aprendizagem Moodle, uma aplicação web de ensino que é utilizada por mais de 316 milhões de usuários [Dougiamas 2022]. Na Seção 5, apresenta-se a validação e correção de alertas. Por fim, na Seção 6, são apresentadas as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Atualmente, a literatura contém trabalhos que utilizam a lista *OWASP Top 10* como referência para o estudo de segurança na Web em diversos contextos, inclusive em aplicações de ensino e aprendizagem. A Tabela 1 sumariza os trabalhos relacionados.

Existem estudos concentrados na execução de análises manuais ou automáticas, com ou sem o apoio de ferramentas a fim de identificar vulnerabilidades no Moodle sem o foco no *OWASP Top 10* [Hernandez and Chavez 2008][Callejas-Cuervo et al. 2016]. Outros trabalhos consideram o *OWASP Top 10* em aplicações web, inclusive o Moodle [Silva et al. 2014][Monteverde 2014]. No entanto, tais trabalhos concentram-se em versões desatualizadas do *OWASP Top 10* (2013) e do Moodle (v. 2.7.1). Ainda considerando a literatura relacionada, cabe mencionar que há abordagens baseadas em varreduras automatizadas [Callejas-Cuervo et al. 2016] e exploração manual [Sampaio 2021] que consideram a versão do *OWASP Top 10* divulgada em 2017, mas não contemplam a plataforma Moodle. Similarmente, o trabalho [Priyawati et al. 2022], que se baseia na versão mais recente do *OWASP Top 10* (2021), também não estuda a plataforma Moodle. Por

Tabela 1. Sumário de trabalhos relacionados.

Ref.	Escopo	Metodologia	Ferramentas	Aplicação	Contramedidas
[Hernandez and Chavez 2008]	Vulnerabilidades em Geral	Análise Manual	N/A	Moodle 1.8.6	Contempla
[Monteverde 2014]	Owasp Top 10 (2013)	Scan Automático	Várias	Aplicações Web Diversas	Não Contempla
[Silva et al. 2014]	Owasp Top 10 (2013)	Teste de Penetração	Várias	Moodle 2.7.1	Contempla
[Callejas-Cuervo et al. 2016]	Vulnerabilidades em Geral	Scan Automático	Várias	Dokeos VLEs, Moodle 2.4.5	Não Contempla
[Sampaio 2021]	Owasp Top 10 (2017)	Exploração Manual	-	Portswigger, Acunetix	Contempla
[Ahamed et al. 2022]	Owasp Top 10 (2017)	Scan Automático	Várias	Aplicações Web Governamentais	Não Contempla
[Priyawati et al. 2022	Owasp Top 10 (2021)	Gray-box testing	OWASP ZAP	Aplicação Web	Não Contempla
Este trabalho	**Owasp Top 10 (2021)**	**Scan Automático, Análise Técnica**	**OWASP ZAP**	**Moodle 4.1**	**Contempla**

fim, tal trabalho ainda não contempla validações acerca das vulnerabilidades encontradas pela ferramenta utilizada – em contraste com o presente trabalho.

3. Materiais e Métodos

Nesta seção são detalhados os materiais e métodos empregados no estudo de caso apresentado neste trabalho. Em suma, a metodologia adotada (ilustrada na Figura 1) compreende a execução da ferramenta OWASP *Zed Attack Proxy* (ZAP) [OWASP 2022b], a qual envia requisições para a plataforma Moodle [Dougiamas 2022] (passo 1) e recebe respostas. Tais requisições podem incluir, por exemplo, tentativa de exploração de vulnerabilidades conhecidas. Por meio da análise das respostas recebidas (passo 2), a ferramenta gera um relatório de alertas de potenciais vulnerabilidades (passo 3).

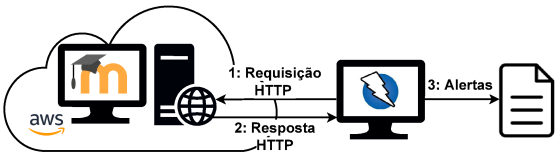

Figura 1. Cenário de experimentação adotado.

A seguir são apresentados mais detalhes sobre tais componentes bem como o processo de estudo a partir do relatório obtido por meio do OWASP ZAP.

Vale ressaltar que existem diversas ferramentas que buscam por vulnerabilidades em aplicações web [Mocelin et al. 2018]. No entanto, optamos pelo OWASP ZAP [OWASP 2022b] por consistir em uma ferramenta de código aberto e por ser um dos projetos da OWASP [OWASP 2022a]. Os principais benefícios de se usar essa ferramenta consiste nas diversas funcionalidades gratuitas oferecidas (*e.g.,* Varredura automatizada de aplicações web, *Web Spidering* e *Unthrottled Intruder*), em contraste com seu principal concorrente *Burp Suite* que apenas possui esses recursos na versão *premium*. Portanto, a ferramenta OWASP ZAP será empregada neste trabalho com a finalidade de efetuar análises a partir da plataforma Moodle [Dougiamas 2022].

O Moodle consiste em uma plataforma de ensino que é usada por milhões de usuários. Especificamente, o Moodle é utilizado por mais de 316 milhões de usuários no mundo todo. São ofertados mais de 41 milhões de cursos em 42 idiomas [Dougiamas 2022]. Portanto, tal plataforma foi escolhida por ser uma plataforma

de código aberto e também amplamente usada em instituições de ensino. Com isso, um dos principais objetivos desta análise consiste em validar se: (i) a plataforma é segura o suficiente para qualquer instituição de ensino apenas baixar, instalar e utilizar, ou (ii) se ela exige configurações adicionais para a mitigação de vulnerabilidades em potencial.

O cenário adotado consiste na instalação do OWASP ZAP na versão 2.12.0 e do Moodle na versão 4.1, lançada em 28 de novembro de 2022. A instalação do OWASP ZAP se deu por meio de uma máquina virtual com o sistema operacional Kali Linux na versão 2022.3. A plataforma Moodle foi instalada em uma máquina virtual hospedada no serviço Amazon AWS, com sistema operacional Ubuntu 22.04 e com o servidor Apache na versão 2.4.52. Uma vez que o cenário foi configurado, utilizou-se o modo de *scan* automático do OWASP ZAP. Esse *scan* é dividido em dois momentos. Primeiramente, ele rastreia e verifica passivamente cada página encontrada. Em seguida, usa um *scanner* ativo para atacar todas as páginas encontradas pelo *scanner* passivo. Em todo momento o ZAP registra todas as requisições e respostas enviadas e gera os alertas de vulnerabilidades baseados nessas informações. Dentre as informações obtidas pelo OWASP ZAP, destacam-se a quantidade de vulnerabilidades disponíveis e seus respectivos tipos, grau de criticidade, bem como a *Uniform Resource Locator* (URL) no qual cada vulnerabilidade foi encontrada.

4. Resultados e Discussão

Nesta seção, apresenta-se um estudo com a finalidade da identificação de potenciais vulnerabilidades existentes na plataforma *Moodle*, com foco nas vulnerabilidades da lista *OWASP Top 10*. A metodologia adotada é descrita na Seção 3. A Tabela 2 sumariza os alertas reportados pela ferramenta OWASP ZAP. Em particular, foram identificados 894 alertas que são divididos de acordo com seu risco: *Informativo*, *baixo*, *médio* e *alto*. Embora existam dez vulnerabilidades no *OWASP Top 10*, os alertas reportados pelo OWASP ZAP para a plataforma Moodle encontram-se entre as cinco primeiras vulnerabilidades (A01, A02, A03, A04 e A05, conforme [OWASP 2022a]).

Note que muitos dos alertas reportados são originados da falta de configurações adicionais além da instalação padrão do Moodle. O objetivo desta análise consiste em revelar o quão vulnerável pode ser uma instalação do Moodle quando realizada por um usuário que não tem o perfil de especialista em segurança da informação e, portanto, não implementa medidas adicionais além daquelas que a própria plataforma oferece.

A seguir, os alertas reportados pela ferramenta OWASP ZAP são apresentados e discutidos. Como contribuição adicional, são descritos os principais ataques que podem ser realizados através da exploração de tais vulnerabilidades, bem como as devidas contramedidas recomendadas para cada situação.

4.1. Vulnerabilidades de Risco Alto

Dentre as vulnerabilidades reportadas e categorizadas pela ferramenta OWASP ZAP [OWASP 2022b], apenas uma categoria foi classificada como tendo alto nível de risco. Foram gerados 5 alertas para a possibilidade de ataques de injeção de comandos através da vulnerabilidade *SQL Injection*. Essa vulnerabilidade pertence à categoria *Injection*, a qual ocupa a terceira posição no *OWASP Top 10*, conforme [OWASP 2022a].

Tabela 2. Alertas reportados pela ferramenta OWASP ZAP na plataforma Moodle.

Nome da vulnerabilidade	Risco	Categoria no *OWASP Top 10*	Alertas
SQL Injection	Alto	A03 - Injection	5
Missing Anti-CSRF tokens	Médio	A05 - Security Misconfiguration	56
CSP Header Not Set	Médio	A05 - Security Misconfiguration	37
HTTP to HTTPS Insecure Transition in Form Post	Médio	A02 - Cryptographic Failures	4
Hidden File Found	Médio	A05 - Security Misconfiguration	1
Missing Anti-Clickjacking Header	Médio	A05 - Security Misconfiguration	11
.htaccess Information Leak	Médio	A05 - Security Misconfiguration	2
Big Redirect Detected[1]	Baixo	A04 - Insecure Design	12
Cookie No HTTPOnly Flag	Baixo	A05 - Security Misconfiguration	5
Cookie without SameSite Attribute	Baixo	A01 - Broken Access Control	5
Disclosure of Date and Time - Unix	Baixo	A01 - Broken Access Control	150
Server Leaks Version Information[2]	Baixo	A05 - Security Misconfiguration	68
Strict-Transport-Security Header Not Set	Baixo	A05 - Security Misconfiguration	11
X-Content-Type-Option Header Missing	Baixo	A05 - Security Misconfiguration	65
Information Disclosure - Suspicious Comments	Informativo	A01 - Broken Access Control	89
Information Disclosure - Sensitive Information in URL	Informativo	A01 - Broken Access Control	2
Modern Web Application	Informativo	Nenhum	55
Re-examine Cache-control Directives	Informativo	Nenhum	8
User Agent Fuzzer	Informativo	Nenhum	240
User Controllabe HTML Element Attribute (Potential XSS)	Informativo	A03 - Injection	68

As falhas de SQL Injection são introduzidas quando os desenvolvedores de software criam consultas de banco de dados dinâmicas construídas com concatenação de *strings* que inclui entrada fornecida pelo usuário. Assim, uma potencial forma de ataque à plataforma Moodle pode se dar manipulando o valor do parâmetro *logintoken* ao enviar uma requisição *POST* a partir da página inicial */moodle/login/index.php*.

De acordo com o relatório da ferramenta, os resultados da página foram manipulados com sucesso através de uma tentativa de ataque automatizado que usa as condições booleanas denotadas nas Equações 1 e 2.

$$[NeM7GmXnkNHvlEGtgNg5SIQ2rFw0EtCe' AND '1' =' 1' - -] \quad (1)$$

$$[NeM7GmXnkNHvlEGtgNg5SIQ2rFw0EtCe' AND '1' =' 2' - -] \quad (2)$$

Como principais contramedidas para a contenção de ataques de injeção, como o *SQL Injection*, recomenda-se verificar todos os dados no lado do servidor. Não deve-se assumir que as informações provindas da aplicação cliente são sempre confiáveis, mesmo que hajam validações implementadas na aplicação cliente. Ademais, o uso de funções como o `mysqli_real_escape_string` ou procedimentos (*procedures*) podem ser usados como formas de mitigação da vulnerabilidade apresentada. Por fim, a adoção do princípio de privilégio mínimo é fortemente recomendada como forma de garantir que um atacante que explore a vulnerabilidade de *SQL Injection* esteja limitado às permissões de seu grupo de usuários.

4.2. Vulnerabilidades de Risco Médio

As vulnerabilidades reportadas pelo OWASP ZAP como risco médio constituem variações das vulnerabilidades *Cryptographic Failures* e *Security Misconfiguration*, as quais ocupam a segunda e quinta posição no *OWASP Top 10*, respectivamente. Foram gerados 111 alertas para seis tipos subcategorias dessas vulnerabilidades, conforme detalhado a seguir.

Os 56 alertas de *Cross-Site Request Forgery* (CSRF) foram causadas devido à ausência de *tokens* que são tipicamente usados para prevenir ataques que explorem essa

vulnerabilidade. O *token anti-CSRF* deve ser representado por um valor de tamanho grande e aleatório para dificultar a sua descoberta e também ser exclusivo por sessão de usuário. Ao explorar essa vulnerabilidade, o atacante força a vítima a enviar solicitações HTTP para um destino sem seu conhecimento ou intenção. Nas fases de arquitetura e design, as contramedidas cabíveis incluem a geração e verificação de *nonces* exclusivos para formulários. Já na fase de implementação, uma contramedida que pode ser adotada consiste na verificação do cabeçalho HTTP *Referer* de modo a confirmar se a solicitação se originou de uma página esperada. Note que a segunda solução pode ser ineficiente em casos em que usuários ou *proxies* legítimos desabilitam o envio deste cabeçalho por razões de privacidade [OWASP 2022c].

Os 37 alertas de *Content Security Policy* (CSP) *Header Not Set* ocorreram devido a falta de configuração deste cabeçalho HTTP. É importante observar que tal configuração não faz parte de um procedimento previsto durante a instalação padrão da plataforma Moodle. Portanto, instalações da plataforma que não contemplarem tais medidas adicionais podem ficar vulneráveis. Um atacante pode explorar tais vulnerabilidades para executar ataques de injeção de dados, por exemplo. Como forma de mitigação, deve-se assegurar que o servidor web esteja devidamente configurado para suportar a definição deste cabeçalho pelo cliente. As definições de cabeçalhos CSP podem variar de acordo com o navegador e suas versões [OWASP 2022c]:

- *X-WebKit-CSP*: Google Chrome (versão 14+) e Safari (versão 6+);
- *X-Content-Security-Policy*: Firefox (versão 4+) e Internet Explorer (versão 10+);
- *X-Content-Security-Policy*: Google Chrome (versão 25+), Firefox (versão 23 ou superior) e Safari (versão 7+).

Os 4 alertas *HTTP to HTTPS Insecure Transition in Form Post* revelam que a plataforma Moodle não foi instalada em um ambiente seguro que contemple o uso do protocolo *Hyper Text Transfer Protocol Secure* (HTTPS). Por conta disso a aplicação fica vulnerável à ataques *Man-In-The-Middle* (MITM), possibilitando que um invasor consiga interceptar a troca de dados entre duas partes e roubar suas informações. Como forma de mitigação, recomenda-se fortemente a utilização do protocolo HTTPS ao invés do HTTP.

O alerta de *Hidden File Found* revela que há um arquivo potencialmente confidencial que pode estar exposto a usuários não autorizados. Isso pode permitir que usuários mal-intencionados possam obter informações administrativas, obter conhecimento da configuração da aplicação e conseguir credenciais de acesso, possibilitando que ele consiga expandir os seus métodos de ataques na aplicação. Uma outra possibilidade consiste em realizar engenharia social usando as informações obtidas. Portanto, para mitigar essa possibilidade de ataque, recomenda-se fortemente desativar da produção qualquer componente que seja desnecessário. Se o componente for necessário, é importante garantir que o seu acesso seja restrito a usuários autenticados.

Os 11 alertas de *Missing Anti-Clickjacking Header* ocorreram pois o cabeçalho *X-Frame-Options* não foi configurado no cabeçalho de resposta HTTP. Com isso, o invasor pode realizar ataque de *Clickjacking*, onde ocorre uma sobreposição de elementos invisíveis ou semi-transparentes em cima de áreas clicáveis da página. Para mitigar essa vulnerabilidade, é necessário certificar-se que o cabeçalho*X-Frame-Options* esteja confi-

gurado para todas as páginas web utilizadas pela aplicação.

Os 2 alertas de *.htaccess Information Leak* ocorrem devido ao acesso indevido ao arquivo *.htaccess* da aplicação. Esse arquivo pode ser utilizado para realizar alterações nas configurações do servidor Apache. Portanto, um invasor ao conseguir acesso a este arquivo, pode realizar a ativação e desativação de funcionalidades que o servidor Apache pode oferecer. Portanto, a melhor maneira para remover esse alerta consiste em desativar qualquer tipo de acesso a este arquivo.

4.3. Vulnerabilidades de Risco Baixo

As vulnerabilidades reportadas pelo OWASP ZAP como risco baixo constituem variações das categorias de *Security Misconfiguration*, *Broken Access Control* e *Insecure Design*. No total, foram gerados 316 alertas para sete tipos de subcategorias dessas vulnerabilidades, conforme detalhado a seguir.

Um exemplo de falha que pode ser explorada para revelar o *cookie* de um usuário para terceiros é o *Cross-Site Scripting*. Portanto, uma forma de mitigar tal acesso de *cookies* por terceiros compreende a inclusão do sinalizador *HTTPOnly* no cabeçalho de resposta HTTP [OWASP 2022c].

Os 5 alertas de *Cookie No HTTPOnly* foram gerados devido a ausência do atributo *HTTPOnly* no cabeçalho HTTP *Set-Cookie*. Este atributo tem por finalidade impedir que os *cookies* sejam acessados através de *scripts*. Dessa forma, um tipo de ataque que pode explorar essa falta de configuração consiste no *Cross-Site Scripting*. Então, para mitigar a possibilidade de um atacante acessar os *cookies* através de *scripts* é definindo o atributo *HTTPOnly* no cabeçalho de resposta [OWASP 2022c].

Os 5 alertas de *Cookie without SameSite Attribute* foram gerados devido à ausência do atributo *SameSite* no cabeçalho HTTP *Set-Cookie*. Tal atributo tem por finalidade evitar que um *cookie* seja enviado como resultado de uma solicitação entre sites. Dessa forma, um tipo de ataque que pode explorar essa falta de configuração consiste no CSRF, já mencionado anteriormente. Então, para mitigar a possibilidade de um atacante falsificar solicitações entre sites através do envio de *cookies*, deve-se habilitar o atributo *SameSite* no cabeçalho de resposta [OWASP 2022c],

Os 12 alertas de *Big Redirect Detected (Potential Sensitive Information Leak)* ocorreram devido ao tamanho da resposta recebida pelo servidor no momento em que o mesmo efetua redirecionamentos, o qual foi considerado longo pela ferramenta OWASP ZAP. O potencial risco dessa vulnerabilidade consiste nos casos em que a resposta do redirecionamento contém informações confidenciais ou dados pessoais. Ademais, com base em tais informações, novos ataques podem ser explorados. Portanto, o jeito mais eficaz de mitigar essa vulnerabilidade é configurar o servidor para que o mesmo não exiba conteúdos confidenciais ou privados no redirecionamento [OWASP 2022c].

Os 150 alertas de *Disclosure of Date and Time - Unix* ocorreram devido ao envio das informações de data e hora do servidor na resposta HTTP. O principal problema do vazamento dessa informação consiste em facilitar a um atacante a obtenção de informações internas para a criação de padrões de exploração. Para mitigar esse tipo de vazamento de informação, deve-se verificar se o uso dessa informação por parte de um atacante pode ser agregada com outras informações para o mapeamento de padrões por atacantes. Em caso

positivo, deve-se remover o campo *timestamp* do cabeçalho resposta [OWASP 2022c].

Os 68 alertas de *Server Leaks Version Information* ocorreram devido ao vazamento da versão do servidor *web* na resposta HTTP. Esse tipo de informação pode ser usada pelo atacante para identificar outras vulnerabilidades que são conhecidas naquela versão em que o servidor se encontra. Como forma de mitigação, pode-se remover o campo *"Server"* do cabeçalho de resposta HTTP.

Os 11 alertas gerados por *Strict-Transport-Security Header Not Set* ocorreram devido a configuração deste cabeçalho HTTP não ser um procedimento que faz parte da instalação padrão da plataforma Moodle. O HTTP *Strict-Transport-Security* (HSTS) consiste em um aprimoramento de segurança opcional que redireciona solicitações HTTP não seguras para HTTPS. A falta da utilização desse cabeçalho possibilita que ataques como *Man-In-The-Middle* ocorra. Portanto, como forma de mitigação, deve-se realizar a configuração deste cabeçalho no servidor web.

Os 65 alertas de *X-Content-Type-Option Header Missing* ocorreram devido a falta de configuração do cabeçalho *X-Content-Type-Option*, o qual faz parte do mecanismo de *Anti-Mime-Sniffing*. Sem essa configuração um invasor pode disfarçar um arquivo, fazendo com que ele se pareça outro, conseguindo assim ataques do tipo *Cross-Site-Scripting*. Portanto, para mitigar essa vulnerabilidade, recomenda-se a configuração do cabeçalho *X-Content-Type-Options* como *nosniff* para todas as páginas web.

4.4. Motivos Informativos

Os 89 alertas de *Suspicious Comments* foram gerados devido às respostas conterem comentários suspeitos. Isso contribui com um atacante à medida que pode fornecer informações sobre o funcionamento do sistema. Ademais, um atacante pode conseguir credenciais de acesso, caso elas estejam disponíveis nos comentários. Um método para mitigar essa vulnerabilidade consiste em remover quaisquer comentários desnecessários ou que possam explanar informações adicionais sobre o funcionamento do sistema.

Os 2 alertas de *Information Disclosure - Sensitive Information in URL* foram emitidos devido a suspeita de vazamento de informações confidenciais na URL: `/moodle/login/index.php?testsession=2`. Incluir informações confidenciais em uma URL aumenta o risco de que informações relevantes para um ataque sejam capturadas por um invasor. No entanto, a mitigação deste risco é simples, basta não transmitir informações sensíveis na URL, tradicionalmente feitas usando o método `GET`.

Os 55 alertas do *Modern Web Applications* foram gerados como um informativo de que a aplicação Moodle contém recursos modernos e pode ser melhor explorada por ferramentas tais como o *Ajax Spider*. Portanto, uma vez que isso não representa uma vulnerabilidade propriamente dita, nenhuma ação é necessária como contramedida.

Os 8 alertas de *Re-examine Cache-control Directives* foram gerados devido a ausência ou má configuração de cabeçalhos de controle de cache. Isso permite que navegadores e *proxies* acessem e alterem o conteúdo que encontra-se em cache. Com isso, um atacante pode explorar tal vulnerabilidade para executar o envenenamento de *cache*, manipulando, por exemplo, o Sistema de Nomes de Domínio, do inglês, *Domain Name System* (DNS). Como forma de mitigação, recomenda-se definir o cabeçalho HTTP de controle de cache como `no-cache`, `no-store`, `must-revalidate`. Caso haja a

necessidade do armazenamento de recursos em *cache*, recomenda-se a definição das diretivas `public`, `max-age`, `immutable` para mitigar e proteger a aplicação de ataques.

Os 240 alertas de *User Agent Fuzzer* foram gerados devido ao recebimento de diferentes respostas ao modificar o campo *User Agent* na requisição. Essa potencial vulnerabilidade demonstra que a aplicação pode estar sujeita a falhas oriundas de modificações no campo *User Agent* da requisição. Para mitigar essa vulnerabilidade, deve-se configurar as URLs para fornecer uma resposta independentemente do *User Agent*.

Os 68 alertas de *User Controllabe HTML Element Attribute (Potencial XSS)* foram gerados pois o ZAP identificou que o parâmetro *?lang*= na URL consegue modificar os atributos HTML do site. Por conta disso, um usuário malicioso pode utilizar desse atributo para a realização de ataques de *Cross-Site Scripting* para roubar os *cookies* da sessão de um usuário. A forma de remover esse alerta, é realizando a higienização dos *inputs* do usuário na URL.

5. Validação e Correção de Alertas

Nesta seção apresenta-se uma análise dos alertas obtidos pelo OWASP ZAP de modo a verificar se tais alertas apresentam risco à aplicação Moodle e apontar as devidas soluções.

5.1. Investigação dos Alertas de Risco Alto

De modo a verificar se a aplicação Moodle é vulnerável à ataques do tipo SQL Injection, foram empregadas as ferramentas para testes automáticos e testes manuais, a saber o *SQL-Map* e *Burp Suite*, respectivamente. Para realizar a validação manual, as configurações do *Burp Suite* foram ajustadas para interceptar as solicitação HTTP que havia gerado o alerta de *SQL Injection* no *OWASP ZAP*. Durante o teste, o parâmetro "*logintoken*", que havia sido reportado na requisição testada pelo OWASP ZAP, foi testado com os valores sugeridos pela ferramenta. No entanto, não houve confirmação de uma injeção de SQL bem-sucedida. Para realizar a validação automática, foram inseridos diversos *payloads* de injeção de SQL no parâmetro "*logintoken*". No entanto, o SQLMap não foi capaz de explorar com sucesso nenhuma dessas injeções. Isso sugere que a plataforma Moodle estava protegida contra vulnerabilidades de injeção de SQL. Dessa forma, verificou-se que o alerta de *SQL Injection* se trata de um falso positivo.

5.2. Investigação dos Alertas de Risco Médio

Atualmente o Moodle implementa o uso de um *token* por sessão para mitigar os riscos dos ataques de CSRF, gerando-se assim o alerta de *Missing Anti-CSRF tokens*. De acordo com a documentação de *CSRF Prevention Cheat Seet* da própria OWASP, esses *tokens* por sessão são uma medida aceitável para evitar o ataque. Em relação ao alerta *CSP Header Not Set*, foi observada a ausência de cabeçalhos CSP nas respostas HTTP. Para corrigir tal problema, foi então configurado um *plug-in* disponibilizado pelo próprio Moodle, que pode ser baixado e instalado pela parte administrativa do site. Uma maneira alternativa de solucionar tal problema que também se mostrou efetiva consiste na configuração do cabeçalho dentro do arquivo de configuração do Apache (*apache2.conf)* utilizando a seguinte linha para configurá-lo:

```
Header set Content-Security-Policy "default-src 'self';"
```

De modo a corrigir-se o alerta *HTTP to HTTPS Insecure Transition in Form Post*, configurou-se um certificado digital de modo a permitir a implementação e uso do protocolo HTTPS, o qual consiste em uma implementação de segurança que utiliza o protocolo TLS/SSL. Já com relação ao alerta *Hidden File Found*, verificou-se que havia um arquivo oculto chamado *composer.lock*, o qual estava exposto para todos os usuários. Então a medida tomada para resolver esse alerta foi criar a seguinte linha no arquivo de configuração do Apache que resulta no bloqueio ao acesso de arquivos com a extensão *.lock*, gerando-se o código HTTP 403 (*forbidden*).

```
<Files *.lock> Require all denied </Files>
```

Em relação ao alerta *Missing Anti-Clickjacking Header*, verificou-se que o cabeçalho para a prevenção de ataques de *Clickjacking* não estavam configurados. No entanto, os testes feitos para a realização deste tipo de ataque nos mostraram que não é possível abrir o Moodle em um *iframe*. Mesmo assim, para realizar a remoção deste alerta, foi realizada a seguinte configuração no arquivo do Apache:

```
Header set X-Frame-Options:  "sameorigin"
```

Já sobre o alerta *htaccess Information Leak*, foi observado que o arquivo *.htaccess* estava exposto a todos os usuários na URL identificada pelo OWASP ZAP. Ao contrário da configuração feita para a mitigação do *composer.lock*, para o *.htaccess* a configuração que melhor resolveu este alerta foi a listada a seguir, onde qualquer usuário que tentar acessar um arquivo *.htaccess* receberá o código de *status* HTTP 404 (*not found*).

```
RedirectMatch 404 /\\.htaccess
```

5.3. Investigação dos Alertas de Risco Baixo

Para remover o alerta *Big Redirect Detected*, as respostas HTTP foram analisadas na tentativa de identificar falhas ou informações sensíveis dos usuários. Também, investigou-se os códigos fontes relacionados. Não foram identificadas informações sensíveis sendo expostas. Já o alerta de *Cookie without SameSite Attribute e Cookie No HTTPOnly Flag* indica a ausência da definição de atributos, o que foi resolvido com a seguinte configuração:

```
Set-Cookie:  HttpOnly; SameSite=Strict
```

O alerta *Disclosure of Date and Time* não apresenta riscos e também foi encontrado em diversas plataformas em produção na web. No caso do Moodle, revelar o dia e a hora na resposta HTTP não gera um risco de segurança para a aplicação.

Ao analisar o alerta *Server Leaks Version Information*, observou-se que as requisições HTTP estavam retornando a versão do servidor no cabeçalho *Server*. Para solucionar este problema, utilizou-se a seguinte configuração:

```
ServerSignature Off
ServerTokens Prod
```

Ao analisar o alerta *Strict-Transport-Security Header Not Set*, observou-se que de fato não havia o cabeçalho Strict-Transport-Security nas respostas HTTP. Portanto, editou-se o arquivo de configuração do Apache a fim de resolver este alerta. Para resolver este problema, a seguinte linha foi adicionada:

```
Header always set "Strict-Transport-Security" "max-age=31536000"env=HTTPS
```

5.4. Investigação dos Alertas de Risco Informativo

O alerta *Information Disclosure - Suspicious Comments* constitui um falso positivo visto que o conteúdo incluído nos comentários analisados no código-fonte que gerou o alerta não apresenta informações sensíveis. Similarmente, o alerta *Information Disclosure - Sensitive Information in URL* indica que o *testsession* passado pela URL pode conter um conteúdo sensível. Esse parâmetro identifica o usuário que está realizando o login na plataforma. A exploração de vulnerabilidades relacionadas a alteração deste parâmetro não foi bem sucedida: o Moodle solicita que o usuário realize *logout* e *login* novamente para acessar com outro id. Adicionalmente, em relação ao alerta *Modern Web Application*, nenhuma medida precisa ser tomada, pois o mesmo tem cunho informativo a respeito do uso de uma aplicação web moderna. Ainda, ao analisar o alerta *Re-examine Cache-Control Directives*, observou-se que o uso das diretivas "no-cache", "no-store", "max-age" e "must-revalidate" possibilita garantir que o conteúdo seja atualizado regularmente e não seja armazenado em cache por muito tempo.

```
Header set Cache-Control "max-age=3600, public"
```

Outro alertas de risco informativo são o *User Agent Fuzzer*, que indica o uso do OWASP ZAP com modificações cabeçalho do usuário (User-Agent), e o alerta *User Controllable HTML Element Attribute*, que notifica que o parâmetro *lang* pode ser alterado pelo usuário para modificar o idioma do site. Ambos não configuram vulnerabilidades.

6. Conclusão e Trabalhos Futuros

Este trabalho apresentou um estudo de caso acerca das potenciais vulnerabilidades da lista *OWASP Top 10* atualizada presente no ambiente virtual de ensino e aprendizagem Moodle por meio da ferramenta OWASP ZAP. A plataforma Moodle é utilizada por mais de 316 milhões de usuários e com o aumento de ataques cibernéticos é de extrema importância verificar o nível de segurança da plataforma ao ser instalada por um usuário comum. Portanto esse estudo mostrou que a plataforma Moodle, ao ser instalada sem qualquer tipo de configuração de segurança adicional está sujeita a diversas vulnerabilidades, através das quais um invasor pode se aproveitar para efetuar ataques na aplicação. Tal análise foi realizada utilizando o *scan* automático da ferramenta OWASP ZAP, o qual gerou um total de 894 alertas divididos entre 20 vulnerabilidades.

Como trabalhos futuros, pretende-se realizar a configuração dos métodos de mitigação sugeridos ao longo do trabalho e realizar novamente o *scan* da ferramenta OWASP ZAP para verificar a eficácia desses métodos. Em outra abordagem, pode-se utilizar outras ferramentas de *scan* de vulnerabilidades (*e.g., OpenVAS*, *Nessus*, *Nikto*, entre outras) para comparar os resultados obtidos utilizando o ZAP e assim conseguir verificar qual possui maior eficiência para detectar vulnerabilidades na aplicação.

Referências

Callejas-Cuervo, M., Alarcón-Aldana, A., and López, A. B. (2016). Security evaluation model for virtual learning environments. In *2016 XI Latin American Conference on Learning Objects and Technology (LACLO)*, pages 1–6, San Carlos, Costa Rica. IEEE, IEEE.

Dougiamas, M. (2022). Aprendizado on-line com o LMS mais popular do mundo - Moodle. Disponível em: `https://moodle.org/`.

Fredj, O. B., Cheikhrouhou, O., Krichen, M., Hamam, H., and Derhab, A. (2020). An owasp top ten driven survey on web application protection methods. In *International Conference on Risks and Security of Internet and Systems*, pages 235–252, New York, NY, USA. Springer, Springer, Cham.

Hernandez, J. C. G. and Chavez, M. A. L. (2008). Moodle security vulnerabilities. In *International Conference on Elec. Enginn., Computing Science and Automatic Control*, pages 352–357, Mexico. IEEE.

Kaspersky (2021). Kaspersky security bulletin 2021 statistics. `https://go.kaspersky.com/rs/802-IJN-240/images/KSB_statistics_2021_eng.pdf`.

Mocelin, B. V., Farias, K., Gonçales, L., and Bischoff, V. (2018). Improvements to the identification process of vulnerable components: Deciding about updates. In *Proceedings of the XIV Brazilian Symposium on Information Systems*, SBSI'18, New York, NY, USA. Association for Computing Machinery.

Monteverde, W. A. (2014). Estudo e análise de vulnerabilidades web. B.S. thesis, UFTPR.

OWASP (2022a). Owasp top 10:2021. Disponível em: `https://owasp.org/www-project-top-ten/`. Acesso em: 25 set. 2022.

OWASP (2022b). Owasp zed attack proxy (zap). Disponível em: `https://www.zaproxy.org/`.

OWASP (2022c). Owasp zed attack proxy (zap) - alerts documentation. Disponível em: `https://www.zaproxy.org/docs/alerts/`. Acesso em: 25 set. 2022.

Priyawati, D., Rokhmah, S., and Utomo, I. C. (2022). Website vulnerability testing and analysis of website application using owasp. *International Journal of Computer and Information System*, 3(3):142–147.

Sampaio, F. F. (2021). *Uma análise prática das principais vulnerabilidades em aplicações web baseado no top 10 OWASP*. Bacharelado, Universidade Federal do Paraná (UFC), Paraná.

Santos, L. C. M. C., Prado, E. P. V., and Chaim, M. L. (2019). Vulnerability Detection Techniques and Tools and Their Relationship to Agile Methods and Software Quality and Service Models. In *Proceedings of the XV Brazilian Symposium on Information Systems*, SBSI'19. Association for Computing Machinery.

Schendes, W. (2022). Ataques cibernéticos no brasil cresceram 46% no segundo trimestre. `https://olhardigital.com.br/2022/08/09/seguranca/ataques-ciberneticos-brasil-cresce-46/`.

Silva, R. R. T., de Lima, R. W., Leite, C. R. M., and da Silva, R. R. T. (2014). Investigação de segurança no moodle. *RENOTE*, 12(2):61.