23° enancib
ENCONTRO NACIONAL DE PESQUISA EM CIÊNCIA DA INFORMAÇÃO
06 a 10 de novembro de 2023
ARACAJU-SE

**GT-8 – Informação e Tecnologia**

**ISSN 2177-3688**

# VULNERABILIDADES DE SEGURANÇA DO SOFTWARE DSPACE: ANÁLISE E CORREÇÃO

## *DSPACE SOFTWARE SECURITY VULNERABILITIES: ANALYSIS AND CORRECTION*

**Milton Shintaku** - Instituto Brasileiro de Informação em Ciência e Tecnologia (IBICT)
**Mirele Carolina Souza Ferreira Costa** - Instituto Brasileiro de Informação em Ciência e Tecnologia (IBICT)
**Lucas Ângelo Silveira** - Instituto Brasileiro de Informação em Ciência e Tecnologia (IBICT)
**Rebeca dos Santos de Moura** - Instituto Brasileiro de Informação em Ciência e Tecnologia (IBICT)
**Raíssa da Veiga de Menêses** - Instituto Brasileiro de Informação em Ciência e Tecnologia (IBICT)

**Modalidade: Trabalho Completo**

**Resumo: Introdução:** A proteção contra ameaças e ataques cibernéticos em aplicações web torna-se mais complexa à medida que a tecnologia avança. Consequentemente, ferramentas de *scanner* de vulnerabilidade têm sido amplamente usadas como mecanismo para verificar a segurança e proteção contra os ataques que possam comprometer as informações de um site. Testes de segurança são vitais para evitar ataques de invasores em aplicações web, de modo que a constante mudança e evolução dos sistemas web são um desafio. **Objetivo:** Neste estudo, buscou-se identificar, analisar, propor soluções e corrigir as ameaças de segurança do software livre DSpace. **Procedimentos metodológicos:** Utilizou-se um *scanner*, uma ferramenta frequentemente empregada por especialistas para testes de segurança automatizados, que realiza varreduras para detectar vulnerabilidades e falhas em aplicações web. Implementou-se soluções para correção das vulnerabilidades apontadas no teste de segurança com DSpace. **Resultados:** Foram encontradas cinco vulnerabilidades com risco alto ou médio no DSpace. Além disso, foram realizados esforços para depurar, compreender e corrigir todas essas ameaças de segurança. **Considerações finais:** Portanto, visou-se à segurança da informação, bem como a confidencialidade, preservação e integridade das informações do DSpace.

**Palavras-chave:** vulnerabilidades de segurança; biblioteca digital; DSpace; segurança da informação.

**Abstract: Introduction:** Protection against threats and cyber attacks in web applications becomes more complex as technology advances. Consequently, vulnerability scanner tools have been widely used as a mechanism to verify security and protect against attacks that could compromise a website's information. Security tests are vital to prevent intruder attacks on web applications, so the constant change and evolution of web systems is a challenge. **Objective:** In this study, we sought to identify, analyze, propose solutions and correct DSpace free software security threats. **Methodological procedures:** A scanner was used, a tool often used by experts for automated security testing, which performs scans to detect vulnerabilities and flaws in web applications. Solutions were implemented to correct the vulnerabilities identified in security tests with DSpace. **Results:** Five high or medium risk vulnerabilities were found in DSpace. In addition, efforts have been made to debug, understand, and correct all these security threats. **Final considerations:** Therefore, information security was sought, as well as the confidentiality, preservation and integrity of DSpace information.

**Keywords:** security vulnerabilities; digital library; DSpace; information security.

## 1 INTRODUÇÃO

Com o desenvolvimento da tecnologia e o surgimento da internet os sistemas informatizados estão cada vez mais presentes, acessados pelos computadores e smartphones. Sob esse contexto, Alazmi e Leon (2023) advertem que as aplicações web tornaram-se gradativamente mais populares no oferecimento de informações e serviços entre empresas e organizações, porém, tornaram-se mais suscetíveis a riscos de segurança. Com isso, a avaliação de vulnerabilidade de aplicações web é fundamental para segurança e proteção contra ameaças e ataques cibernéticos. As ferramentas de *scanners* são um recurso poderoso para essa finalidade, pois são capazes de realizar varreduras no sistema a fim de encontrar vulnerabilidades a serem corrigidas.

Dentre os sistemas de informação mais utilizados nas universidades estão os repositórios, eles disponibilizam a produção acadêmica e, na maioria das vezes, são implementados com o software DSpace, segundo relatam Shintaku e Vechiato (2018), em grande parte, em razão das ações do Instituto Brasileiro de Informação em Ciência e Tecnologia (Ibict). Da mesma forma, órgãos de governo têm usado o DSpace na criação de Bibliotecas Digitais para gestão da memória técnica (MACÊDO; SHINTAKU; BRITO, 2015).

Dessa forma, este estudo teve como objetivo identificar, analisar, propor soluções e corrigir as vulnerabilidades de segurança encontradas no sistema informacional DSpace. A análise foi realizada de acordo com a abordagem do *Open Web Application Security Project* (OWASP), usando a ferramenta de *scanner Zed Attack Proxy* (ZAP)[1] e visando à segurança da informação sobre os dados. Buscou-se encontrar vulnerabilidades e riscos do website, além de fornecer recomendações para melhorar a segurança (NURBOJATMIKO; LATHIFAH; AMRI; ROSIDAH, 2022), assim, contribuindo com as discussões sobre a segurança de ferramentas que oferecem robustez aos sistemas de informação.

## 2 FERRAMENTA DE *SCANNER* OWASP ZAP

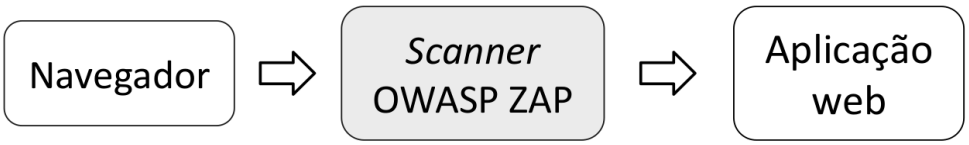
Desde o seu lançamento em 2010, o OWASP ZAP é uma ferramenta livre e de código aberto de *scanner* muito utilizada no mundo, mantida sob o projeto de segurança de aplicativos da web com colaboração oriunda de vários países. OWASP ZAP foi projetado

[1] O site oficial da Ferramenta OWASP ZAP está disponível em: https://www.zaproxy.org/.

especificamente para testar esses aplicativos, é flexível, extensível e gratuito. Além disso, a ferramenta permite uma verificação automatizada e a exploração manual de aplicações web (MAKINO; KLYUEV, 2015). Albahar, Alansari e Jurcut (2022) defendem que o OWASP ZAP é a melhor opção não comercial para testes de vulnerabilidade, após um estudo de comparação com outras ferramentas levando em consideração a robustez das funcionalidades propostas pelo OWASP ZAP, aliada a facilidade de uso.

Em sua essência, o OWASP ZAP é conhecido como "*proxy man-in-the-middle (MitM)*", que atua como uma interface entre a ferramenta de navegação e o aplicativo. A Figura 1 ilustra como o *scanner* posiciona-se entre o navegador do testador e o aplicativo web alvo para que possa interceptar e inspecionar as mensagens enviadas entre o navegador e o aplicativo web, modificar o conteúdo, se necessário, e encaminhar esses pacotes para o destino (OWASP, c2023). Dessa forma, o OWASP ZAP pode verificar vulnerabilidades analisando as mensagens trocadas entre esses elementos.

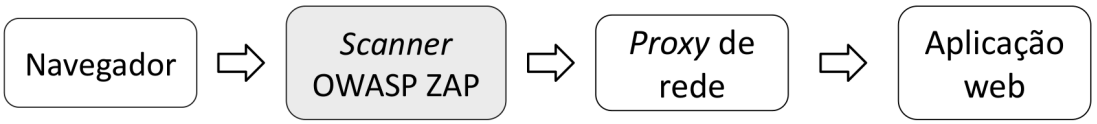
**Figura 1 –** Ferramenta OWASP ZAP atua entre o navegador e a aplicação web




**Fonte:** Adaptado de Owasp (c2023, p. 2)

Como em muitos ambientes corporativos existem proxy de rede em uso, o OWASP ZAP pode ser configurado para se conectar a um, conforme ilustrado na Figura 2. Essa flexibilidade possibilita que o OWASP ZAP seja ajustado aos ambientes corporativos, possibilitando ser utilizado pelas mais diversas infraestruturas computacionais.

**Figura 2 –** Ferramenta OWASP ZAP atua entre o navegador e o proxy de rede




**Fonte:** Adaptado de Owasp (c2023, p. 2)

## 2.1 Testes de segurança

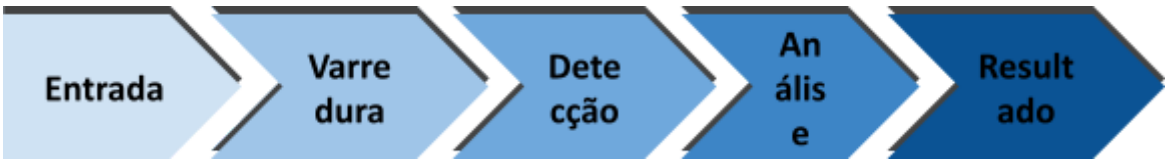
Os testes no OWASP ZAP são geralmente divididos de acordo com o tipo de vulnerabilidade que está sendo testada ou o tipo de teste que está sendo realizado, segundo OWASP (c2023) são eles:

> Avaliação de vulnerabilidade - O sistema é escaneado e analisado quanto à problemas de segurança; Teste de penetração - O sistema passa por análise e ataque de invasores maliciosos simulados; Teste de tempo de execução -

> O sistema passa por análises e testes de segurança de um usuário final; Revisão de Código - O código do sistema sob (OWASP, c2023, p. 1).

Jobin e Babu (2021) defendem 5 etapas no teste de segurança da ferramenta OWASP ZAP descritas na Figura 3. Na entrada, um host/ID ou *Uniform Resource Locator* (URL) deve ser fornecido para o ataque. O processo de varredura é então realizado para detectar as ameaças, seguido do método de análise de risco que é realizado após as vulnerabilidades terem sido detectadas e, por fim, o resultado é concluído.

**Figura 3 –** Etapas do teste de segurança da ferramenta OWASP ZAP




**Fonte:** Adaptado de Jobin e Babu (2021, p. 107)

As vulnerabilidades identificadas são divididas em categorias na análise de risco, são elas: alto; médio; baixo; informativo e falso positivo. Segundo o guia do OWASP (c2023), o *scanner* rastreia a aplicação web, constrói um mapa das páginas web e os recursos usados para renderizar as páginas. Na sequência, registra as solicitações e respostas enviadas para cada página web e cria alertas com a categoria de risco se houver alguma potencial ameaça em uma solicitação ou resposta.

## 3 PROCEDIMENTOS METODOLÓGICOS

O estudo aplicou aspectos da área de informática alinhada com a ciência e gestão da segurança da informação em bibliotecas digitais e repositórios implementados com o software DSpace. A abordagem pode ser replicada em outros sistemas implementados com o DSpace, bem como, a implementação das soluções dessas vulnerabilidades é uma inovação na segurança do sistema.

A versão mais recente e estável do DSpace até o momento é a 7.6, que foi lançada em julho de 2023. O DSpace 7.x está atualmente em processo de consolidação, com a comunidade e seus mantenedores realizando extensos testes.

Neste estudo, optou-se pela versão 6.4 do DSpace, que representa a edição mais recente e estável da série 6.x, lançada em julho de 2022. Essa escolha se deve ao fato de que as versões 6.x continuam sendo amplamente empregadas, especialmente por universidades e órgãos governamentais no Brasil.

O software DSpace foi instalado em uma configuração típica amplamente adotada pela comunidade, que utiliza ferramentas de código aberto. Para isso, foi implementado em um ambiente operacional baseado na distribuição Debian GNU/Linux. A infraestrutura incluiu a linguagem Java com o kit de desenvolvimento OpenJDK, o servidor web Apache Tomcat, o banco de dados PostgreSQL, o serviço de indexação e busca Apache Solr, e a interface de usuário JSPUI (Java Server Pages User Interface). Além disso, a instalação contou com o suporte dos softwares Apache Maven e Apache Ant, que facilitaram o gerenciamento das dependências necessárias.

O teste teve como objetivo compreender o comportamento em instalações mais comuns do software DSpace, que, conforme a recomendação da comunidade, empregam um ambiente operacional baseado na distribuição Debian GNU/Linux. Portanto, não foram realizados testes em diferentes sistemas operacionais, uma vez que a ênfase estava em avaliar o desempenho e a estabilidade em um cenário típico de uso.

Atualmente, há uma variedade de softwares de *scanners* disponíveis para testes de vulnerabilidade em aplicações web. Entre eles, destacam-se ferramentas comerciais como Nessus[2] e Burp Suite[3], que oferecem edições gratuitas, e também ferramentas de código aberto como Nikto[4]. Para conduzir os testes de vulnerabilidade, optou-se pela utilização da ferramenta de código aberto OWASP ZAP, versão 2.12.0[5], mantida pela comunidade OWASP. Isso implica que a ferramenta é respaldada por uma comunidade ativa composta por profissionais de segurança e desenvolvedores, o que garante atualizações frequentes e suporte contínuo ao longo do tempo. Além disso, uma característica essencial do OWASP ZAP é sua capacidade de automatizar os testes de segurança, o que desempenha um papel fundamental na identificação ágil de vulnerabilidades em aplicações web.

Neste contexto, a metodologia empregada foi realizada em 3 etapas, a saber:

1. Teste no *scanner* OWASP ZAP:
    a. Identificação de vulnerabilidades: detectar ameaças e potenciais caminhos de ataques de invasores;
    b. Avaliação de vulnerabilidades: categorizar as vulnerabilidades identificadas;

[2] Ferramenta de *scanner* Nessus. Disponível em: https://pt-br.tenable.com/products/nessus/.
[3] Ferramenta de *scanner* Burp Suite. Disponível em:https://software.com.br/p/burp-suite/.
[4] Ferramenta de *scanner* Nikto. Disponível em: https://cirt.net/Nikto2/.
[5] Ferramenta de *scanner* OWASP ZAP. Disponível em: https://www.zaproxy.org/download/.

2. Estudo exploratório: estudo sobre as vulnerabilidades identificadas para verificar seu impacto potencial à confidencialidade e à integridade das informações e as possíveis soluções;
3. Correção: implementação das soluções para correção das vulnerabilidades encontradas no teste de vulnerabilidade com DSpace.

O caráter aplicado do estudo tem aspectos exploratórios na medida em que torna mais familiar para os gestores dos repositórios e bibliotecas digitais as preocupações dos administradores de ambiente computacional com a segurança. Nesse ponto, ressalta a interdisciplinaridade da ciência da informação e sua grande proximidade com a informática.

## 4 RESULTADOS

Com o objetivo de buscar a melhor prática e a segurança das informações mantidas pelo DSpace, foi aplicada a análise de vulnerabilidade e, posteriormente, aplicadas as correções necessárias para tornar o ambiente mais seguro.

### 4.1 Testes no *scanner* OWASP ZAP

Para alcançar a identificação e validação de possíveis vulnerabilidades, testou-se o software DSpace na ferramenta OWASP ZAP.

#### 4.1.1 Identificação e avaliação das vulnerabilidades

O teste de vulnerabilidade ocorreu de forma que o website foi escaneado e analisado quanto aos problemas de segurança. Dessa forma, estão descritas no Quadro 1 as 5 vulnerabilidades encontradas e categorizadas com risco alto ou médio pelo OWASP ZAP, levando em consideração o código default do DSpace.

**Quadro 1 –** Vulnerabilidades encontradas pelo *scanner* OWASP ZAP

| Vulnerabilidade | Risco |
|---|---|
| BIBLIOTECAS *JAVASCRIPT* (JS) VULNERÁVEIS | Alto |
| AUSÊNCIA DE TOKENS ANTI *CROSS-SITE REQUEST FORGERY* (CSRF) | Médio |
| *CONTENT SECURITY POLICY* (CSP) | Médio |
| CABEÇALHO *X-CONTENT-TYPE-OPTIONS* AUSENTE | Médio |

| Vulnerabilidade | Risco |
|---|---|
| CABEÇALHO *ANTI-CLICKJACKING* AUSENTE | Médio |

**Fonte:** Elaborado pelos autores

A avaliação de risco realizada pelo *scanner* determina o nível de risco associado à vulnerabilidade encontrada. O risco é categorizado com base em diversos fatores, incluindo o impacto potencial de uma vulnerabilidade que se refere às consequências negativas que sua exploração pode causar no sistema. Isso inclui o acesso não autorizado a informações sensíveis, interrupção dos serviços, perda de integridade de dados, danos à reputação da organização, entre outros. Portanto, quanto maior o impacto, maior o risco associado à vulnerabilidade.

### 4.2 Estudo exploratório

Foram realizados estudos sobre cada uma das vulnerabilidades apontadas nos testes práticos de segurança. O estudo buscou esforços para depurar, compreender e propor soluções para essas ameaças de segurança do DSpace.

#### 4.2.1 Bibliotecas JS vulnerável

Vulnerabilidades em bibliotecas JavaScript desatualizadas representam riscos altos devido a diversos fatores. Primeiro, as vulnerabilidades conhecidas podem ser exploradas por invasores, comprometendo a segurança do sistema. Segundo, a falta de correções de segurança nessas bibliotecas deixa o sistema vulnerável a ameaças conhecidas. Terceiro, a ampla adoção dessas bibliotecas aumenta a superfície de ataque, afetando muitos sistemas. Quarto, a injeção de código malicioso pode ocorrer, permitindo ataques de *Cross-Site Scripting* (XSS) e comprometendo dados do usuário. Por fim, tais vulnerabilidades afetam a integridade e confiabilidade do sistema, podendo resultar em manipulação de dados.

A biblioteca jQuery (versão 1.10.2), usada no DSpace, apresenta vulnerabilidade conhecida conforme o repositório *Common Vulnerabilities and Exposures* (CVE), mantido pela Corporação MITRE desde 1999 (CVE, 2022). Esse repositório é um dos bancos de dados mais populares que identifica, define e cataloga vulnerabilidades de segurança cibernética divulgadas publicamente (PANDA; RASS; MOSCHOYIANNIS; LIANG; LOUKAS; PANAOUSIS, 2022).

Essa vulnerabilidade é solucionada com a atualização das bibliotecas mencionadas para versões que não possuam mais alguma CVE detectada e que não impactam no funcionamento da aplicação web. Assim, observou-se que a versão adequada é a 3.6.1[6] e, para evitar que algumas funcionalidades do layout se quebrem, é necessário atualizar a versão do Bootstrap para 3.4.1[7].

### 4.2.2 Ausência de tokens anti-CSRF

CSRF é uma vulnerabilidade que permite a falsificação de solicitações entre sites. Esse tipo de ataque pode forçar uma vítima (usuário) a enviar uma solicitação HTTP para um destino alvo, sem seu conhecimento ou intenção, a fim de se passar por um usuário autêntico com permissões, na tentativa de obter privilégios e acessos às operações do sistema (ZAP, c2023a). Desse modo, soluciona-se essa vulnerabilidade com a implementação do token em todos os formulários *HyperText Markup Language* (HTML), no elemento de formulário *<forms>* no DSpace.

### 4.2.3 Content Security Policy

A política de segurança de conteúdo (CSP) opera como uma defesa em profundidade, é uma camada adicional de segurança que ajuda a detectar certos tipos de ataques. Desenvolvedores usam-na para bloquear suas aplicações de várias maneiras, prevenindo os riscos de vulnerabilidades de injeção de conteúdo, como scripts entre sites e *Cross-Site Scripting* (XSS) (W3C, 2016). Essa camada ajuda a evitar a desfiguração de sites e distribuição de softwares maliciosos (STAMM; STERNE; MARKHAM, 2010). Além disso, reduz danos que uma injeção maliciosa pode causar, mas não substitui a validação de entrada cuidadosa e a codificação de saída. Para habilitar o CSP, é necessário configurar o servidor web para retornar o cabeçalho HTTP *Content-Security-Policy* e inserir a tag *<meta>* no cabeçalho da aplicação web.

### 4.2.4 Cabeçalho *X-Content-Type-Options* ausente

O X-Content-Type-Options é um cabeçalho HTTP, uma camada de segurança para proteger contra vulnerabilidades de *MIME-sniffing*. Essa vulnerabilidade permite que o invasor disfarce um arquivo HTML como um tipo de arquivo diferente, por exemplo imagens

[6] Conforme consultado em https://security.snyk.io/package/npm/jquery
[7] Conforme consultado em https://security.snyk.io/package/npm/bootstrap

com formatos JPEG, PNG etc., ou arquivos compactados como zip, dentre outros.

Versões antigas de navegadores executam essa prática para determinar tipos de mídia de internet, o tipo MINE, possivelmente fazendo com que o corpo da resposta seja interpretado e exibido como um tipo de conteúdo diferente do tipo de conteúdo declarado (ZAP, c2023b). Esse mecanismo de segurança examina o conteúdo da resposta em vez de confiar no cabeçalho Content-Type. Portanto, soluciona-se essa vulnerabilidade habilitando o cabeçalho X-Content-Type-Options no servidor (nginx ou apache2) para bloquear possíveis *MIME-sniffing*.

### 4.2.5 Cabeçalho *anti-clickjacking* ausente

O ataque *clickjacking*, também conhecido como ataque de reparação de interface do usuário, induz a vítima (usuário) a clicar em páginas web aparentemente inofensivas que foram incorporadas indevidamente em elementos HTML, como <*frame*>, <*iframe*>, <*embed*> ou <*object*>. Essa vulnerabilidade pode trazer sérios danos, como expor informações confidenciais, ou até mesmo permitir que o invasor assuma o controle do computador do usuário.

Existem mecanismos de segurança para defender-se contra esses tipos de ataques, é possível impedir que o navegador renderize a página web maliciosa usando cabeçalhos HTTP como *X-Frame-Options* ou *Content Security Policy*, mencionado anteriormente. Uma forma de solucionar essa vulnerabilidade é habilitando o cabeçalho *X-Frame-Options* a nível de servidor, há solução tanto para o nginx quanto para o apache2. A solução visa a identificar se o navegador deve ou não renderizar páginas web dentro de elementos HTML (ROSS; GONDROM, 2013). Essa prática protege contra possíveis ataques de *clickJacking*, assegurando que os conteúdos não sejam incorporados em outros sites (MDN, 2023).

## 4.3 Correção

As vulnerabilidades de segurança identificadas pelo *scanner* OWASP ZAP foram corrigidas no DSpace. As soluções foram aplicadas com base no estudo exploratório e nas recomendações da ferramenta de varredura. O DSpace é uma ferramenta robusta desenvolvida em linguagem de programação Java. Assim, foram realizadas correções de segurança em cerca de 130 arquivos específicos do software, abrangendo linguagens de programação como Java, HTML, *Cascading Style Sheets* (CSS) e *JavaScript*. Além disso, o

servidor web Apache Tomcat versão 8 foi configurado para atender as diretrizes das soluções propostas.

### 4.3.1 Biblioteca JS vulnerável

Apesar de apresentar o risco mais alto identificado pela ferramenta OWASP ZAP, para corrigir essa vulnerabilidade basta atualizar as bibliotecas em questão, se possível, para suas versões mais atuais disponíveis. Essas atualizações incluem correções de segurança que abordam tanto as vulnerabilidades identificadas quanto melhorias sistemáticas nas bibliotecas.

As bibliotecas identificadas com vulnerabilidades CVE foram atualizadas da seguinte forma: Bootstrap para a versão 3.4.1 e jQuery para a versão 3.6.1. No caso do Bootstrap, a versão mais atual não foi adotada para garantir a continuidade da operabilidade do software, mas a versão escolhida não possui vulnerabilidades conhecidas.

### 4.3.2 Ausência de tokens anti-CSRF

A implementação do token em todos os formulários HTML (métodos GET e POST) nos sistemas informacionais teve como objetivo solucionar a vulnerabilidade de Token anti-CSRF ausente e fortalecer a segurança do sistema. A seguir, apresentamos um exemplo simplificado de input do tipo *hidden* (campo oculto) que ilustra a implementação do token em um formulário HTML:

```
<input type="hidden" name="_token" value="c2945dc5f1dd8ae0b5f9bf49daa84f29">
```

Ao enviar o formulário, o token é incluído nas solicitações GET ou POST, permitindo que o servidor verifique a validade do token ao processar a requisição. Caso o token esteja ausente ou seja inválido, o servidor pode rejeitar a solicitação como uma medida de segurança.

### 4.3.3 Content Security Policy (CSP)

A habilitação do CSP requer a configuração do servidor web para retornar o cabeçalho HTTP Content-Security-Policy e a inserção da tag <meta> no cabeçalho da aplicação web. Para aplicar a mesma configuração do Content Security Policy (CSP) em servidores web como Nginx ou Apache, é necessário realizar alterações na configuração do servidor correspondente. No caso do Nginx, a configuração do CSP é adicionada ao arquivo

de configuração do servidor ou no bloco de configuração do site específico, com o seguinte comando *add_header*:

```
add_header Content-Security-Policy "default-src 'none'; style-src 'self' 'unsafe-inline' 'unsafe-eval'; script-src 'self' 'unsafe-inline'; connect-src 'self'; img-src 'self' data:; font-src 'self'; frame-ancestors 'self'; form-action 'self';"
```

No Apache, é possível adicionar a configuração do CSP ao arquivo *.htaccess* no diretório raiz do site ou diretamente no arquivo de configuração do Apache. A diretiva *Header* é usada nesse caso:

```
Header always set Content-Security-Policy "default-src 'none'; style-src 'self' 'unsafe-inline' 'unsafe-eval'; script-src 'self' 'unsafe-inline'; connect-src 'self'; img-src 'self' data:; font-src 'self'; frame-ancestors 'self'; form-action 'self';"
```

Além da configuração do servidor, é necessário adicionar a tag *<meta>* ao cabeçalho de cada página da aplicação web. A tag inserida no cabeçalho de ambas as aplicações web foi:

```
<meta http-equiv="Content-Security-Policy" content="default-src \'none\'; style-src \'self\' \'unsafe-inline\' \'unsafe-eval\'; script-src \'self\' \'unsafe-inline\'; connect-src \'self\'; img-src \'self\' data:; font-src \'self\'; form-action \'self\';" />
```

A CSP definida restringe o acesso a recursos externos, permitindo apenas recursos provenientes do próprio domínio, com definições específicas para estilos inline (CSS), scripts, conexões de rede, imagem, fontes e ação de formulários.

### 4.3.4 Cabeçalho *X-Content-Type-Options* ausente

Para corrigir a vulnerabilidade de cabeçalho *X-Content-Type-Options* ausente, a solução é configurar o servidor web para enviar o cabeçalho *X-Content-Type-Option*. As soluções implementadas para o servidor Nginx e Apache são mostradas a seguir.

Comando add_header para web server Nginx:

```
add_header X-Content-Type-Options nosniff
```

Diretiva Header no .htaccess para web server Apache

```
Header set X-Content-Type-Options: "nosniff"
```

Em ambos os casos, o valor "nosniff" é usado, o que impede que o navegador execute o *MIME sniffing*.

### 4.3.5 Cabeçalho anti-clickjacking ausente

Uma forma de solucionar a vulnerabilidade de cabeçalho *anti-clickjacking* ausente é habilitando o cabeçalho *X-Frame-Options* no servidor web. As soluções implementadas para o servidor Nginx e Apache são mostradas a seguir.

Comando add_header para web server Nginx:

```
add_header X-Frame-Options: "sameorigin"
```

Diretiva Header no .htaccess para web server Apache

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

Em ambos os casos, o valor "sameorigin" permite que o conteúdo seja exibido em um frame somente se estiver no mesmo domínio.

## 5 CONSIDERAÇÕES FINAIS

A análise das vulnerabilidades encontradas no software DSpace com o auxílio do *scanner* OWASP ZAP revelou informações cruciais para a melhoria da segurança do sistema. Por meio de um estudo exploratório, as vulnerabilidades foram analisadas e soluções foram propostas. As correções propostas foram aplicadas, o que acarretou atualizações de bibliotecas JavaScript, ajustes nas configurações do servidor web e mudanças diretas no código fonte da aplicação.

O caráter aplicado do estudo, restrito ao software DSpace, apresenta de forma geral um modelo para tornar a aplicação mais segura, na medida em que tem sido utilizada por grande parte das universidades e alguns órgãos governamentais. Apresentando o uso do OWASP ZAP na detecção de vulnerabilidades e suas correções, torna o estudo um exemplo para que outras iniciativas possam reproduzir e melhorar a segurança do repositório ou da biblioteca digital.

Portanto, a correção de vulnerabilidades é uma medida proativa para mitigar riscos de segurança, uma vez que, a ausência de correções pode levar a explorações bem-sucedidas que resultam em violações de dados no DSpace. Vale ressaltar que a correção de vulnerabilidades não se limita a uma simples tarefa técnica, trata-se, na verdade, de uma estratégia essencial para proteger a integridade, confiabilidade dos dados e a reputação de uma organização. Ao implementar correções de vulnerabilidade no DSpace de forma eficaz, as universidades e órgãos de governo podem fortalecer sua postura de segurança e reduzir

significativamente os riscos associados a ameaças cibernéticas. Essa abordagem proativa é fundamental para manter a segurança em um ambiente digital em constante evolução.

**REFERÊNCIAS**

ALAZMI, Suliman; LEON, Daniel Conte de. Customizing OWASP ZAP: a proven method for detecting SQL injection vulnerabilities. *In:* IEEE INTERNATIONAL CONFERENCE ON BIG DATA SECURITY ON CLOUD (BIG DATA SECURITY), IEEE INTL CONFERENCE ON HIGH PERFORMANCE AND SMART COMPUTING, (HPSC) AND IEEE INTL CONFERENCE ON INTELLIGENT DATA AND SECURITY (IDS), 9., 2023, New York. **Proceedings [...]**. New York: IEEE, 2023. Disponível em: https://ieeexplore.ieee.org/document/10132146. Acesso em: 01 jul. 2023.

ALBAHAR, Marwan; ALANSARI, Dhoha; JURCUT, Anca. An empirical comparison of pen-testing tools for detecting web app vulnerabilities. **Electronics**, [*s. l.*], v. 11, n. 19, p. 1-25, 2022. Disponível em: https://doi.org/10.3390/electronics11192991. Acesso em: 01 jul. 2023.

CVE. **Homepage**, 2022. Disponível em: https://cve.mitre.org/. Acesso em: 01 jul. 2023.

JOBIN, T. J.; BABU, Karthika Suresh. Owasp Zed Attack Proxy. *In:* NATIONAL CONFERENCE ON EMERGING COMPUTER APPLICATIONS (NCECA), 3., 2021, Kerala, Índia. **Proceedings ]...]**. Kerala: Amal Jyothi College of Engineering, 2021.

MACÊDO, Diego José; SHINTAKU, Milton; BRITO, Ronnie Fagundes de. Dublin Core usage for describing documents in Brazilian Government digital libraries. *In*: INTERNATIONAL CONFERENCE ON DUBLIN CORE AND METADATA APPLICATIONS, 2015, São Paulo. **Proceedings [...]**. São Paulo: UNESP: 2015. Disponível em: https://dcpapers.dublincore.org/pubs/article/view/3768. Acesso em: 01 jul. 2023.

MAKINO, Yuma; KLYUEV, Vitaly. Evaluation of web vulnerability scanners. *In:* INTERNATIONAL CONFERENCE ON INTELLIGENT DATA ACQUISITION AND ADVANCED COMPUTING SYSTEMS: TECHNOLOGY AND APPLICATIONS (IDAACS), 8., 2015, Warsaw, Poland**. Proceedings [...].** Warsaw: IEEE, 2015. Disponível em: 10.1109/IDAACS.2015.7340766. Acesso em: 01 jul. 2023.

MDN. **X-Frame-Options**, 2023. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Headers/X-Frame-Options. Acesso em: 01 jul. 2023.

NURBOJATMIKO; LATHIFAH, Ari; AMRI; Faaza Bil; ROSIDAH, Ani. Security vulnerability analysis of the sharia crowdfunding website using OWASP-ZAP. *In:* INTERNATIONAL CONFERENCE ON CYBER AND IT SERVICE MANAGEMENT (CITSM), 10., 2022, Indonesia. **Proceedings [...].** Yogyakarta: IEEE, 2022. Disponível em: https://ieeexplore.ieee.org/document/9935837/authors#authors. Acesso em: 01 jul. 2023.

OWASP. **OWASP Zap 2.11**: getting started guide. [*S. l.*]: OWASP, c2023. Disponível em: https://www.zaproxy.org/pdf/ZAPGettingStartedGuide-2.11.pdf. Acesso em: 01 jul. 2023.

PANDA, Sakshyam; RASS, Stefan; MOSCHOYIANNIS, Sotiris; LIANG, Kai-Rong; LOUKAS, George; PANAOUSIS, Emmanouil. HoneyCar: a framework to configure honeypot

vulnerabilities on the internet of vehicles. **IEEE Access**, [*s. l.*], vol. 10, p. 104671-104685, 2022. Disponível em: 10.1109/ACCESS.2022.3210117. Acesso em: 01 jul. 2023.

ROSS, David; GONDROM, Tobias. Http header field x-frame-options. **Request for Comments**, [*s. l.*], v. 7034, p. 1-14, 2013. Disponível em: https://www.rfc-editor.org/rfc/pdfrfc/rfc7034.txt.pdf. Acesso em: 01 jul. 2023.

SHINTAKU, Milton; VECHIATO, Fernando Luiz. Histórico do uso do DSpace no Brasil com foco na tecnologia. **Revista Informação na Sociedade Contemporânea**, Natal, v. 2, p. 1-16, jan./jun. 2018. Disponível em: https://periodicos.ufrn.br/informacao/article/view/13097/9501. Acesso em: 01 jul. 2023.

STAMM, Sid; STERNE, Brandon; MARKHAM, Gervase. Reining in the web with content security policy. *In:* INTERNATIONAL CONFERENCE ON WORLD WIDE WEB, 19., 2010, Raleigh, North Carolina. **Proceedings [...].** New York: Association of Computing Machinery, 2010. Disponível em: https://doi.org/10.1145/1772690.1772784. Acesso em: 01 jul. 2023.

W3C. **Content Security Policy Level 2**, 2016. Disponível em: https://www.w3.org/TR/CSP2/. Acesso em: 01 jul. 2023.

ZAP. **Absence of anti-csrf tokens**, c2023a. Disponível em: https://www.zaproxy.org/docs/alerts/10202/. Acesso em: 01 jul. 2023.

ZAP. **X-Content-Type-Options header missing**, c2023b. Disponível em: https://www.zaproxy.org/docs/alerts/10202/. Acesso em: 01 jul. 2023.