FACULDADE DE TECNOLOGIA DE AMERICANA "MINISTRO RALPH BIASI"
Curso Superior de Tecnologia em Segurança da Informação

Thiago Henrique Rodrigues Edmundo

IMPLANTAÇÃO DE TESTE DE VULNERABILIDADES EM APLICAÇÕES WEB SEGUINDO METODOLOGIA OWASP

Americana, SP
2021

FACULDADE DE TECNOLOGIA DE AMERICANA "MINISTRO RALPH BIASI"

Curso Superior de Tecnologia em Segurança da Informação

THIAGO HENRIQUE RODRIGUES EDMUNDO

IMPLANTAÇÃO DE TESTE DE VULNERABILIDADES EM APLICAÇÕES WEB SEGUINDO METODOLOGIA OWASP

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Segurança da Informação, sob a orientação do Prof. Me. Edson Roberto Gaseta

Área de concentração: Segurança da Informação.

.

Americana, SP

2021

THIAGO HENRIQUE RODRIGUES EDMUNDO

IMPLANTAÇÃO DE TESTE DE VULNERABILIDADES EM APLICAÇÕES WEB SEGUINDO METODOLOGIA OWASP

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Segurança da Informação, pelo CEETEPS/Faculdade de Tecnologia – Fatec/ Americana.
Área de concentração: Segurança da Informação

Americana, 09 de dezembro de 2021.

Banca Examinadora:

Edson Roberto Gaseta (Presidente)
Mestre
Fatec Americana

Renato Kraide Soffner (Membro)
Doutor
Fatec Americana

Elton Rafael Mauricio da Silva Pereira (Membro)
Mestre
Fatec Americana

RESUMO

Este trabalho traz uma breve explicação sobre as aplicações web e como impactam no cotidiano da população, sendo utilizadas por mais da metade da população mundial, facilitando o dia a dia, tanto das pessoas quanto das empresas e criando um mundo cada vez mais globalizado. Grande parte das aplicações web armazenam dados confidenciais de usuários e até mesmo de organizações, diante disso, é necessária uma conscientização sobre a importância de testar a segurança dos aplicativos antes de colocá-los em ambiente de produção. O objetivo principal deste trabalho é apresentar, de forma prática, a ferramenta OWASP ZAP, demonstrando os resultados de uma varredura em busca de fraquezas em um produto em fase de desenvolvimento e trazer referências do impacto que essas vulnerabilidades podem causar quando exploradas e como mitigá-las.

Palavras-chave: vulnerabilidades, impacto, OWASP, aplicações, teste.

ABSTRACT

This work brings a brief explanation about web applications and how they impact the daily life of the population, being used by more than half of the world population, facilitating the day to day, both for people and companies and creating an increasingly globalized world. A large part of the web applications store confidential data of users and even of organizations, therefore, it is necessary to raise awareness about the importance of testing the security of applications before putting them into production environment. The main objective of this work is to present, in a practical way, the OWASP ZAP tool, demonstrating the results of a scan in search of weaknesses in a product under development and bringing references to the impact that these vulnerabilities can cause when exploited and how to mitigate them.

Keywords: vulnerabilities, impact, OWASP, applications, test.

LISTA DE ILUSTRAÇÕES

FIGURAS

TABELAS

SUMÁRIO

1. INTRODUÇÃO

O uso da internet transformou o dia a dia das pessoas, com a globalização e o aumento da tecnologia, a sociedade possui mais facilidades de acesso. A utilização de serviços *web* permite realizar transações que eram demoradas e complexas, de maneira mais simples e rápida, em sua própria residência a um clique de distância, trazendo economia de tempo, evitando burocracia, filas e custos com a movimentação.

Desde compras online, uma simples troca de mensagens instantâneas, sistemas de internet banking ou até mesmo acesso a serviços de *streaming*, as aplicações *web* sempre estão presentes e os hackers enxergam nelas oportunidades de ataques através de vulnerabilidades que podem ser encontradas com uso de *scans* (varreduras).

A metodologia utilizada no trabalho segue os padrões disponibilizados pela *Open Web Application Security Project* (OWASP), uma organização sem fins lucrativos que busca um mundo cibernético mais seguro, além disso apresenta uma definição das dez vulnerabilidades mais exploradas no mundo *web* e disponibiliza a ferramenta que foi utilizada para as verificações de segurança da aplicação *web* testada.

As aplicações *web* geralmente solicitam aos usuários informações pessoais para cadastro e uso, diante disto, é necessário buscar garantir a segurança dessas informações, já que muitos ataques são voltados contra a confidencialidade.

Este trabalho tem como principal objetivo demonstrar a importância, de forma prática, da implantação da ferramenta OWASP *Zed Attack Proxy* (ou simplesmente, OWASP ZAP) em uma empresa que cedeu o acesso para testar um produto que está em fase de desenvolvimento e encontrar as possíveis vulnerabilidades que podem existem, evitando assim possíveis danos que ataques *hackers* podem gerar.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Segurança da Informação

A Segurança da Informação é de extrema importância para qualquer organização, como podemos encontrar na NBR 27002 (2013, p.5):

> "Organizações de todos os tipos e tamanhos (incluindo o setor privado e público, organizações comerciais e sem fins lucrativos), coletam, processam, armazenam e transmitem informações em diferentes formatos, incluindo o eletrônico, físico e verbal (por exemplo, conversações e apresentações). O valor da informação vai além das palavras escritas, números e imagens: conhecimento, conceitos, ideias e marcas são exemplos de formas intangíveis da informação. Em um mundo interconectado, a informação e os processos relacionados, sistemas, redes e pessoas envolvidas nas suas operações são informações que, como outros ativos importantes, têm valor para o negócio da organização e, consequentemente, requerem proteção contra vários riscos."

De acordo com Sêmola (2003, p. 45), a segurança da informação consiste na preservação de seus três princípios básicos, sendo estes:

> Confidencialidade: toda e qualquer informação deve ser protegida, de acordo com o grau de sigilo estipulado, buscando a limitação de acesso e uso apenas às pessoas para quem elas são destinadas.
> Disponibilidade: toda informação adquirida ou produzida, seja por um indivíduo ou por uma instituição, deve estar disponível aos seus usuários quando os mesmos delas necessitem.
> Integridade: toda informação deve ser mantida no estado em que ela foi disponibilizada pelo seu proprietário, buscando protegê-las de alterações indevidas, sejam estas, intencionais ou acidentais.

2.2 Conceito de páginas WEB

A criação das páginas web teve início devido a necessidade de facilitar a troca de informações e de conhecimento. Em 1989, a *World Wide Web* (o famoso WWW) foi concebida, pelo engenheiro de sistemas inglês Tim Berners-Lee, segundo Aquino (2006, p. 5):

> "[..] surgiu em função de um problema de troca de informações sofrido pelos pesquisadores do CERN. O centro era composto por diversos pesquisadores em diferentes projetos de investigação, cujo trabalho nem sempre era desenvolvido dentro do instituto."

Antes do surgimento das páginas *web*, os documentos, informações e o conhecimento eram compartilhados por papel, e isso gerava atrasos e dificultava o trabalho dos pesquisadores do CERN (Centro Europeu de Pesquisa Nuclear), sendo muito difícil e trabalhoso organizar e compartilhar tantos papéis.

Na internet juntamente com o hipertexto, Tim Berners-Lee viu a oportunidade de criar algo que facilitaria a troca de informações e dos resultados adquiridos nas pesquisas, e assim, deixando de lado a logística que os papéis exigiam. Conforme explica Aquino (2006, p. 5):

> "[..] a Internet e o hipertexto já eram difundidos e utilizados no meio e assim, Berners-Lee ficou responsável por encontrar uma plataforma eletrônica para a troca de informações entre os pesquisadores do CERN. Através de muito trabalho, o inglês acabou inventando o HTML11, um novo formato para armazenar documentos no disco rígido de um computador que tivesse acesso permanente à Internet."

As páginas web são documentos que fazem parte de um site e que podem ser acessados e interpretados por navegadores através da rede. A base do desenvolvimento, "o esqueleto", das páginas *web* é o HTML, que é uma linguagem de marcação. Para se entender melhor o conceito de páginas *web*, é necessário entender o básico de URL, HTML, HTTP e links, podemos verificar abaixo:

- URL: as URLs (*Uniform Resource Locator*), em português: "Localizador Padrão de Recursos", são os identificadores únicos e exclusivos das páginas *web*, segundo a definição que encontramos na RFC 1738 (BERNERS-LEE; MASINTER; MCCAHILL, 1994), a URL é uma sequência de caracteres que seguem determinada regra e são utilizadas para identificar e localizar recursos.

- HTML: o HTML é uma linguagem de marcação, ou seja, não é uma linguagem de programação, é utilizada apenas formatação de texto, inserção de imagem, vídeos, áudios e links. A DevMedia (FEITOSA, 2012) define o HTML como uma linguagem para publicação de conteúdo que surgiu do conceito do hipertexto, que são vários elementos conectados e que formam uma rede de informação.

- HTTP: o Hypertext Transfer Protocol (HTTP) é um protocolo de transferência, ele permite o acesso às páginas *web* através da

inserção da URL no navegador, de acordo com Tanenbaum (2003) “O protocolo de transferência utilizado em toda a *World Wide Web* é o HTTP (*HyperText Transfer Protocol*). Ele especifica as mensagens que os clientes podem enviar aos servidores e que respostas eles receberão.”

- Links: os links são elementos clicáveis, que direciona o navegador para determinadas URLs, ou seja, é uma ligação entre conteúdos.

Com o passar dos anos, a web deixou de ser apenas documentos de hipertexto relacionados através de links, atualmente, fazemos uso das aplicações web, permitindo acessar inúmeros serviços e conteúdo. Através de aplicações web podemos pagar contas, acessar serviços de streaming, fazer compras online, acessar redes sociais, fechar negócios, trocar mensagens instantâneas com quem está do outro lado do mundo, enfim, existem diversos tipos de aplicações web. Conforme a definição de Pinto e Stuttard (2011, p.23):

> Por "aplicativos da web", queremos dizer aqueles que são acessados usando um navegador da web para se comunicar com um servidor da web. [...] de diferentes tecnologias, como bancos de dados, sistemas de arquivo e serviços da web”.

2.2.1 Aplicações Web

As aplicações web são programas que podem ser acessados por um navegador web, não precisam ser instaladas na máquina do usuário, e podem ser definidas como:

> "[...] qualquer software baseado em web que realize ações (funcionalidades) de acordo com uma entrada de usuário e que normalmente interaja com sistemas de *backend*. Quando um usuário interage com um web site para realizar alguma ação, por exemplo, fazer login, fazer compras ou acessar o banco, temos uma aplicação web. (Pauli 2015, p. 26)"

O uso de aplicações web faz parte do nosso dia a dia, seja para uso pessoal ou no trabalho, estima-se que mais da metade da população mundial tem acesso à internet, esse número vem aumentando, conforme o levantamento *da We Are Social HootSuite* (2021) “4,66 bilhões de pessoas em todo o mundo usam a Internet em

janeiro de 2021, um aumento de 316 milhões (7,3%) desde então no ano passado. A penetração global da Internet agora é de 59,5%.” Esses dados tornam evidente a importância da internet para a sociedade, estamos conectados e buscando cada vez mais as facilidades que a internet nos proporciona. Contudo, hackers viram oportunidades nas páginas e aplicações web, isso pode ser comprovado pelo Cert.br, no primeiro semestre de 2020 ocorreram 318.697 incidentes de segurança, sendo 8.811 relacionados a web, como podemos observar na tabela abaixo:

Tabela 1 - Incidentes Reportados ao CERT.br - janeiro a junho de 2020

Mês	Total	worm (%)		**dos** (%)		invasão (%)		web (%)		**scan** (%)		**fraude** (%)		outros (%)	
jan	**35094**	6703	19	3579	10	54	0	999	2	20704	59	2701	7	354	1
fev	**40229**	8215	20	7424	18	52	0	897	2	21184	52	2319	5	138	0
mar	**63313**	9316	14	16837	26	93	0	1045	1	31633	49	3111	4	1278	2
abr	**55255**	7680	13	6092	11	110	0	1458	2	36287	65	3537	6	91	0
mai	**59820**	10698	17	4815	8	122	0	2225	3	38389	64	3512	5	59	0
jun	**64986**	13033	20	7417	11	184	0	2187	3	39243	60	2844	4	78	0
Total	318697	55645	17	46164	14	615	0	8811	2	187440	58	18024	5	1998	0

Fonte: CERT.BR (2020)

De acordo com Eoin Keary (2020) o problema do software inseguro é talvez o desafio técnico mais importante de nosso tempo. A dramática ascensão de aplicativos da web que permitem negócios, redes sociais etc., apenas aumentou os requisitos para estabelecer uma abordagem robusta para escrever e proteger nossa Internet, aplicativos da Web e dados.

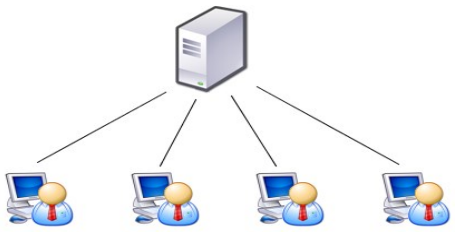
2.2.2 Arquitetura cliente-servidor

Um usuário quando faz a solicitação de uma página web pela internet faz uso da arquitetura cliente-servidor, segundo Kurose e Ross (2013, p. 62):

> "Em uma arquitetura cliente-servidor há um hospedeiro sempre em funcionamento, denominado servidor, que atende a requisições de muitos outros hospedeiros, denominados clientes. Um exemplo clássico é a aplicação Web na qual um servidor Web que está sempre em funcionamento atende a solicitações de navegadores de hospedeiros clientes. Quando recebe uma requisição de um objeto de um hospedeiro cliente, um servidor Web responde enviando o objeto solicitado. Observe que, na arquitetura cliente-servidor, os clientes não se comunicam diretamente uns com os outros; por exemplo, na aplicação Web, dois navegadores não se comunicam de modo direto."

Na Figura 1, vemos um exemplo de arquitetura cliente-servidor:

Figura 1 – Arquitetura cliente/servidor



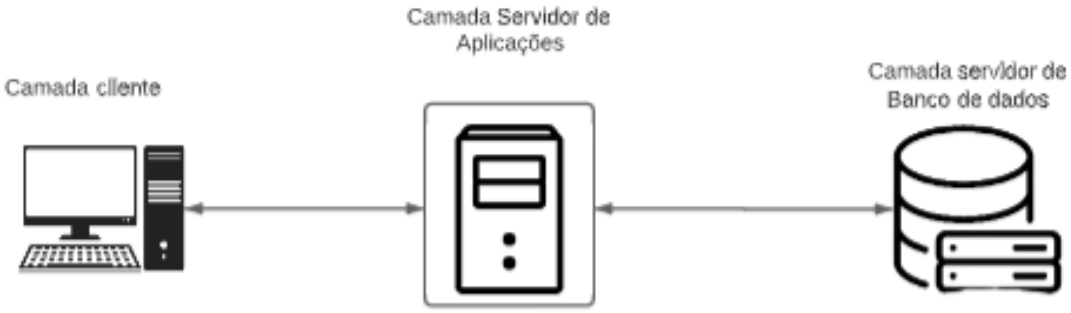
Fonte: arquiserv (2012)

As aplicações web atuais geralmente utilizam um modelo de arquitetura de três camadas. Nesse tipo de arquitetura, a interface do usuário é desassociada da funcionalidade e das informações. Como exemplo podemos citar a arquitetura Modelo-Visão-Controlador (MVC), conforme Pressman (2011, p. 349):

> "O modelo (algumas vezes conhecido como "objeto-modelo") contém todo o conteúdo e a lógica de processamento específicos à aplicação, inclusive todos os objetos de conteúdo, acesso a fontes de dados/informações externas e toda a funcionalidade de processamento específica para a aplicação. A visão contém todas as funções específicas à interface e possibilita a apresentação do conteúdo e lógica de processamento, inclusive todos os objetos de conteúdo, acesso a fontes de dados/informações externas e toda a funcionalidade de processamento requerida pelo usuário final. O controlador gerencia o acesso ao modelo e à visão e coordena o fluxo de dados entre eles. Em uma *WebApp*, "a visão é atualizada pelo controlador com dados do modelo baseados nas informações fornecidas pelos usuários"

Na Figura 2 é mostrado um exemplo de uma arquitetura de três camadas:

Figura 2 – Exemplo de arquitetura de 3 camadas



Fonte: Autor (2021)

2.2 Vulnerabilidades e ameaças

A OWASP (2014) define vulnerabilidade como uma falha ou uma fraqueza no aplicativo, que pode ser uma falha de design ou um bug de implementação, que permite que um invasor cause danos às partes interessadas de um aplicativo. As partes interessadas incluem o proprietário do aplicativo, usuários do aplicativo e outras entidades que dependem do aplicativo.

Segundo a OWASP (2014, p.27) uma ameaça pode ser considerada qualquer coisa que pode causar um incidente ou algum dano, através da exploração de uma vulnerabilidade, podemos concluir que, as vulnerabilidades são os “caminhos” para que uma ameaça cause um dano. Conforme Sêmola (2014, p. 18):

> “A todo instante, os negócios, seus processos e ativos físicos, tecnológicos e humanos, são alvo de investidas de ameaças de toda ordem, que buscam identificar um ponto fraco compatível, uma vulnerabilidade capaz de potencializar sua ação. Quando essa possibilidade aparece, a quebra de segurança é consumada.”

Podemos exemplificar, um ataque de injeção SQL, onde através de uma fraqueza no código (vulnerabilidade) um hacker (ameaça) consegue validar uma entrada em um banco de dados. Os ataques de injeção SQL serão mais detalhados posteriormente.

2.2.1 Ataques

O Cert.br define ataque como “Qualquer tentativa, bem ou malsucedida, de acesso ou uso não autorizado de um serviço, computador ou rede.” De acordo com Shirey (2000) um ataque à segurança do sistema derivado de uma ameaça inteligente, ou seja, um ato inteligente que é uma tentativa deliberada (especialmente no sentido de um método ou técnica) de escapar aos serviços de segurança e violar a política de segurança de um sistema.

Existem diversos meios e ferramentas que os atacantes utilizam para conseguir violar a segurança da informação. Podemos dividir os tipos de ataques em dois grandes grupos, os ataques passivos e os ataques ativos, segundo Vieites (2009) quando se trata de estudar os diferentes tipos de ataques de computador, podemos nos diferenciar em primeiro lugar entre os ataques ativos, que causam mudanças nas informações e no estado dos recursos do sistema, e ataques passivos, que se limitam a registrar o uso de recursos e/ou para acessar as informações armazenadas ou transmitidas pelo sistema.

De acordo com a *Living-in-belgium* (2021) os ataques passivos violam a confidencialidade da informação, uma vez que, este tipo de ataque busca ler a informação e fazer uso dela, sem que a organização saiba. Em contrapartida, os ataques passivos atentam contra a disponibilidade e a integridade da informação, já que este tipo de ataque busca alterar as informações, modificar recursos de sistemas ou torná-los indisponíveis.

2.3 OWASP

A *Open Web Application Security Project* (OWASP) é uma organização sem fins lucrativos e se define como uma comunidade mundial livre e aberta focada em melhorar a segurança de aplicações. De acordo com Eoin Keary (2020), no The Open Web Application Security Project (OWASP), estamos tentando tornar o mundo onde software inseguro é a anomalia, não a norma e para alcançar tal objetivo, Keary complementa: Um mundo sem alguns padrões mínimos em termos de engenharia e tecnologia é um mundo em caos.

A OWASP disponibiliza matérias, artigos, metodologias, ferramentas e documentação de maneira gratuita e livre, buscando promover o conhecimento sobre

a segurança de aplicativos web para qualquer um que precisar ou que procure a aprender sobre o assunto.

De acordo com a OWASP (2014) a missão da organização é fazer segurança de aplicativos "visível", para que pessoas e organizações possam tomar decisões informadas sobre riscos de segurança de aplicativos. Cada um é livre para participar da OWASP e de todos os nossos os materiais estão disponíveis sob uma licença de software livre e aberta.

2.3.1 OWASP Top 10

Todos os ataques cibernéticos são feitos através da exploração de vulnerabilidades, e isso não poderia ser diferente nos ataques voltados às aplicações web. De acordo com Harpreet Singh e Himanshu Sharma (2020) é necessário executar um scan no alvo para achar as vulnerabilidades, uma que foram encontradas as vulnerabilidades, é necessário avaliá-las, descobrir qual é o tipo da vulnerabilidade ou se não é um falso positivo.

A OWASP define as dez vulnerabilidades mais exploradas em aplicações web, segundo a OWASP (2017) O OWASP Top 10 é um documento padrão de conscientização para desenvolvedores e segurança de aplicativos da web. Ele representa um amplo consenso sobre os riscos de segurança mais críticos para aplicativos da web.

Nesse trabalho iremos detalhar nos próximos capítulos os seguintes ataques: os de injeção e quebra de autenticação, sendo ataques muito comuns em aplicações web.

2.3.1.1 Injeção (A1:2017-Injection)

Segundo Lopes (2018):

> “A injeção é uma classe inteira de ataques que dependem da injeção de dados em um aplicativo da Web para facilitar a execução ou interpretação de dados mal-intencionados de maneira inesperada. “

De acordo com a OWASP (2017) estes ataques ocorrem quando dados não confiáveis são enviados para um interpretador como parte de um comando ou de uma consulta legítima, com isso, os dados dos atacantes podem enganar o interpretador levando-o a executar comandos não pretendidos ou dar acesso aos dados sem a devida autorização.

Nesse tipo de ataque, são exploradas vulnerabilidades encontradas a partir de scanners, segundo a OWASP (2017, p. 9) pode resultar em perda ou corrupção de dados, falha de responsabilização, ou negação de acesso. A injeção pode, às vezes, levar ao controle total do sistema. O impacto no negócio depende das necessidades de proteção da aplicação ou dos seus dados.

Um dos ataques mais comuns de injeção é conhecido como *SQL Injection,* ou simplesmente, Injeção SQL (*Structured Query Language).* Segundo a Devmedia (2007) “O SQL Injection é uma técnica de ataque baseada na manipulação do código SQL, que é a linguagem utilizada para troca de informações entre aplicativos e bancos de dados relacionais.”

Como afirma Vieites (2009) o ataque de injeção de SQL é produzido quando não é devidamente filtrado as informações enviadas pelo usuário. Um usuário malicioso pode incluir e executar textos que representam novas instruções SQL que o servidor não deve aceitar. Este tipo de ataque é independente do sistema de banco de dados subjacente, pois depende exclusivamente de um validação inadequada dos dados de entrada.

2.3.1.2 Quebra de autenticação

As quebras de autenticação e gerenciamento de sessão ocorrem quando um atacante consegue acesso as credenciais de outros usuários, de acordo com a OWASP (2017) as funções da aplicação que estão relacionadas com a autenticação e gestão de sessões são muitas vezes implementadas incorretamente, permitindo que um atacante possa comprometer *passwords*, chaves, tokens de sessão, ou abusar doutras falhas da implementação que lhe permitam assumir a identidade de outros utilizadores (temporária ou permanentemente).

Para Pinto e Stuttard (2011, p. 7) a quebra de autenticação envolve casos em que o aplicativo falha em proteger adequadamente o acesso aos seus dados e

funcionalidade, potencialmente permitindo que um invasor visualize dados confidenciais de outros usuários mantidos no servidor ou realizar ações privilegiadas.

De acordo com a OWASP (2017) os atacantes podem detectar quebras de autenticação através de processos manuais a ferramentas automáticas com listas de palavras-passe e ataques de dicionário, conhecidos como ataques de força bruta. Como consequência de uma quebra de autenticação bem-sucedida os invasores podem obter acesso apenas algumas contas ou apenas um administrador para comprometer todo o sistema, dependendo do domínio do aplicativo, isso pode permitir lavagem de dinheiro, fraude de previdência social e até mesmo roubo de identidade ou revelação de informação altamente sensível.

3. OWASP ZED ATTACK PROXY

O OWASP Zed Attack Proxy, de acordo com a OWASP, é o scanner (varredor) de aplicativos da web mais utilizado no mundo, é um software livre e de código aberto e mantido por uma equipe internacional de voluntários.

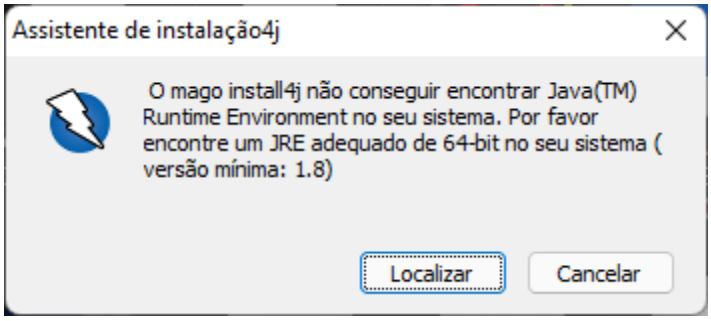
3.1 Requisitos para instalação

O OWASP ZAP é bem versátil, podendo ser executado em diversos sistemas operacionais, podendo ser executado em Windows, Linux e macOS. Roda em arquitetura de 64 e 32 bits.

De acordo com a OWASP, as versões do Windows e do Linux requerem a instalação prévia dl Java 8 ou superior, no macOS uma versão do Java vem incluída.

Caso tente executar o assistente de instalação sem uma versão adequada do Java, aparecerá a seguinte mensagem que encontramos na Figura 3.

Figura 3 – Mensagem de requisição do Java



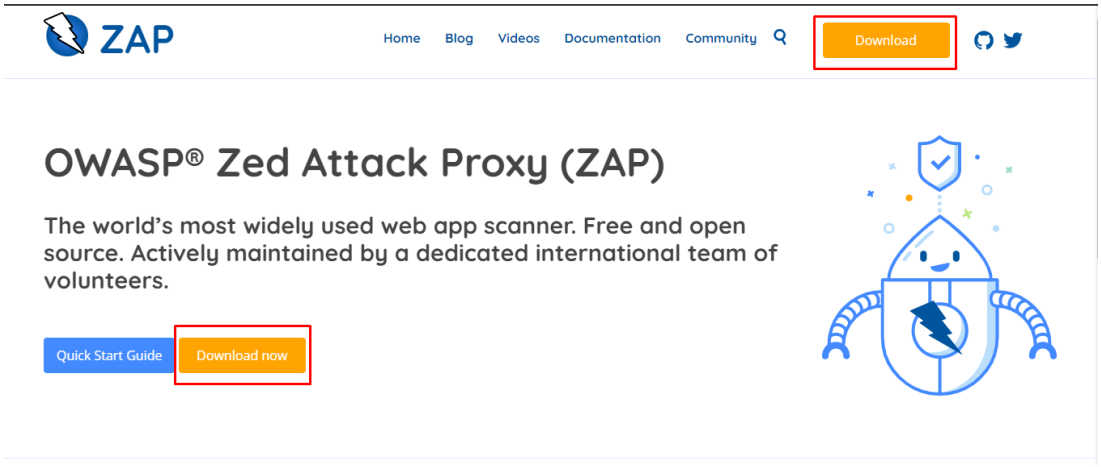
Fonte: Assistente de Instalação OWASP ZAP (2021)

3.1.1 Instalação

A instalação do OWASP Zed Attack Proxy, ou simplesmente, OWASP ZAP, é bem simples, o primeiro passo é acessar a página da ferramenta, como a seguinte url: https://www.zaproxy.org/

Após o acesso à página, clique no botão “Download” ou “Download Now”, como podemos ver na Figura 4:

Figura 4 – Página inicial do OWASP ZAP



Fonte: Site de download OWASP ZAP (2021)

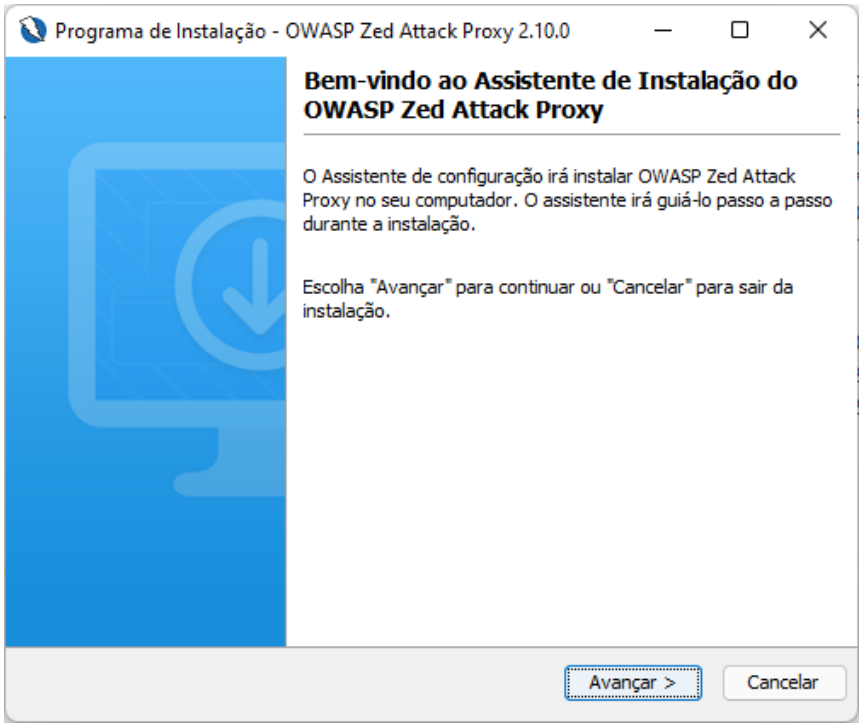
Na tela seguinte, Figura 5, escolha a opção de acordo com seu Sistema Operacional, no caso aqui apresentado, utilizaremos a versão *Windows (64) Installer* e clicar no botão de download:

Figura 5 – Página de Download do OWASP ZAP

Fonte: Página de seleção de download do OWASP ZAP (2021)

Após o download, execute o assistente de configuração, que guiará a instalação de maneira fácil e intuitiva, como podemos observar na Figura 6:

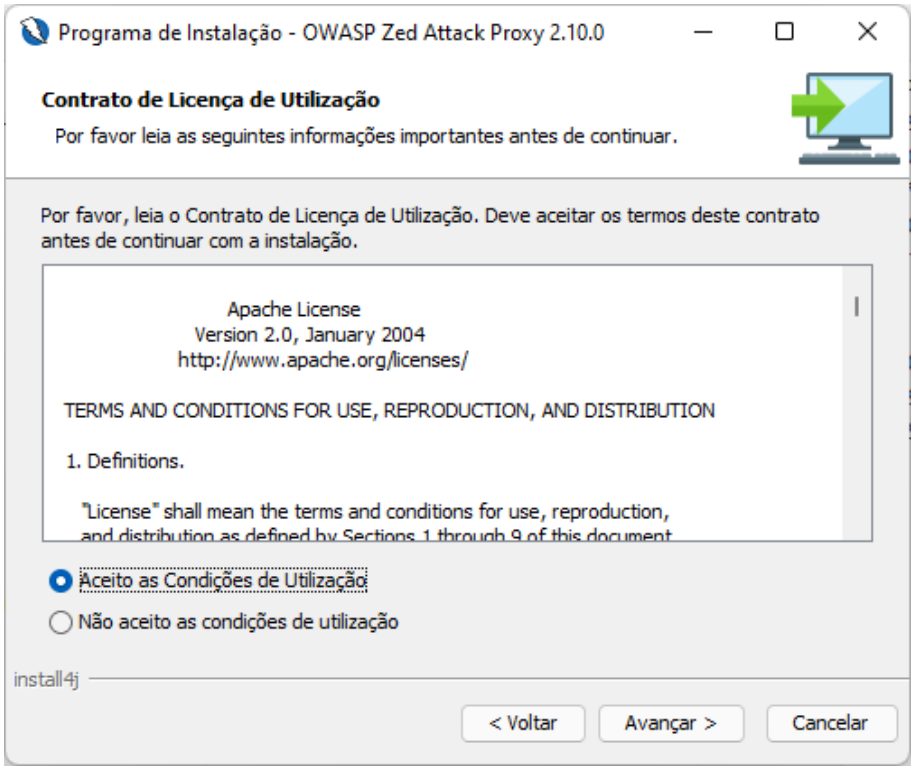
Figura 6 – Tela de boas-vindas do assistente de configuração do OWASP ZAP



Fonte: Instalador do OWASP ZAP (2021)

A próxima tela do assistente de configuração será a do contrato de licença, onde, para prosseguir com a instalação é necessário aceitá-lo, caso recuse, a instalação será interrompida. Nesse caso, foi aceito o contrato e selecionado o botão avançar para continuar com a instalação, como podemos verificar na figura 7:

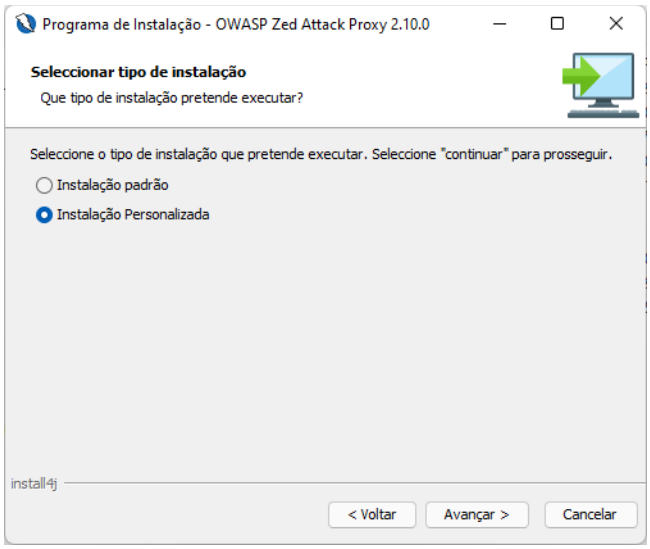
Figura 7 – Tela do contrato de licença das condições de utilização do OWASP ZAP



Fonte: Instalador do OWASP ZAP (2021)

A seguir, podemos optar por dois tipos de instalação, a padrão, onde o OWASP ZAP é instalado em um caminho determinado e a personalizada, onde é possível escolher o diretório de instalação, como podemos observar na Figura 8, foi selecionado a Instalação Personalizada.

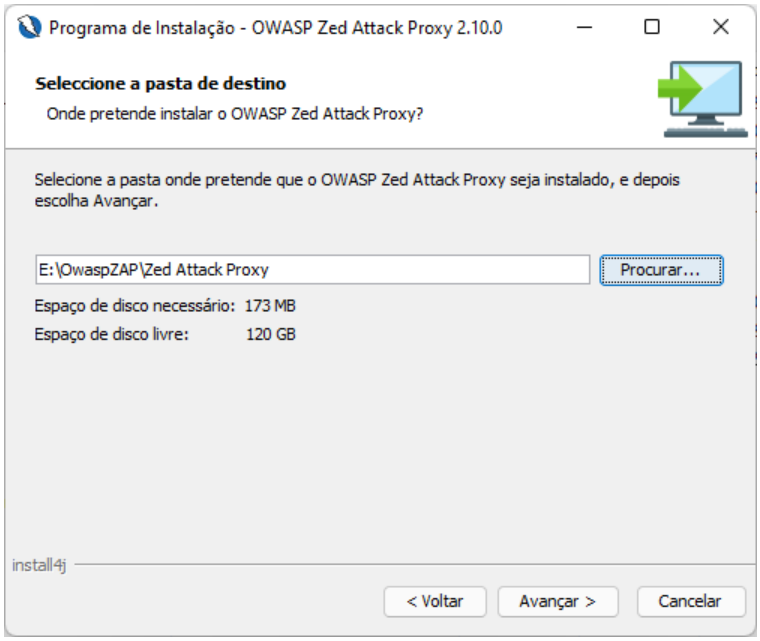
Figura 8 – Tela de seleção do tipo de instalação do OWASP ZAP



Fonte: Instalador do OWASP ZAP (2021)

Na tela de instalação personalizada, foi selecionado uma pasta de destino criada previamente, como podemos verificar na Figura 9.

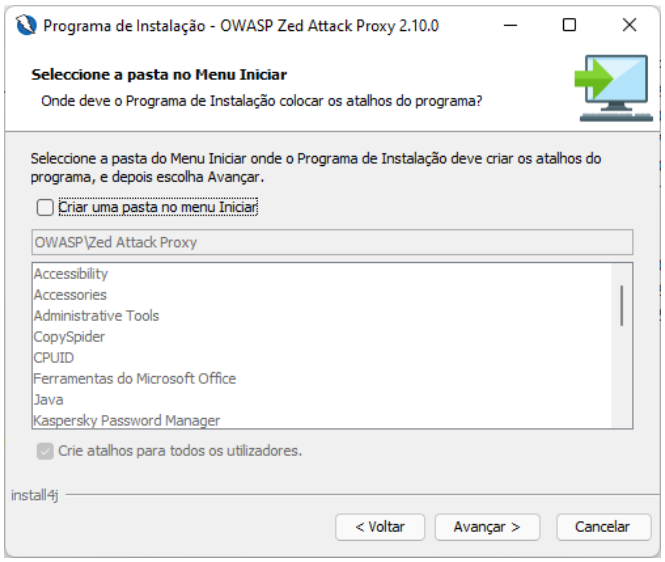
Figura 9 – Tela de seleção do tipo de instalação do OWASP ZAP



Fonte: Instalador do OWASP ZAP (2021)

Na próxima tela, Figura 10, o assistente de lhe dará opções de onde será criado os atalhos do aplicativo, nesse caso, foi deixado desmarcado:

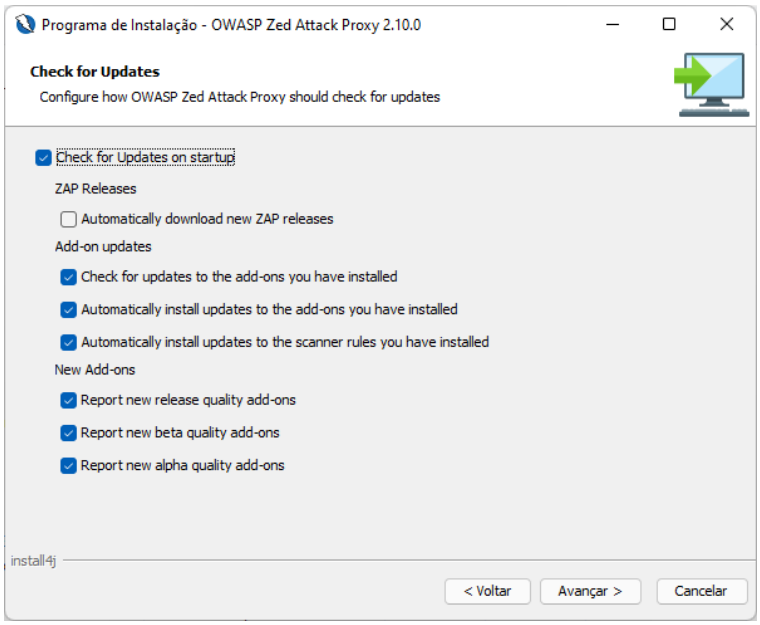
Figura 10 – Tela de seleção da localização dos atalhos do OWASP ZAP



Fonte: Instalador do OWASP ZAP (2021)

Após clicar em “Avançar”, terá opções para criar ícone para o Ambiente de Trabalho, o assistente de configuração lhe dará opções de checar por atualizações automaticamente, clique em “Avançar” para que a aplicação busque por suas atualizações de maneira automática, como podemos ver na Figura 11:

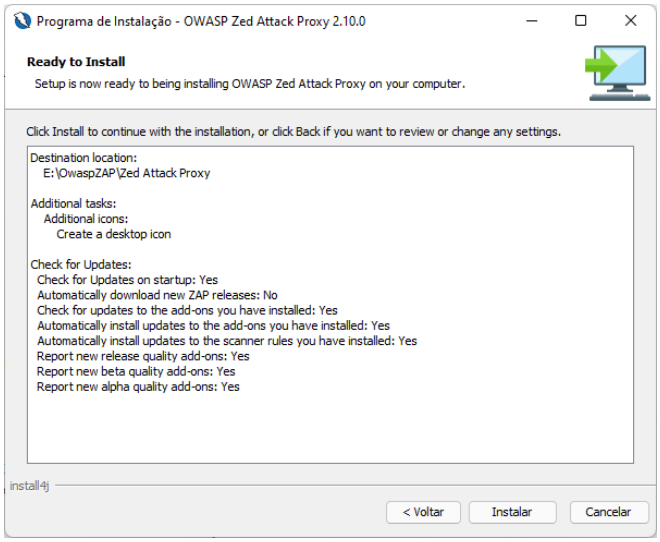
Figura 11 – Tela de configuração das checagens de atualização do OWASP ZAP



Fonte: Instalador do OWASP ZAP (2021)

Na Figura 12 podemos observar o resumo do que foi escolhido para a instalação, agora basta clicar em “Instalar” para prosseguir:

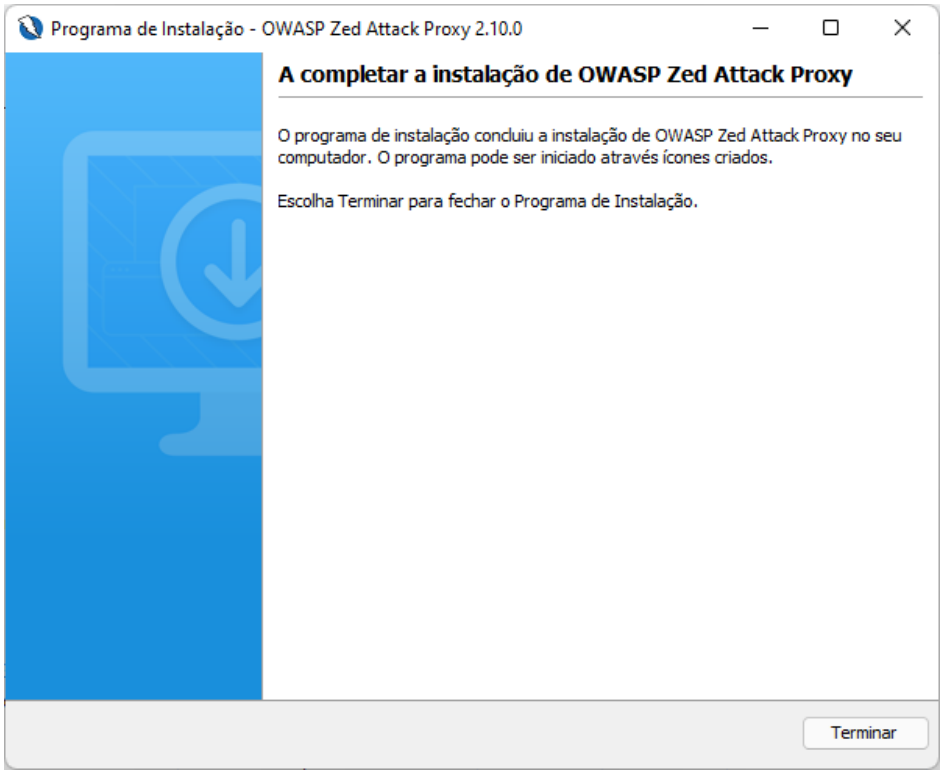
Figura 12 – Resumo da instalação



Fonte: Instalador do OWASP ZAP (2021)

Após finalizar a instalação, clique em terminar, como na Figura 13:

Figura 13 – Tela de finalização da Instalação

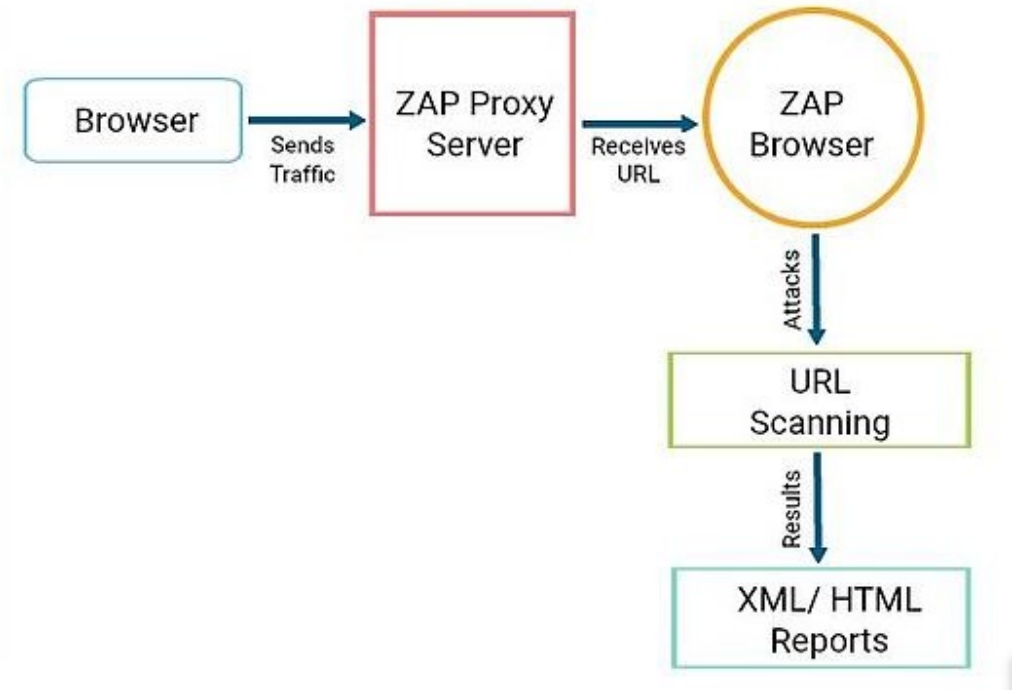


Fonte: Instalador do OWASP ZAP (2021)

3.2 Funcionamento do OWASP ZAP

De acordo com Gokte (2017) a ferramenta OWASP ZAP cria um servidor proxy e faz com que o tráfego do seu site passe por esse servidor que é composto por scanners automáticos que ajudam a interceptar as vulnerabilidades no site. Podemos verificar o fluxo do tráfego gerado pelo OWASP ZAP na Figura 14:

Figura 14 – Funcionamento do OWASP ZAP



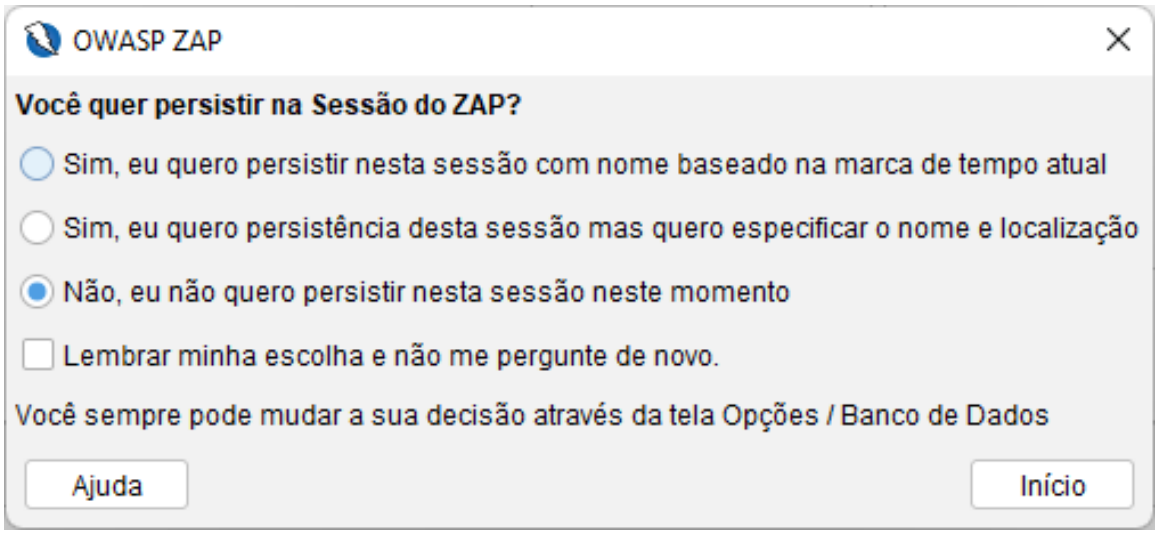
Fonte: Gokte (2017)

De acordo com a OWASP o servidor proxy que o ZAP cria fica entre o navegador do testador e o aplicativo da web para que possa interceptar e inspecionar mensagens enviadas entre o navegador e o aplicativo da web, modificar o conteúdo se necessário e, em seguida, encaminhar esses pacotes para o destino.

3.2.1 Iniciando o OWASP ZAP

Sempre ao iniciar a ferramenta, o usuário será perguntado se deseja persistir na sessão ou iniciar uma nova, como mostrado na Figura 15:

Figura 15 – Persistência de sessão OWASP ZAP



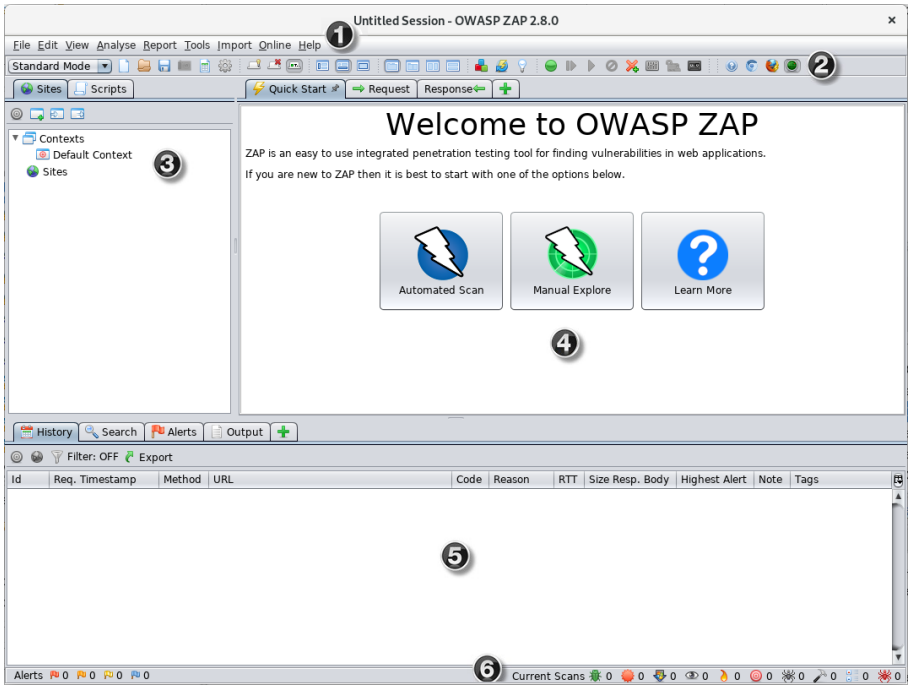
Fonte: OWASP ZAP (2021)

De acordo com a OWASP, por padrão, as sessões do ZAP sempre são salvas em disco em um banco de dados com um nome e localização padrão, se o usuário optar por não persistir na sessão, os arquivos serão deletados. Se o usuário optar por persistir na sessão, as informações da sessão serão salvas no banco de dados local, podendo ser acessadas mais tarde e estará habilitado para salvar os arquivos em outro local e com outro nome.

3.2.2 Apresentação da ferramenta

A interface gráfica do OWASP ZAP é simples e amigável, iremos definir os principais elementos mostrados na Figura 16:

Figura 16 – Tela inicial do OWASP ZAP



Fonte: OWASP (2021)

De acordo com a OWASP, podemos definir as áreas como:

1. Barra de menu: fornece acesso a diversas ferramentas automatizadas e manuais;
2. Barra de ferramentas: fornece acesso rápido e fácil aos botões dos recursos mais utilizados;
3. Janela em árvore: exibe a árvore de sites e a árvore de script;
4. Janela de espaço de trabalho: exibe as solicitações, respostas e os scripts e permite que os edite;
5. Janela de Informações: exibe os detalhes das ferramentas automatizadas e manuais;
6. Rodapé: exibe o status das principais ferramentas automatizadas e um resumo dos alertas.

3.3 Demonstração do OWASP ZAP

Para efeito de demonstração e uso da ferramenta, foi utilizado o OWASP *Broken Web Applications Project,* o objetivo é demonstrar que a ferramenta OWASP ZAP é realmente efetiva e cumpre sua proposta.

O OWASP Broken Web Applications Project, de acordo com a OWASP, “é uma coleção de aplicativos da web vulneráveis que são distribuídos em uma máquina virtual”.

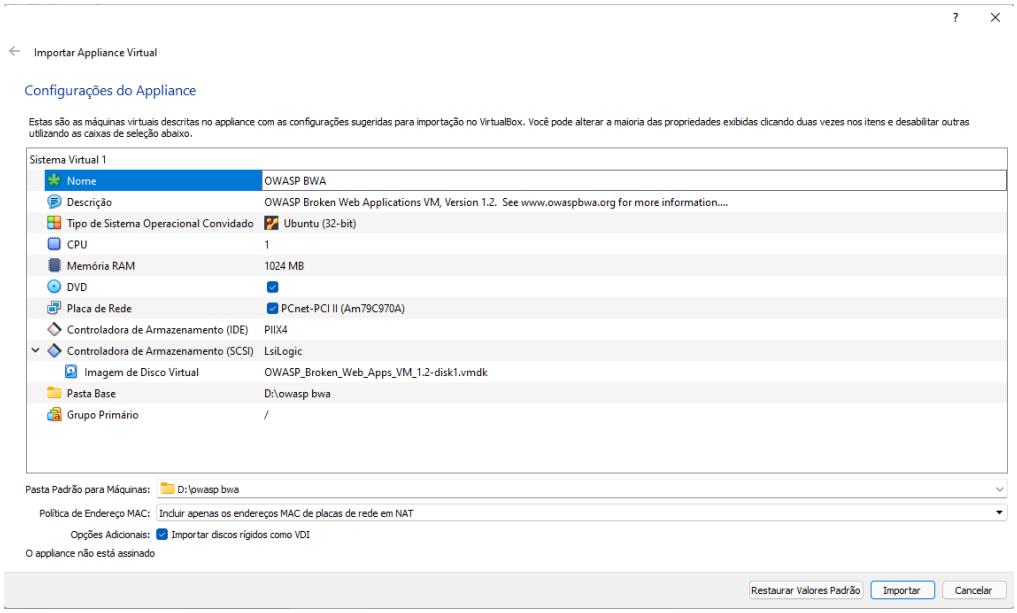
Na máquina virtual produzida para o OWASP Broken Web Applications Project, existe uma variedade de aplicações web com diversas vulnerabilidades conhecidas, é indicada para aqueles que querem aprender mais sobre teste de segurança em aplicações web.

3.3.1 Preparação do ambiente de testes

Para que seja possível gerenciar e executar a máquina virtual do OWASP Broken Web Applications Project é necessário utilizar algum software de virtualização, nesse caso foi utilizado o VirtualBox, um software muito popular e de código livre. A imagem de disco da máquina virtual pode ser encontrada na seguinte url: https://sourceforge.net/projects/owaspbwa/files/latest/download.

Feito o download do arquivo basta e com o VirtualBox instalado, basta dar duplo clique no arquivo com a extensão “.ova” que será direcionado para as configurações da importação, como podemos verificar na Figura 17, apenas foi alterado os campos nome l e pasta padrão para a máquina, feito isso, basta clicar em no botão Importar:

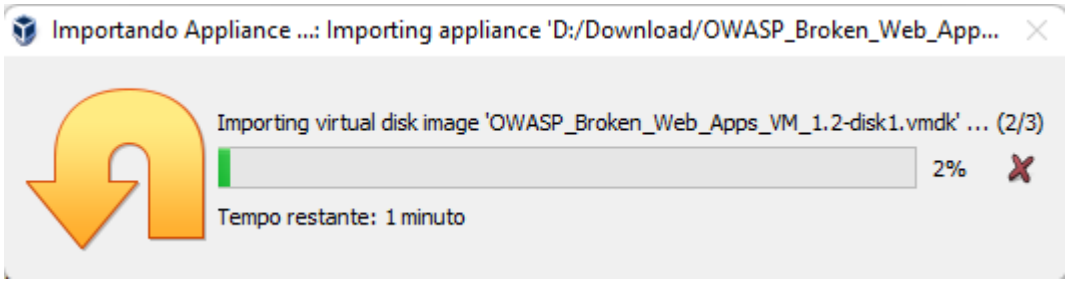
Figura 17 – Configuração da importação da máquina virtual



Fonte: Virtual Box (2021)

Após, o VirtualBox trará o tempo estimado para completar a importação como pode ser observado na Figura 18:

Figura 18 – Tempo estimado para importação

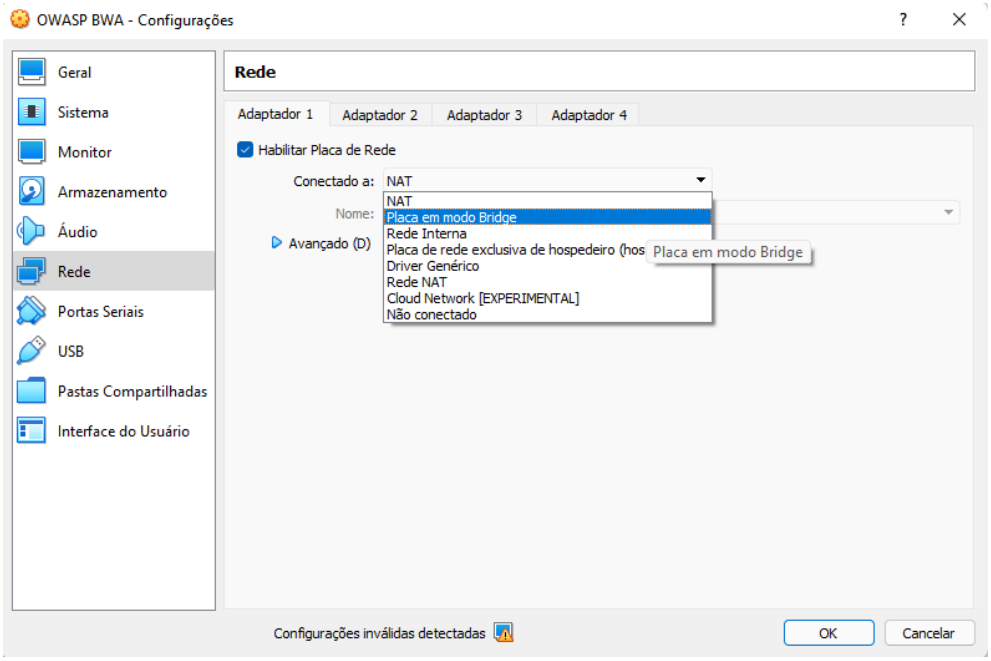


Fonte: Virtual Box (2021)

Após finalizar a importação, foi configurado a placa de rede a máquina virtual em modo *Bridge,* para que ela seja acessada através da rede local e a realizarmos a demonstração do OWASP ZAP.

Para colocar a placa de rede virtual em modo *bridge* acessar as configurações da máquina, logo após, selecionar rede e no campo “Conectado a:” selecionar “Placa em modo Bridge” e clicar em “OK” para aplicar as configurações, como podemos observar na Figura 19:

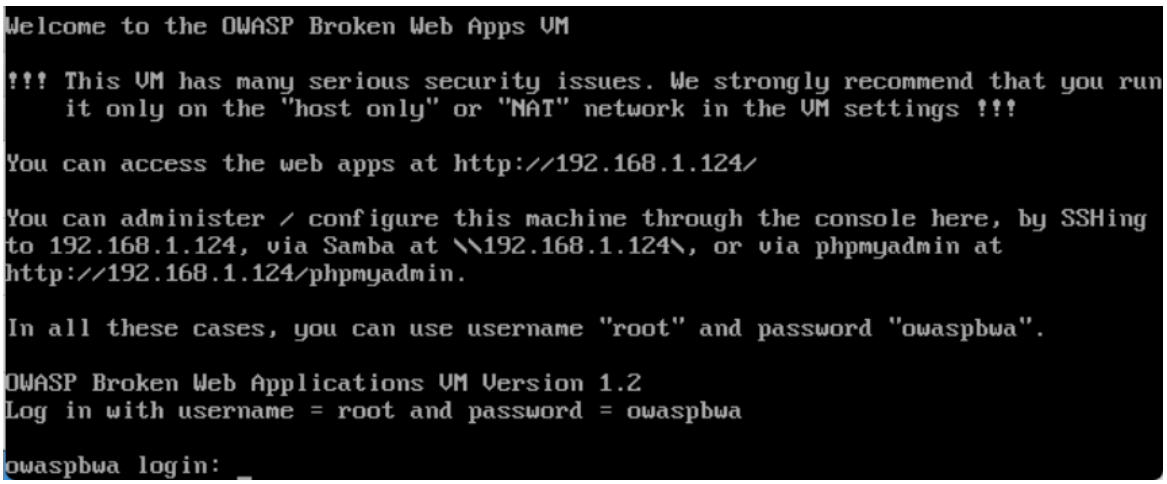
Figura 19 – Configuração da Placa de rede em modo Bridge



Fonte: Virtual Box (2021)

Feito isso, a máquina virtual está pronta para ser executada. Com o ambiente virtualizado em execução, precisamos do IP que pode ser encontrado assim que a máquina é iniciada, como podemos verificar na Figura 20, o IP para acessar os aplicativos é 192.168.1.124:

Figura 20 – Inicialização OWASP *Broken Web Applications Project*



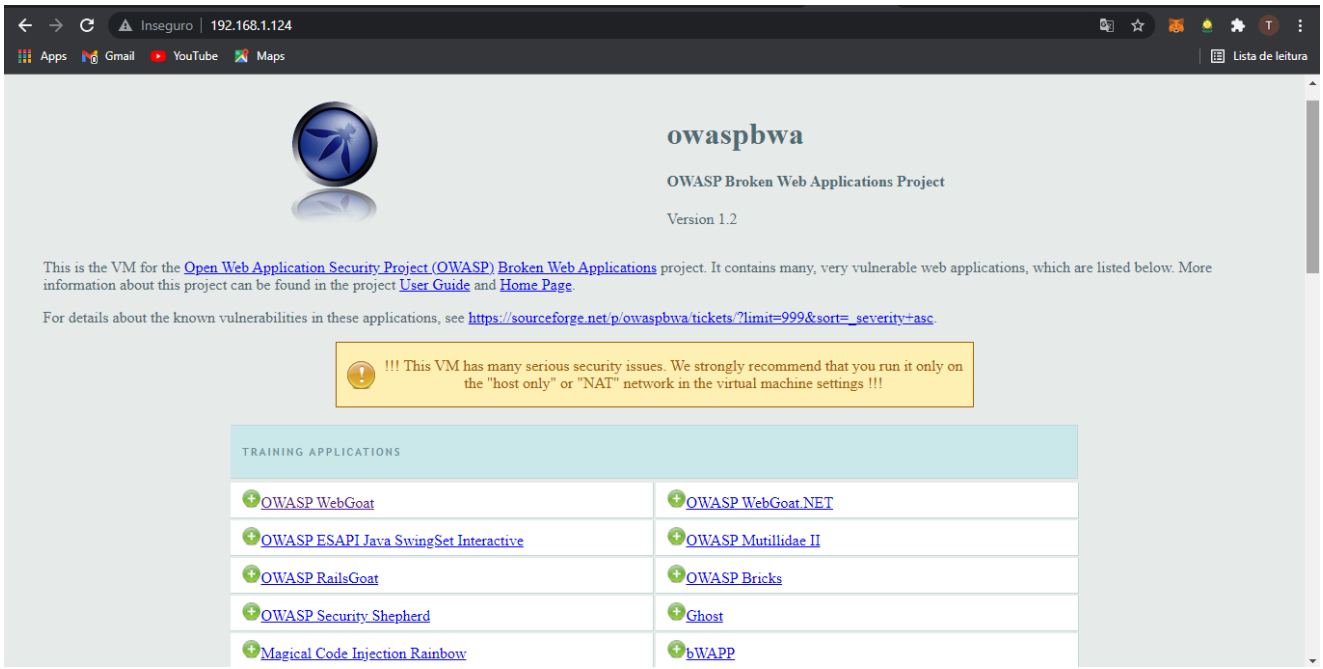
Fonte: OWASP Broken Web Applications Project (2021)

Observação: Como podemos observar na figura 20, a OWASP recomenda que a máquina virtual seja executada no modo “host Only” ou “NAT”, já que a máquina

virtual possuí muitos problemas de segurança, para que ela esteja isolada da nossa rede local, como configuramos em modo *bridge*. Mas, como se trata apenas de uma simples demonstração, foi mantido em modo *bridge* durante os testes.

Agora que obtemos o IP, com um navegador, teremos acessos as aplicações web vulneráveis que o OWASP Broken Web Applications Project traz, como podemos observar na Figura 21:

Figura 21 – Tela inicial do OWASP Broken Web Applications Project



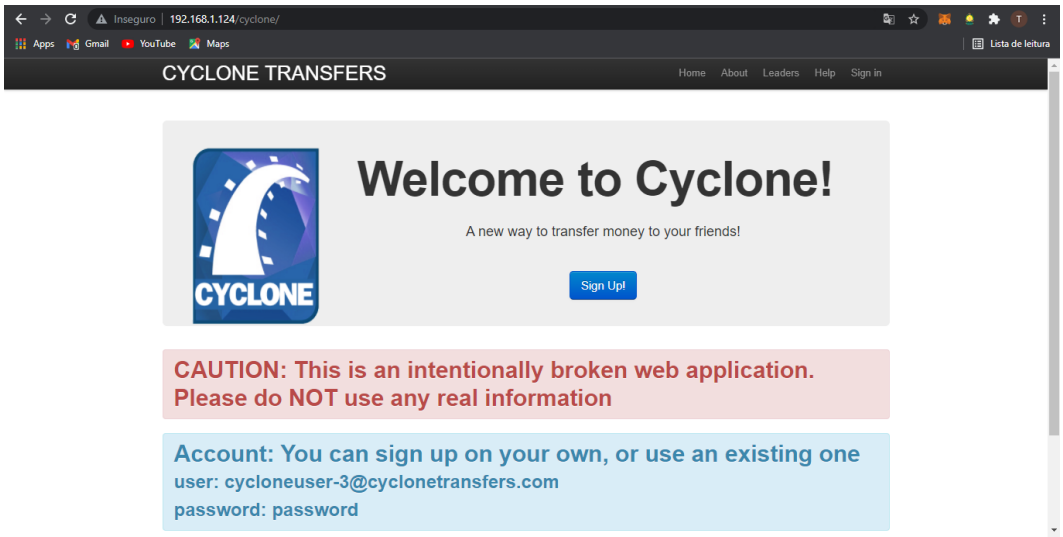
Fonte: OWASP *Broken Web Applications Project* (2021)

Nesse ponto é possível selecionar a vulnerabilidade a ser testada, existem diversas categorias e formas de explorar.

3.3.2 Varredura de vulnerabilidades

Testaremos a aplicação Cyclone, que simula a aplicação de transferência de dinheiro. Como podemos ver na Figura 22, na página inicial da aplicação é exibido uma mensagem para não utilizar dados reais pois se trata de uma aplicação vulnerável:

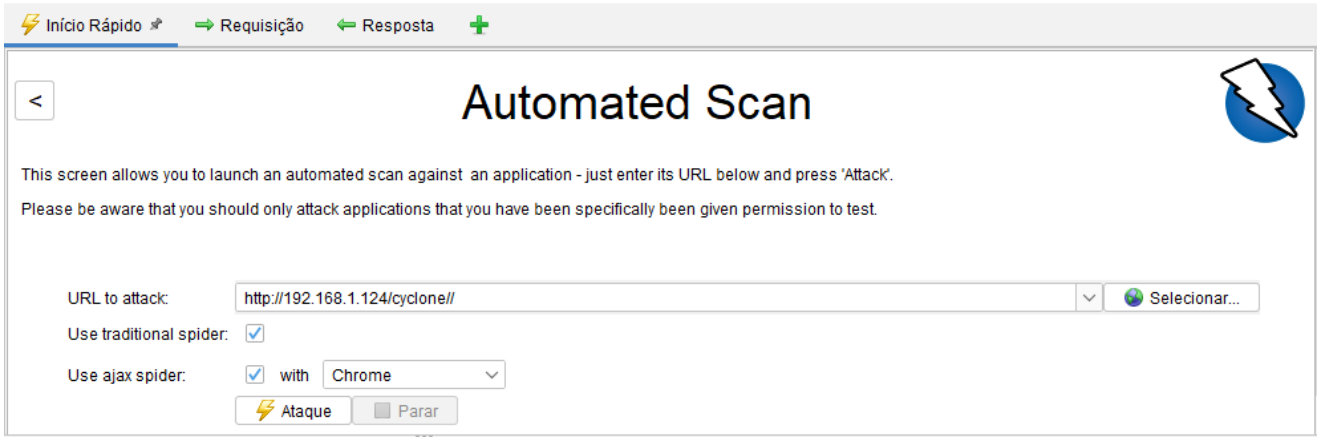
Figura 22 – Aplicação Cyclone do OWASP Broken Web Applications Project



Fonte: OWASP *Broken Web Applications Project* (2021)

Depois de iniciar o OWASP ZAP e selecionar o modo automático clicando em *"Automated Scan"* é necessário inserir a URL da aplicação que será testada no campo *"URL to attack"* e depois clicar no botão ataque, como podemos ver na figura 23:

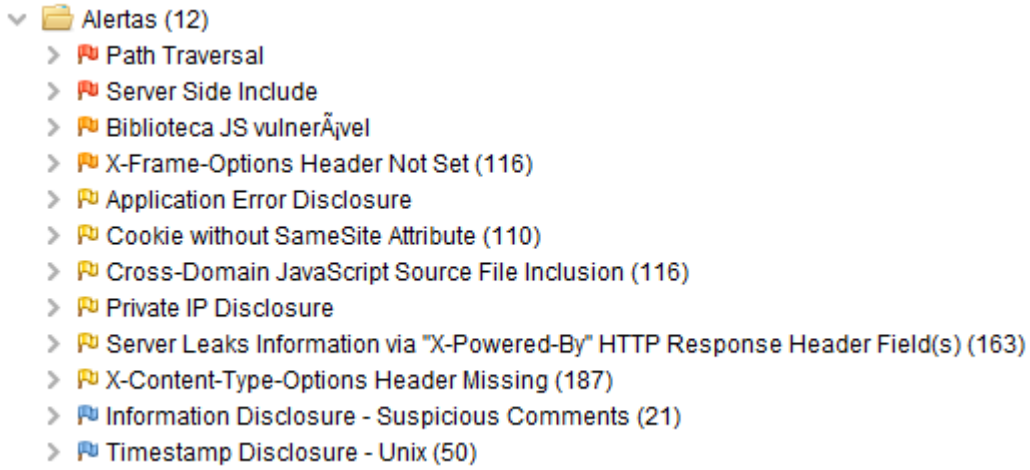
Figura 23 – Tela inicial do OWASP Broken Web Applications Project



Fonte: OWASP ZAP (2021)

Foram retornados 12 alertas, sendo dois de alto risco, como podemos verificar na Figura 24:

Figura 24 – Vulnerabilidades encontradas no Cyclone



Fonte: OWASP ZAP (2021)

Selecionando algum alerta, o OWASP ZAP traz de forma detalhada a vulnerabilidade, traz URL onde foi encontrada a vulnerabilidade, o nível do risco, o nível de confiança da vulnerabilidade, parâmetro, ataques, referências e outras diversas informações, como podemos verificar na Figura 25:

Figura 25 – Detalhes da vulnerabilidade

Fonte: OWASP ZAP (2021)

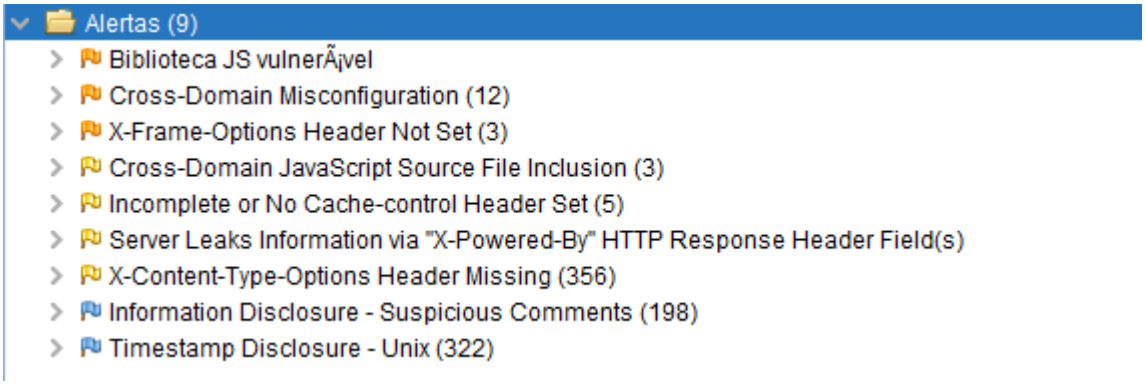
Podemos constatar que o OWASP ZAP traz diversas, dando informações valiosas, tanto para os profissionais de segurança da informação quanto para os atacantes, além de trazer referências que mostram os impactos que uma vulnerabilidade pode causar e a solução para mitigá-la.

4. Resultados obtidos

O produto testado ainda está em fase de desenvolvimento, por motivos de confidencialidade todas as funções ou conteúdos que possam identificá-lo, serão ocultados. No desenvolvimento do trabalho, apenas serão demonstradas as vulnerabilidades e as referências utilizadas para mitigá-las.

Nos testes realizados com o OWASP ZAP foram encontrados 9 alertas, podemos observar os alertas na figura 26:

Figura 26 – Alertas encontrados no produto



Fonte: OWASP ZAP (2021)

Como podemos observar, não foi encontrada nenhuma vulnerabilidade de alto risco, mas, os alertas apontam 3 vulnerabilidades de médio risco, 4 de baixo risco e 2 avisos, cada vulnerabilidade de nível médio encontrada será mais detalhada nos próximos tópicos.

4.1 Biblioteca JS vulnerável

Utilizando o OWASP ZAP, foi constatado que produto utiliza uma versão desatualizada do Javascript, de acordo com Higa (2017) “[..] a biblioteca mais comum é o jQuery, com 84,5% de participação nas páginas. Desses sites, 36,7% utilizam uma versão vulnerável.”

A versão da biblioteca jQuerry utilizada na aplicação é a 3.3.1, de acordo com a Snyk, essa versão, pode apresentar dois problemas de segurança, o *Cross-site Scripting (XSS)* e a *Prototype Pollution,* a solução recomenda para mitigar essas vulnerabilidades é fazer o *upgrade* da biblioteca para a versão 3.4.0.

4.1.1 Cross-site Scripting (XSS)

A vulnerabilidade *Cross-site Scripting,* também conhecido apenas por XSS, de acordo com Barbosa (2018):

> "É uma vulnerabilidade presente em aplicações web que permite que o cibercriminoso insira códigos Javascript para obter certos tipos de vantagem sobre as vítimas."

No OWASP TOP 10 de 2017, o XSS foi considerado a segunda maior vulnerabilidade em aplicações web, sendo destacado que dois terços das aplicações são vulneráveis a esse tipo de ataque.

De acordo com a OWASP (2017) existem 3 tipos de XSS cujo foco geralmente são o navegador dos utilizadores, sendo estes:

- ***Reflected XSS*:** A aplicação inclui dados de entrada do utilizador como parte do HTML da resposta sem que estes tenham sido validados e/ou caracteres especiais tratados. Um ataque bem-sucedido pode permitir a execução de código HTML e JavaScript no navegador da vítima. Normalmente a vítima segue um endereço malicioso para uma página controlada pelo atacante.
- ***DOM Based XSS:*** Essa categoria de XSS executa comandos localmente no navegador da vítima, de acordo com Barbosa (2018):

  > "Esse tipo de ataque é menos comum pois depende que a página alvo tenha componentes específicos que permitam que a ativação dos códigos aconteça em tempo de execução"

- ***Stored XSS:*** A aplicação armazena dados de entrada do usuário de forma não tratada, os quais serão mais tarde acessados por outro utilizador ou administrador. Este tipo de XSS é considerado de risco alto ou crítico. De acordo com Barbosa (2018):

  > "[..] essa forma de ataque exige que os cibercriminosos possuam uma forma de escrever dados diretamente na página, como por exemplo campos de comentários, testemunhos e livro de visitas."

4.1.2 Prototype Pollution

Em JavaScript os Protótipos são os meios pelos quais os objetos herdam recursos uns dos outros. De acordo com o CVE-2019-11358, as versões do jQuery anteriores a 3.4.0 são vulneráveis à poluição de protótipo.

Nas palavras de Pardeiro (2019):

> "Prototype Pollution é uma falha de segurança que permite que invasores modifiquem o protótipo de objeto JavaScript de um aplicativo da Web. Isso pode causar a falha do aplicativo ou alterar seu comportamento se ele não receber os valores esperados."

De acordo com Ramić (2021) um atacante é capaz de manipular atributos ou poluir um protótipo de objeto injetando outros valores. As propriedades serão herdadas por todos os objetos por meio da cadeia de protótipo, e isso pode resultar em negação de serviço (se acionado exceções de JavaScript), execução remota de código (se adulterado o código-fonte do aplicativo) ou XSS.

4.3 *Cross-Domain Misconfiguration*

De acordo com we45, *Cross-Origin Resource Sharing*, mais conhecido como CORS, é um mecanismo que permite que os navegadores da web executem solicitações entre domínios usando a API XMLHttpRequest de maneira controlada. Essas solicitações de origem cruzada têm um cabeçalho de origem, que identifica o domínio que inicia a solicitação. Ele define o protocolo a ser usado entre um navegador da web e um servidor para determinar se uma solicitação de origem cruzada é permitida. Ou seja, o CORS permite compartilhar recursos entre diferentes origens, permitindo que um site utilize estes recursos de outro site mesmo estando em domínios diferentes.

Todas as vulnerabilidades do CORS são causadas pela *misconfiguration* (configuração incorreta) e pode comprometer todo o aplicativo web e até mesmo que os dados de acesso do usuário sejam enviados à fontes maliciosas. Para mitigar essas vulnerabilidades, é necessário especificar no cabeçalho HTML que o compartilhamento do recurso deve ser permitido apenas entre fontes confiáveis a

serem especificadas, não permitindo que qualquer domínio possa interagir com os recursos.

4.3 X-Frame-Options Header Not Set

De acordo com a MDN o cabeçalho HTTP X-Frame-Options é usado para indicar ao navegador se ele deve ou não renderizar um frame. Se um cabeçalho X-Frame-Options for aplicado de maneira inconsistente ou estiver ausente a aplicação web pode estar vulnerável a ataques clickjacking.

Ataques do tipo Clickjacking é uma técnica maliciosa que busca enganar o usuário, fazendo com ele clique em algo diferente do que era para ser clicado, nas palavras de Rydstedt é quando um invasor usa várias camadas transparentes ou opacas para enganar um usuário e fazê-lo clicar em um botão ou link em outra página quando pretendia clicar na página de nível superior. Assim, o invasor está “sequestrando” cliques destinados à sua página e os direcionando para outra página, provavelmente pertencente a outro aplicativo, domínio ou ambos. Usando uma técnica semelhante, os pressionamentos de tecla também podem ser sequestrados. Com uma combinação cuidadosamente elaborada de folhas de estilo, iframes e caixas de texto, um usuário pode ser levado a acreditar que está digitando a senha em seu e-mail ou conta bancária, mas em vez disso está digitando em um quadro invisível controlado pelo invasor.

Ainda de acordo com Rydsted, para mitigar ataques clickjacking é necessário configurar o servidor web para incluir o cabeçalho X-Frame-Options, indicando ao navegador se deve ou não ter permissão para renderizar a página dentro de um frame. Também pode ser implementada uma Política de Segurança de Conteúdo (mais conhecida como CSP) instruindo ao navegador a não permitir frames de outros domínios.

5 CONSIDERAÇÕES FINAIS

Com o aumento do uso da tecnologia e com o grande fluxo de informação online a Segurança da Informação tem se tornado prioridade, por este motivo, esse trabalho buscou demonstrar a importância de se testar a segurança de aplicações web, principalmente na fase de desenvolvimento do *software,* antes de colocá-lo em ambiente de produção.

O OWASP *Zed Attack Proxy* se mostrou versátil e extremamente eficiente para encontrar as vulnerabilidades de aplicativos web, podendo ser implementado e utilizado por qualquer empresa, independentemente de seu porte. É uma ferramenta que pode e deve ser usada para aperfeiçoar a segurança em aplicações web, trazendo informações relevantes para implementar as correções das falhas de segurança, evitando futuros ataques e desta forma, aumentar a proteção dos dados.

As vulnerabilidades encontradas foram reportadas aos gestores da empresa que irão analisar quais medidas deverão ser tomadas para corrigir as falhas de segurança antes que a aplicação seja colocada em um ambiente de produção.

REFERÊNCIAS

ARQUISERV. **Como funciona a arquitetura cliente servidor.** Disponível em: https://arqserv.wordpress.com/2012/03/17/como-funciona-a-arquitetura-cliente-servidor/. 18 de março de 2021.

AQUINO, M. C**. Um resgate histórico do hipertexto: O desvio da escrita hipertextual provocado pelo advento da *Web* e o retorno aos preceitos iniciais através de novos suportes.** Ed. Unirevista: Porto Alegre, v. 1, n. 3, p.1-14, jul. 2006.

BARBOSA, Daniel Cunha, **Cross-Site Scripting (XSS): entenda o que é e saiba como estar protegido.** Disponível em: https://www.welivesecurity.com/br/2018/12/27/cross-site-scripting-xss-entenda-o-que-e-e-saiba-como-estar-protegido/. Acesso em 12 de Outubro de 2021.

CERT.BR, **Incidentes Reportados ao CERT.br - Janeiro a Junho de 2020,** Disponível em: https://www.cert.br/stats/incidentes/2020-jan-jun/total.html. Acesso em 27 de Março de 2021.

DEVMEDIA, **SQL *injection.*** Disponível em: https://www.devmedia.com.br/sql-injection/6102. Acesso em 16 de Maio de 2021.

FEITOSA, Eduardo, **O que é o HTML 5.** DEVMEDIA. 2012, Disponível em: https://www.devmedia.com.br/o-que-e-o-html5/25820. Acesso em 21 de Março de 2021.

GOKTE, Surabhi, *An intro to OWASP Zed Attack Proxy.* 2017, Disponível em: https://www.srijan.net/resources/blog/intro-owasp-zed-attack-proxy. Acesso em: 14 de Novembro de 2021.

HIGA, Paulo, **A web está insegura por causa de JavaScript velho**. 2017 Disponível em: https://tecnoblog.net/210656/web-insegura-javascript-velho/. Acesso em 05 de Outubro de 2021.

KEMP, Simon. WE ARE SOCIAL E HOOTSUITE. *We Are Social e HootSuite*. **Digital 2021: The last Insights into the "state of digital".** Disponível em: https://wearesocial.com/blog/2021/01/digital-2021-the-latest-insights-into-the-state-of-digital. Acesso em 24 de Março de 2021.

KUROSE, JIM F.; ROSS, K. W. **Redes de Computadores e a Internet - Uma Abordagem Top-Down.** 6 ed. São Paulo: Pearson, 2013. 62 p.

LIVING IN BELGIUM, **Diferença entre ataques ativos e passivos.** Disponível em: https://pt.living-in-belgium.com/difference-between-active-and-passive-attacks-375 Acesso em 16 de Maio de 2021.

LOPES, Petter. **Ataques de Injeção.** Disponível em: https://periciacomputacional.com/ataques-de-injecao-injection-attacks/. Acesso em 16 de Maio de 2021.

KEARY, Eoin; MEUCCI, Matteo; MULLER, Andrew. **OWASP Testing Guide v4**. Maryland, Eua: Owasp, 2014. 5 p.

NORMA ABNT NBR BRASILEIRA, **ISO/IEC 27002.** Disponível em: http://static.cegoc.pt/pdf/894.pdf. Acesso em 16 de Maio de 2021.

PRESSMAN, Roger S. **Engenharia de Software: Uma abordagem profissional.** [recurso eletrônico]/ Roger S. Pressman, Bruce R. Maxim; [tradução: João Eduardo Nóbrega Tortello] revisão técnica: Reginaldo Arakaki, Renato Manzan de Andrade – 8 ed. Porto Alegre: AMGH, 2016.

OWASP, *Open Web Application Security Project.* ***OWASP Broken Web Applications****.* Disponível em: https://owasp.org/www-project-broken-web-applications/migrated_content#:~:text=Applications%20Project%20provides%3A-,OWASP%20Broken%20Web%20Applications%20Project%20provides%3A,to%20safely%20practice%20penetration%20testing. Acesso em: 08 de Outubro de 2021.

OWASP, *Open Web Application Security Project.* ***Vulnerabilities.*** Disponível em: https://owasp.org/www-community/vulnerabilities/. Acesso em 16 de Maio de 2021.

OWASP, *Open Web Application Security Project.* **Top 10 Web Application Security Risks**. Disponível em: https://owasp.org/www-project-top-ten/. Acesso em 16 de Maio de 2021.

OWASP*.* ***OWASP TOP 10 – 2017: The Ten Most Critical Web Application Security Risks*****.** 2017. 8 – 10 p.

OWASP, *Open Web Application Security Project.* ***OWASP Zed Attack Proxy.*** Disponível em: https://www.zaproxy.org/. Acesso em 13 de Setembro de 2021.

OWASP, *Open Web Application Security Project.* ***Getting Started Guide****.* Disponível em: https://www.zaproxy.org/getting-started/. Acesso em: 14 de Setembro de 2021.

PAULI, J. Introdução ao Web Hacking. 1. ed. [S.l.]: novatec, 2015. 26 p.

RAMIĆ, Omer. ***Everything you need to know about Prototype Pollution***, 2021. Disponível em: https://www.neuralegion.com/blog/prototype-pollution/. Acesso em: 02 de novembro de 2021

Rydstedt, Gustav. ***Clickjacking.*** Disponível em: https://owasp.org/www-community/attacks/Clickjacking. Acesso em 02 de Novembro de 2021.

SÊMOLA. Marcos. **Gestão da Segurança da Informação.** Ed. Elsevier, 2003. 17ª triagem. São Paulo, 2003. 45 p.

SÊMOLA. Marcos. **Gestão da Segurança da Informação:** uma visão executiva**.** Ed. Elsevier, 2 ed. São Paulo, 2014. 18 p.

SNYK, **jquery@3.3.1.** Disponível em: https://snyk.io/test/npm/jquery/3.3.1. Acesso em: 05 de Outubro de 2021.

SINGH, Harpreet; SHARMA, Himanshu. ***Hands-On Web Penetration Testing with Metasploit: The subtle art of using Metasploit*** 5.0 for web application exploitation. Packt Publishing Ltd, v. 3, f. 272, 2020. 544 p.

SHIREY, R. RFC 2828: Internet Security Glossary. 2000. Disponível em: https://datatracker.ietf.org/doc/html/rfc2828. Acesso em: 13 de maio de 2021

STUTTARD, D., PINTO, M. ***The web application hacker's handbook: discovering and exploiting security flaws***. Ed. John Wiley & Sons, 2 ed. Nova Iorque, EUA. 2011. 23 p.

[The MITRE Corporation] **CVE-2019-11358**. Disponível em: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-11358. Acesso em: 03 de Outubro de 2021.

VIEITES. Álvaro Gómez. ***Tipos de ataques e intrusos en las redes informáticas***. Disponível em: https://www.edisa.com/wp-content/uploads/2019/08/ponencia_-_tipos_de_ataques_y_de_intrusos_en_las_redes_informaticas.pdf. Acesso em: 25 de Maio de 2021.

WE45, ***3 Ways You Can Exploit CORS Misconfigurations.*** Disponível em: https://we45.com/blog/3-ways-to-exploit-cors-misconfiguration. Acesso em: 20 de Outubro de 2021.

GLOSSÁRIO

Cabeçalho HTML: parte de marcação superior da página HTML.

Conteúdo CSP: mecanismo de política de segurança de conteúdo que fornece capacidade de criar ou remover regras sobre o conteúdo exibido.

CVE: *Common Vulnerabilities and Exposures* (Vulnerabilidades e exposições comuns) é um banco de dados que armazena vulnerabilidades e falhas registradas.

Frame: elemento do HTML responsável por exibir outra página HTML dentro de uma área.

Protótipo de objeto JavaScript: um objeto java é um membro dentro de uma classe na linguagem de programação java, ele tem seu comportamento baseado na programação interna.

Script: série de orientações que visam roteirizar ou automatizar um processo.

SNYK: plataforma de desenvolvimento para segurança de código aberto.

Upgrade: atualizar ou renovar algum objeto ou situação.