UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CIÊNCIAS DA COMPUTAÇÃO

Samuel Cardoso

# Ambiente para Aprendizado e Prática de Técnicas de Segurança Ofensiva em Aplicações Web Vulneráveis

Florianópolis
2023

Samuel Cardoso

# Ambiente para Aprendizado e Prática de Técnicas de Segurança Ofensiva em Aplicações Web Vulneráveis

Trabalho de Conclusão de Curso submetido ao Curso de Graduação em Ciências da Computação do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito para obtenção do título de Bacharel em Ciência da Computação.
Orientadora: Profª. Carla Merkle Westphall, Dra.

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Cardoso, Samuel
Ambiente para aprendizado e prática de técnicas de segurança ofensiva em aplicações web vulneráveis / Samuel Cardoso ; orientadora, Carla Merkle Westphall, 2023.
117 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Ciências da Computação, Florianópolis, 2023.

Inclui referências.

1. Ciências da Computação. 2. Segurança Ofensiva. 3. Aplicações Web. 4. Vulnerabilidades. I. Westphall, Carla Merkle. II. Universidade Federal de Santa Catarina. Graduação em Ciências da Computação. III. Título.

Samuel Cardoso

**Ambiente para Aprendizado e Prática de Técnicas de Segurança Ofensiva em Aplicações Web Vulneráveis**

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharel em Ciência da Computação e aprovado em sua forma final pelo curso de Graduação em Ciências da Computação.

Florianópolis, 2023.

---

Prof[a]. Lúcia Helena Martins Pacheco, Dra.
Coordenadora do Curso

**Banca Examinadora:**

---

Prof[a]. Carla Merkle Westphall, Dra.
Orientadora
Universidade Federal de Santa Catarina

---

Fabricio Bortoluzzi, Me.
Avaliador
Noroff University College

---

Rômulo Augusto Oliveira Cruz Bittencourt de Almeida, Me.
Avaliador
Universidade Federal de Santa Catarina

## AGRADECIMENTOS

A Deus, por sua eterna bondade.
A minha família, por todo suporte dado e pela paciência nestes anos.
A minha orientadora Carla Merkle Westphall por todo auxílio.
A minha dívida na Biblioteca Universitária que não me permitiu trancar o curso na pandemia.
A dívida que fiz no nome do meu pai que não me permitiu ir de arrasta para cima.
A Osamu Nishi pela criação da obra que mais me apoiou durante minha depressão:
Mairimashita! Iruma-kun.
Aos meus amigos que tanto me auxiliaram na graduação, em especial agradeço a: José, João, Bernardo, Gabriel, André, Hans, Nicolas, Leonardo, Arthur e Thiago.

## RESUMO

Este trabalho discorre sobre o estado da arte da segurança da informação com base na OWASP Top Ten:2021, elenca dez vulnerabilidades que ainda impactam as aplicações web no ano de 2023 e desenvolve um ambiente para aprendizado e prática de técnicas de segurança ofensiva por meio de uma aplicação web com estas vulnerabilidades. Este ambiente é desenvolvido em Docker para que a instalação do ambiente seja simplificada, com isso, o tempo gasto com o ambiente fica totalmente focado na exploração das vulnerabilidade e não na instalação do sistema.



**Palavras-chave:** Segurança Ofensiva. Aplicações Web. Vulnerabilidades.

## ABSTRACT

This work discusses the state of the art of information security based on the OWASP Top Ten:2021, lists ten vulnerabilities that still have an impact on web applications in the year 2023 and develops an environment for learning and practive offensive security techniques through a web applications with these vulnerabilities. This environment is developed in Docker so that the installation of the environment is simplified, with this, the time spent with the environment is totally focused on exploiting the vulnerabilities and not on installing the system.



**Keywords:** Offensive Security. Web Applications. Vulnerabilities.

## LISTA DE FIGURAS

**SUMÁRIO**

# 1 INTRODUÇÃO

Nos últimos cem anos, o uso da Internet experimenta um crescimento exponencial devido à sua democratização, afetando a vida cotidiana e corporativa das pessoas [5], porém este aumento contém consequências. Nos últimos anos problemas com ataques cibernéticos passaram a ser cada vez mais comuns ao redor do mundo e inclusive no Brasil [2]. Em contra ponto, políticas e leis tornam-se mais rigorosas quanto ao tratamento e a proteção de dados [20], levando a segurança da informação e computacional a um novo nível de importância.

Para prevenção contra infortúnios, muitas empresas estão utilizando técnicas ofensivas para monitorar o nível de segurança e aperfeiçoar suas defesas, já que é mais fácil corrigir uma vulnerabilidade ao ter o conhecimento dela [31]. Existem, porém, complicações e barreiras no aprendizado de segurança ofensiva, uma vez que há uma linha tênue entre legalidade e ilegalidade quando se realiza testes em sites de terceiros.

Visto a lacuna existente na área para aplicar o conhecimento teórico de forma prática, este trabalho se propõe a desenvolver um ambiente para aprendizado e prática de técnicas ofensiva em aplicações web vulneráveis. Ambiente este, desenvolvido em Docker para facilitar a execução do mesmo em múltiplos Sistemas Operacionais, permitindo o estudo dessas técnicas sem necessidade de alocar tantos recursos computacionais como outros mecanismos de virtualização fazem [32].

O ambiente proposto é constituído de uma imagem Docker contendo uma aplicação web que ficará disponível para acesso local, a aplicação conterá algumas falhas selecionadas, que em sua maioria podem ser encontradas no OWASP TOP 10:2021. O ambiente foi construído em Docker para facilitar a instalação e otimizar a ocupação de recursos computacionais, uma vez que a maioria dos ambientes com vulnerabilidade estão disponíveis como imagens para Máquinas Virtuais.

OWASP Top 10 que é um framework da OWASP, uma organização sem fins lucrativos, que traz informações sobre as falhas mais utilizadas em um determinado ano. Neste trabalho o OWASP Top 10 utilizado será do ano de 2021 [23]. Desta forma, foram selecionadas falhas da OWASP Top 10:2021, pois como são frequentemente encontradas falhas novas de segurança [13], algumas técnicas passam a ser mais usadas e outras caem em desuso. Por isso, é importante para quem estuda tais técnicas, praticar em falhas atuais.

## 1.1 OBJETIVOS

### 1.1.1 Objetivo geral

O objetivo geral deste trabalho é realizar a implementação de um ambiente em Docker contendo aplicações web vulneráveis.

### 1.1.2 Objetivos específicos

Os objetivos específicos são os seguintes:

- Elencar o estado da arte das vulnerabilidades em aplicações web;
- Selecionar as vulnerabilidades mais pertinentes para estudo por estudantes/profissionais interessados no atual estado da arte;
- Desenvolver o ambiente em Docker de modo que sua instalação e uso por terceiros fique simplificada, deixando o foco da atividade no estudo das falhas propriamente dito.

# 2 AMBIENTES DE TESTES

O estudo em segurança cibernética tem um início lento e difícil por inúmeros motivos, dentre eles a necessidade de bases em outras áreas para uma compreensão do que é feito para explorar determinadas vulnerabilidades, além disso, existe a necessidade de amparo legal para as ações referentes aos testes de vulnerabilidades realizados. Não é possível apenas testar uma vulnerabilidade em um site qualquer, pois tais testes podem ser interpretados como uma tentativa de invasão e levar a problemas legais. E por isso ambientes de teste tem muitas utilidades dentro de cada tema explorado para estudo.

Parte das dificuldades encontradas na área de aprendizado de segurança cibernética podem ser sanadas pela utilização de ambientes de teste. O mesmo não tem o mesmo valor que uma aplicação real, mas pode ser muito útil para pessoas que querem testar novas ferramentas, explorar novas vulnerabilidades ou iniciar o seu estudo na área da segurança da informação.

Um ambiente de testes, consegue isolar os testes para a própria máquina local e isso na segurança da informação é muito importante, já que não é possível para *pentesters* ou profissionais que estejam testando vulnerabilidades, testarem aplicações reais sem as devidas permissões. Ao mesmo tempo, realizar determinados testes em aplicações reais pode gerar custos desnecessários e perdas econômicas no caso da indisponibilidade do serviço, acontecimento este que poderia ser mitigado utilizando um ambiente próprio para a validação de vulnerabilidades da aplicação.

Profissionais de segurança da área defensiva da segurança da informação podem utilizar destes ambientes para entender melhor certas falhas e com base nisto aperfeiçoar seus próprios sistemas sem ter que colocar em um primeiro momento seus sistemas a prova. Assim, um ambiente *Sandbox* para testes de vulnerabilidade se torna útil não apenas aos profissionais que querem se aperfeiçoar no ataque de sistemas, mas também aos profissionais de cibersegurança defensiva que querem melhorar a segurança de suas aplicações.

Existem vários ambientes de segurança que podem ser utilizados para a realização de testes de vulnerabilidade, desenvolvidos em diversas tecnologias. Os mais comuns estão disponíveis como aplicações web ou imagens de máquinas virtuais.

É importante que estes ambientes, quando disponíveis para execução em máquinas de usuário, sejam executados localmente de forma isolada, uma vez que as vulnerabilidades encontradas nas aplicações podem resultar em vulnerabilidades mais graves de rede e/ou sistema operacional. Alguns bons exemplos de ambientes e plataformas que aulixiam no aprendizado de tecnicas de segurança ofensiva, são:

1. Mutillidae II
2. VulnHub
3. TryHackMe
4. ImmersiveLabs

## 2.1 MUTILLIDAE II

A Mutillidae II é uma aplicação web open-source que oferece mais de 40 vulnerabilidades e desafios com vulnerabilidades encontradas desde a OWASP TOP Ten de 2007 até a de 2017. O ambiente conta com tutoriais e dicas para auxiliar no aprendizado e possui várias falhas simples de sempre exploradas para estimular o aprendizado da área [22].

Entre as vulnerabilidades propositalmente expostas estão, mas não se limitam a, SQL Injection, XSS e inclusão de arquivos arbitrários. Na Figura 1 é possível ver uma das páginas da Mutillidae II.

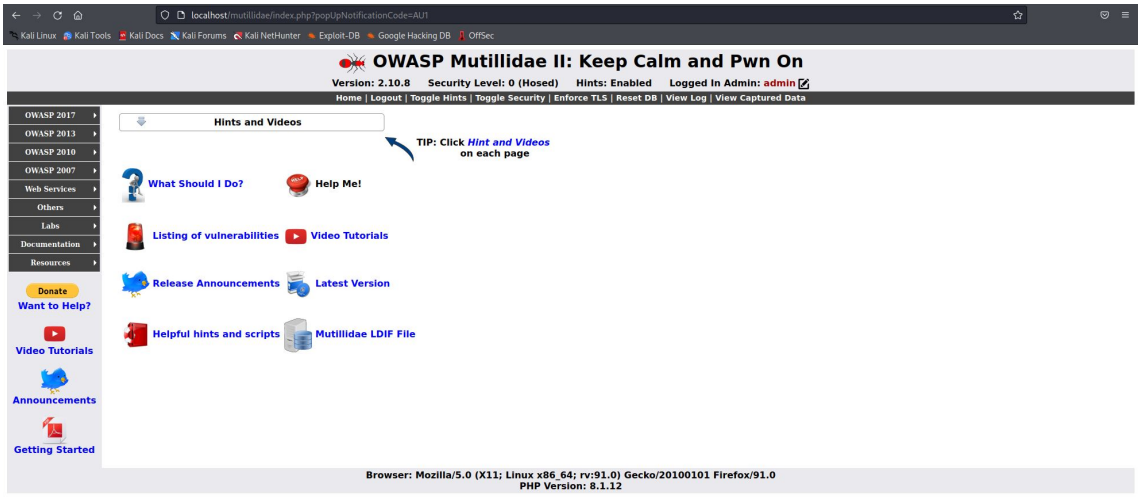


Figura 1 – Captura de tela da Mutillidae II

## 2.2 VULNHUB

A VulnHub opera como uma plataforma que disponibiliza máquinas virtuais propositalmente vulneráveis para auxiliar no aprendizado da área de segurança da informação. Essas máquinas virtuais ficam à disposição no site para download e são normalmente divididas por dificuldade, tendo níveis variados para auxiliar desde iniciantes na área de segurança até profissionais mais experientes.

As máquinas são projetadas como simulações de ambientes reais, simulando redes, servidores, banco de dados, aplicações e outros. Por tanto, os contextos de aplicações possíveis em máquinas virtuais são inúmeros e as possibilidades de vulnerabilidades também.

A plataforma também conta com um fórum que permite com que usuários discutam sobre as vulnerabilidades encontradas nas máquinas virtuais e se auxiliem nas soluções e aprendizado [9].

Na Figura 2 é possível ver a página inicial do VulnHub.

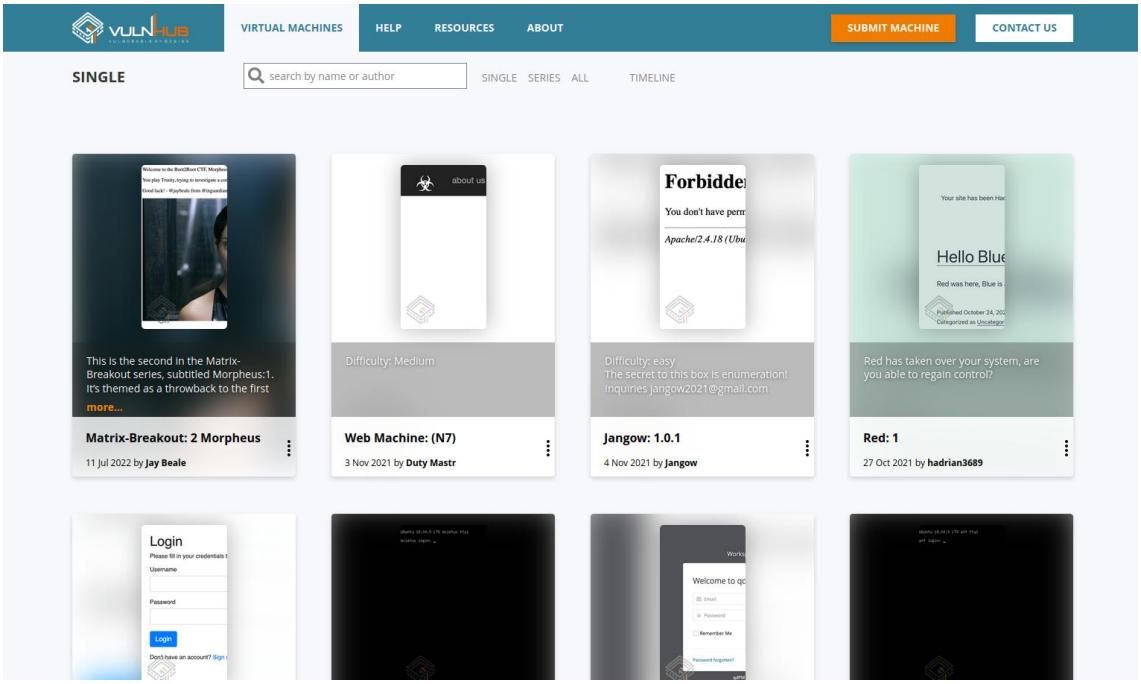


Figura 2 – Captura de página web do VulnHub

## 2.3 TRYHACKME

TryHackMe é uma plataforma não apenas de disponibilização de laboratórios de segurança, mas também de treinamentos de segurança, sobre as áreas de análise de vulnerabilidade, testes de penetração, bases da segurança e área correlatadas. Uma plataforma que não foca em apenas um tipo de usuário, mas disponibiliza materias, laboratórios e treinamentos para vários níveis de expertise em segurança da informação.

Possui como ambientes algumas máquinas virtuais pré-configuradas com aplicações e serviços propositalmente vulneráveis, simulando cenários reais. Também possui ambientes não apenas locais, desafios e Captures the Flag que funcionam para estimular o pensamento de cibersegurança em cenários diferentes do habitual. A TryHackMe conta com suporte técnico e com um forum para discussões dos temas da plataforma [29].

Na Figura 3 é possível ver uma das páginas da TryHackMe.

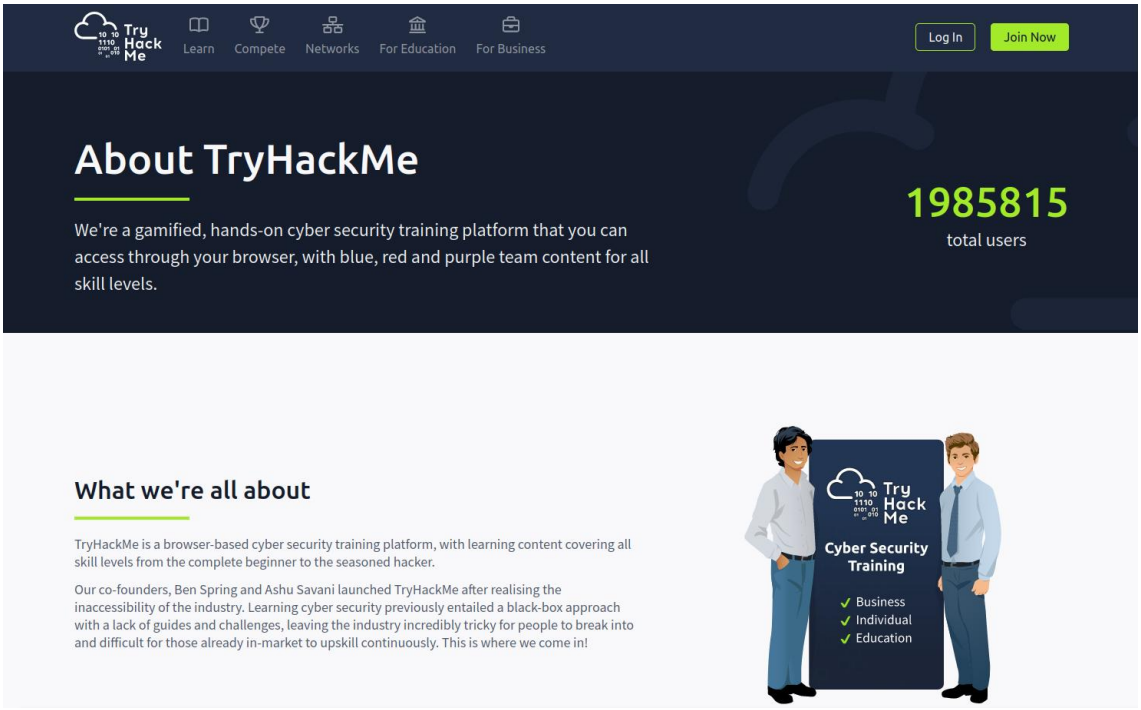


Figura 3 – Captura de página web do TryHackMe

## 2.4 IMMERSIVELABS

ImmersiveLabs é uma plataforma de aprendizado de segurança, contudo é uma plataforma com uma abordagem mais prática e mais voltada para execução de laboratórios de segurança virtuais com inúmeros temas e objetivos, alguns até mesmo com objetivos de prática forense. Parte dos Capture the Flags criados pela ImmersiveLabs são baseados em situações reais [30].

A plataforma conta com vários auxílios de aprendizado, utilizando de metologias de aprendizado adaptativo, indicadores e métricas de desempenho baseadas nas ações dos usuários sobre os Laboratórios. Como suporte, a plataforma disponibiliza vídeos, artigos e exercícios ao usuário.

## 2.5 CONSIDERAÇÕES

Existem inúmeros ambientes de aprendizado de segurança cibernética ofensiva, parte deles é disponibilizado de forma online, parte deles são disponibilizados como o download de máquinas virtuais. Este trabalho desenvolverá um ambiente semelhante à aplicação Mutillidae II, mas seu desenvolvimento será executado em outras tecnologias para serem disponíveis via imagens Docker, tornando o ambiente mais leve, uma vez que não necessita da virtualização de um SO, usando a Docker Engine para compartilhar os recursos do Kernel [24].

# 3 VULNERABILIDADES

Vulnerabilidades são falhas encontradas em um serviço, sistema, rede, aplicativo ou em algum processo que permita a exploração e por conseguinte a execução de operações indevidas. Existem inúmeros motivos para a existência de uma vulnerabilidade, envolvendo falhas introduzidas por programadores, configurações impróprias de sistemas, a não atualização de serviços, entre outros. Seus impactos, assim como são motivos, são muito variáveis, alguns resultam vazamentos de dados outros podem chegar a conceder o controle total de um sistema a um atacante [10].

As ameaças podem ser entendidas como todo e qualquer vetor de ação que utilize uma vulnerabilidade para a realização de uma operação indevida. Então não apenas atacantes mal-intencionados ou malwares são considerados ameaças, mas também usuários, administradores de sistema e até mesmo complicações em ambientes físicos, como um impacto em um disco de armazenamento ou um furacão, pois estes são potenciais ameaças ao pleno funcionamento de um sistema [10].

O nome dado para a técnica, código ou ferramenta utilizada para explorar uma vulnerabilidade é *exploit*. Os objetivos dos exploits variam conforme a vulnerabilidade que exploram, e eles são divididos entre os exploits conhecidos e os desconhecidos, que tão são chamados de *0-day*. Contra o primeiro grupo de exploits existem normalmente medidas a serem tomadas para mitigar as vulnerabilidades, já contra o segundo grupo não apenas a mitigação é dificultada como a própria identificação da vulnerabilidade [1].

## 3.1 OWASP

A OWASP é uma organização sem fins lucrativos que como projeto open-source aspira auxiliar no desenvolvimento de aplicações e ambientes mais seguros. Dentre seus projetos existe a OWASP TOP:10 focado em informar aos profissionais de segurança e demais pessoas interessadas no estado atual da segurança mundial quais tipos de vulnerabilidades tem sido mais exploradas, chamando atenção para estas falhas mais comumente exploradas por invasores e mostrando assim aspectos novos a serem observados com mais atenção nas aplicações. É importante frisar que vários tipos de vulnerabilidades são difíceis de serem quantificadas e monitoradas, por isso a OWASP não se baseia apenas em números de detecções, mas também no estado da arte, publicações e estudo de vulnerabilidades atuais [23].

No ano de 2021 a OWASP TOP:10 publicou um novo informativo, classificando distintamente os tipos de vulnerabilidades antes previstos, criando categorias e alterando também o ranking de cada tópico. Neste ano os tópicos se baseiam muito na adequação as leis que envolvem a proteção de dados dos usuários. Na Figura 4 são apresentados os 10 maiores riscos do ano de 2021 segundo a OWASP e a transição do ranking de 2017 para o mais recente, de 2021.

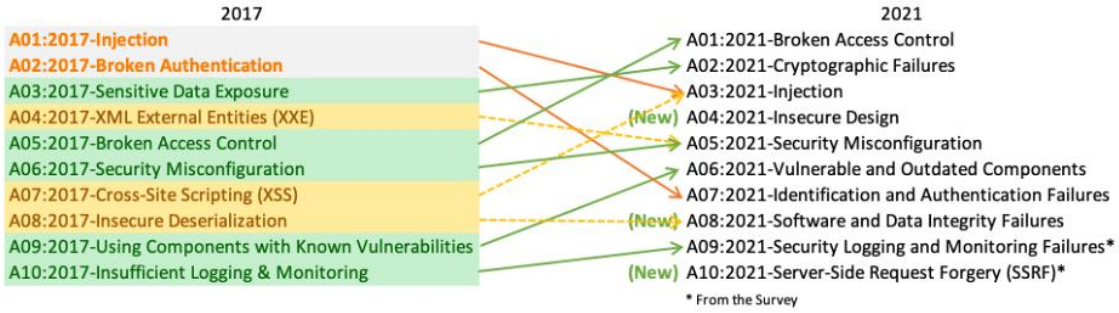


Figura 4 – OWASP Top Ten 2017 vs 2021 [23]

## 3.2 BROKEN ACCESS CONTROL

Broken Access Control é um tipo de vulnerabilidade caracterizada pelo acesso a um ativo indevidamente por um usuário, sistema ou dispositivo não autorizado [12]. Em sistemas as permissões de acesso deveriam ser bem definidas. Quando um sistema, que precisa segregar funções ou recursos, não implementa corretamente as permissões baseadas em acesso, independe se por meio de sessão, autorização ou autenticação, esse sistema se torna vulnerável. Os ataques de Broken Access podem se dar por acessos direto em URL, manipulação de parâmetros, previsibilidade de sessões, privilégios excessivos, falta de verificação de autorização, entre outros.

Um sistema de hotelaria, por exemplo, não deveria permitir que um usuário autenticado como um comprador crie anúncios de quartos de Hotel através de um perfil administrativo. Na máquina Mutillidae II [7] existe um exemplo de Broken Access Control, causada por cada usuário ter um *uid* fixo e ordenado crescentemente. Para explorar a falha basta inserir credenciais válidas de qualquer usuário e alterar o *uid* enviado na requisição em um valor de perfil com acessos administrativos.

## 3.3 CRYPTOGRAPHIC FAILURES

Cryptographic Failures são as vulnerabilidades que expõem dados sensíveis de uma aplicação por conta de uma criptografia fraca ou inexistente [19]. Os dados podem ir desde informações médicas até senhas ou números de cartão de crédito. Nas aplicações modernas os dados passam por manipulações de dados em repouso e em trânsito, tornando ainda mais necessário um controle rígido de criptografia para que essas vulnerabilidades sejam dificilmente aproveitadas. Este tipo de vulnerabilidade estava em terceiro lugar no ano de 2017, mas subiu para segundo lugar no ano de 2021 [23], pois problemas de segurança causados por senhas *hard-coded* tem se tornado muito comuns.

Por mais que o nome “Falhas criptográficas” pareça intuitivo, este tópico não fala apenas sobre problemas do processo de criptografia, ele envolve também atividades ligadas aos processos de criptografia, como o gerenciamento correto das chaves de segurança, a escolha do

tipo de algoritmo criptográfico e os cuidados com informações criptográficas.

## 3.4 INJECTION

Injection é a classe de vulnerabilidades que engloba as falhas que ocorrem no momento que um dado é injetado em uma aplicação e consegue com isso alterar o comportamento planejado da aplicação, tornando, por exemplo, uma simples *string* que serviriam como nome em um código malicioso executável [14]. Essas falhas geralmente ocorrem por uma falta de validação das entradas ou uma neutralização imprópria das entradas das aplicações.

Como exemplo de Injection temos um caso de SQL Injection, onde se a aplicação tiver dentro de seu código uma execução semelhante a “query = ’SELECT * FROM users WHERE id=’ + getParamer("id") + ’;’” seria possível dar como id o valor “’ or 1=1’” e alterar o sentido original da query já que ela foi projetada para retornar os usuários onde o *id* é igual ao informado, mas com esta injeção toda a tabela é retornada como resposta.

## 3.5 INSECURE DESIGN

Insecure Design é a classe de vulnerabilidades existentes quando um projeto possui problemas de segurança de forma que mesmo que as melhores práticas de desenvolvimento sejam utilizadas o sistema ainda continuará a ser inseguro. Então a causa da falha já faz parte do design escolhido para o projeto que podem levar a, por exemplo, vazamento de informações da aplicação [21].

As vulnerabilidades deste tópico da OWASP decorrem na concepção de um projeto de sistema, aplicativo ou infraestrutura principalmente quando os riscos não são corretamente identificados e avaliados, quando algum dos princípios de segurança não são cumpridos como o de menor privilégio ou de defesa em profundidade, quando existe negligência na criptografia dos dados armazenas e em transmissão, quando existe um design não seguro de controle de acessos e autenticação, quando o projeto não passa por testes de segurança.

Uma das vulnerabilidades da classe é justamente a "CWE-209 Geração de mensagem de erro contendo informações confidenciais", erros como armazenamento desprotegido de credenciais ou violação de limite de confiança também vão ser vulnerabilidades desta classe.

## 3.6 SECURITY MISCONFIGURATION

Security Misconfiguration em resumo são as falhas de segurança motivadas por falta de configuração ou configuração padrão em aplicações e/ou serviços [8]. Quando um banco de dados é configurado com valores padrão e executado em porta padrão, ele carrega inúmeros problemas e falhas de segurança. Falhas que podem ser facilmente encontradas por códigos maliciosas que procuram automaticamente serviços executando em determinadas portas. Nas Security Miscunfigurations também estão faltas por falta de configurações de header.

As vulnerabilidades ligadas ao tópico 5 da OWASP Top Ten:2021 estão presentes em algumas camadas de sistema, como: configurações de segurança de servidores web, configurações de segurança de banco de dados, configurações de segurança de redes, configurações de autenticação e controle de acesso. Essas vulnerabilidades, então, causar impacto de exposição de informação, comprometimento parcial ou total de sistema, interrupção de serviços e violações de conformidade.

## 3.7 VULNERABLE AND OUTDATED COMPONENTS

Vulnerable and Outdated Components é a classe que engloba as vulnerabilidades que envolvem as falhas por conta de componentes de software datados ou que possuem falhas conhecidas adicionadas ao serviço em questão [11]. Durante o desenvolvimento de novos serviços é difícil criar tudo do zero, sendo comum, que sejam utilizados componentes prontos de terceiro, o problema é que quando um componente com falha é adicionado em uma aplicação, essa aplicação obviamente herda a vulnerabilidade que pode ou não ter algum peso no contexto da aplicação.

Sobre componentes, é importante entender que são serviços também expostos a riscos de segurança, questões de integridade também podem afetá-los e isso significa que não é pelo software estar atualizado que ele está seguro, já que o fornecedor de uma desses componentes pode ter sofrido com alguma vulnerabilidade e agora está distribuindo uma nova versão comprometida de seu sistema que, por exemplo, repasse os dados transmitidos para um banco de dados de atacantes ou permita uma execução remota através do seu componente.

## 3.8 IDENTIFICATION AND AUTHENTICATION FAILURES

Identification and Authentication Failures é a classe que aloca as falhas que envolvem controle de autenticação, como quebra em gerenciamento de sessão de usuários por má implementação do serviço [28]. Essas falhas podem ter origens como: não tratamento para força bruta, permissão de senhas fracas, permissão de usuários default, processos ineficiente de recuperação de credenciais, e reutilização de identificadores de sessão. Muitas ferramentas são desenvolvidas para automatizar a exploração dessas vulnerabilidades, no caso de não tratamento de força bruta é possível explorar com ataques de dicionário.

Quando um sistema não garante a identificação única de um usuário, permitindo que um identificador seja duplicado, este sistema está permitindo que outros usuários assumam determinadas identidades, isso pode resultar no controle falho dos recursos do sistema e/ou gerar outros tipos de vulnerabilidade, uma vez que um atacante consiga se identificar como outro usuário, ele pode confundir os mecanismos de registro da aplicação, realizar atividades indevidas e culpabilizar outra pessoa.

## 3.9 SOFTWARE AND DATA INTEGRITY FAILURES

Software and Data Integrity Failures é a classe de vulnerabilidades que se aproveitam da falta de confirmação de integridade do software e de dados [27]. Quando em um ambiente, podem levar a corrupção de dados, manipulação de informação, violação de privacidade, entre outros. Pode ocorrer, por exemplo, quando uma aplicação usa fontes externas não confiáveis, quando não existe um pipeline seguro no CI/CD, quando um invasor pode obter acesso a dados e/ou estruturas que possam ser vistas e alteradas.

Múltiplos tipos de vulnerabilidade podem afetar um sistema e sua integridade. Uma vulnerabilidade de injeção pode realizar alterações nos dados do sistema, comprometendo a integridade dos dados e informações. Uma falha no ambiente pode permitir que softwares maliciosos infectem o sistema e comprometam componentes, fontes, configurações, entre outros. Um erro humano pode resultar na manipulação indevida de dados, na configuração incorreta de segurança durante uma atualização, na indisponibilidade do sistema e assim por diante.

## 3.10 SECURITY LOGGING AND MONITORING FAILURES

O item Security Logging and Monitoring Failures da OWASP TOP Ten vai além de apenas classificar falhas, mas visa abordar os tópicos de resposta e monitoramento dos ataques e ações rotineiras realizadas em um ambiente [3]. Esta etapa se mistura com vários itens do OWASP TOP Ten, uma vez que para um correto monitoramento é necessário ter os logs armazenados seguramente, portanto um design correto para isso é necessário assim como confirmar a integralidade dos logs para se necessário usar os mesmos como provas. Dentro dessa classe de falhas é importante entender que além de logs insuficientes nas aplicações, logs com dados sensíveis também podem gerar problemas de segurança e logs com dados demais podem dificultar o monitoramento.

Além dos registros de log, um sistema se torna vulnerável quando não tem monitoramentos em tempo real. Entre inúmeros riscos envolvidos de se ter registros de ação, mas sem monitoramento em tempo real é a demora no tempo de resposta a incidente contra ataques DDoS, causando a indisponibilidade do sistema por uma medida não ter sido tomada a tempo. Mesmo que as requisições tenham sido registradas e identificadas, o sistema já foi comprometido em sua disponibilidade.

## 3.11 SERVER-SIDE REQUEST FORGERY

Server-Side Request Forgery como o nome diz, é uma vulnerabilidade causada pela falsificação de uma requisição do lado do servidor e ela acontece quando o servidor recupera o conteúdo de uma requisição, mas por não validar o URL acaba não garantindo que a requisição chegue ao destino esperado, esta falha tem um CWE mapeado que é a CWE-918 [15]. Esta

vulnerabilidade consegue enviar as requisições a destinatários mesmo quando protegidos por firewall, VPN ou outro ACL.

Um exploit em um SSRF geralmente se baseia na validação ou filtragem de requests de usuários mal implementada. Um atacante obtedo sucesso pode explorar o sistema de dentro, recuperando chaves de acesso, tokens, dados internos, informações sensíveis, pode realizar outros tipos de ataque como de negação de serviço para o próprio serviço e para aplicações externas, usando o sistema invadido como uma ferramenta de ataque.

## 3.12 SELEÇÃO DE VULNERABILIDADES

Sendo o OWASP Top Ten uma referência em classificação de vulnerabilidades mais relevantes em criticidade e quantidade de um intervalo de tempo, o projeto se torna um bom ponto de partida para a seleção de vulnerabilidades. Selecionar vulnerabilidades relacionadas a OWASP enfatiza um estudo de impacto imediato no estado da arte da segurança cibernética.

O trabalho visa montar um ambiente para prática de técnicas de segurança ofensiva, por tanto, se faz necessário a existência de instruções básicas de como explorá-las. A OWASP traz uma abordagem sistemática das vulnerabilidades, deixando as informações de forma estruturada, porém, importante ainda frisar que esse trabalho não se propõe a utilizar apenas vulnerabilidades documentadas pela OWASP e sim abordar os tópicos selecionados como um ponto de partida e quando de interesse comum, recorrer às vulnerabilidades mapeadas.

Alguns tópicos do OWASP Top Ten podem não ser tão instigantes de serem desenvolvidas neste trabalho, uma vez que tenham suas falhas mais comuns, tenham um envolvimento maior com segurança defensiva e compliance, como as vulnerabilidades do tópico A09:2021-Security Logging and Monitoring Failures. Com base nas informações deste capítulo foram escolhidas 9 CWEs (Common Weakness Enumeration) nos tópicos da OWASP para se transformarem em 10 vulnerabilidades do ambiente, estas são CWEs dividas nos tópicos do:

- A01:2021 - Broken Access Control
  - CWE-284: Improper Access Control - Nível de criticidade: média
  - CWE-285: Improper Authorization - Nível de criticidade: alta
- A02:2021 - Cryptographic Failures
  - CWE-200: Exposure of Sensitive Information to an Unauthorized Actor - Nível de criticidade: alta
- A03:2021 - Injection
  - CWE-79: Improper Neutralization of Input During Web Page Generation - Criticidade média
  - CWE-400: Uncontrolled Resource Consumption - Nível de criticidade: crítica

  - CWE-89: Improper Neutralization of Special Elements used in an SQL - Nível de criticidade: alta
  - CWE-94: Improper Control of Generation of Code - Nível de criticidade: crítica

- A05:2021 - Security Misconfiguration
  - CWE-22: Improper Limitation of a Pathname to a Restricted Directory - Nível de criticidade: alta

- A08:2021 - Software and Data Integrity Failures
  - CWE-784: Reliance on Cookies without Validation and Integrity Checking in a Security Decision - Nível de criticidade: alta

Mais detalhes das vulnerabilidades serão expostos neste trabalho nos capítulos 5 e 6. O nível de criticidade é baseado não apenas na vulnerabilidade específica, mas também em seu potencial sobre o sistema.

# 4 TECNOLOGIAS DE DESENVOLVIMENTO

Existe uma similaridade entre as falhas das aplicações da OWASP Broken Web Applications e o ambiente proposto neste trabalho, os tipos de falhas encontrados nesta máquina são muito semelhantes às falhas desenvolvidas neste ambiente, porém a grande diferença é que este ambiente será desenvolvido em Docker com a intenção de facilitar a instalação e diminuir a ocupação de dados alocados em máquina. Enquanto uma imagem ISO Ubuntu tem cerca de 5Gb e a OWASP BWA tem 1.7Gb, todos os containers docker desenvolvidos neste trabalhos juntos ocupam cerca de 1Gb.

## 4.1 DOCKER

O Docker é um sistema que automatiza a implementação e gerenciamento de *containers*. O Docker diferente dos sistemas de virtualização não necessita de Hardware virtualizado e nem necessariamente de uma instalação completa de um S.O. dentro do *container* [6]. Para seu funcionamento ele usa algumas *features* do *kernel* como o Cgroups e o namespaces, essas features são utilizadas para rodar os processos independentemente. Este modelo permite uma melhor organização da infraestrutura e, em simultâneo, mantém os ambiente seguros já que eles são separados.

A Figura 5 compara a virtualização de um Virtual Machine e uma Virtual Environment.

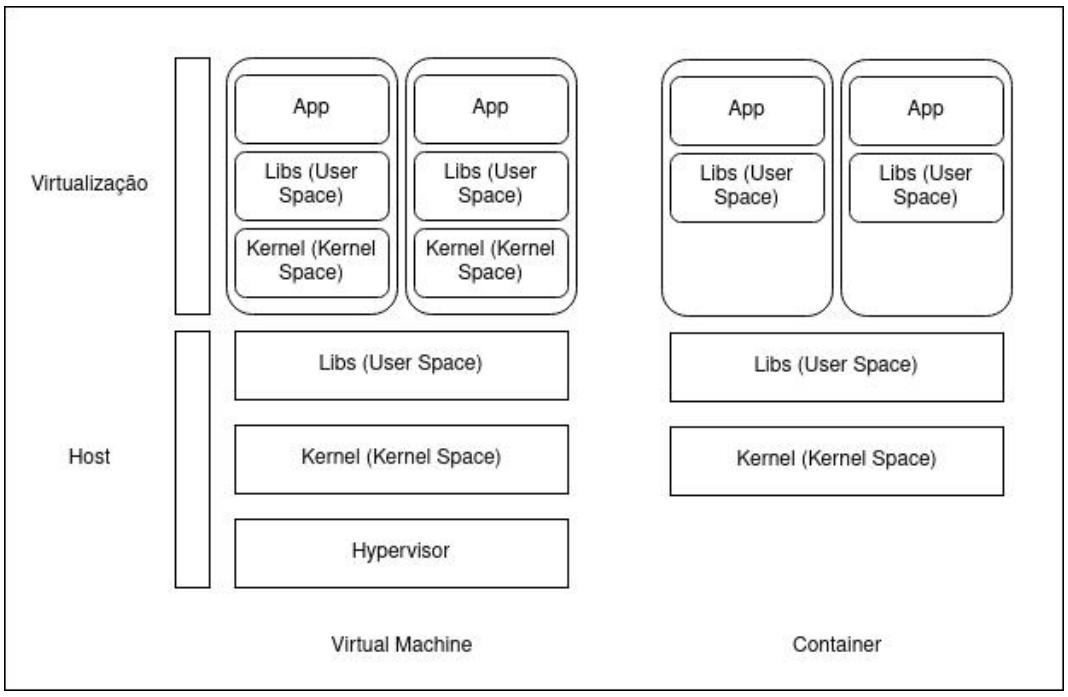


Figura 5 – Virtual Machine Vs Virtual Environment.

A principal característica de um container, que permite com que ele seja mais leve, é o compartilhamento de kernel. Na figura 5 é possível perceber que para que a virtualização de uma máquina virtual aconteça, um espaço de kernel é alocado, enquanto nos containers isso

não é necessário. Ao mesmo tempo que o Docker compartilha o kernel, ele isola os processo para que o host não seja acessível aos containers sem a devida permissão, então ao rodar um comando que liste os processos dentro do container, apenas os processos internos serão exibidos [17].

O ambiente foi desenvolvido com base em três imagens docker, para permitir que sejam exploradas não apenas falhas em aplicações web e em banco de dados, mas falhas em aplicações web que levem a falhas em nível de sistemas operacional, as imagens base usadas foram: mysql:5.7, node:16-alpine e alpine:latest.

As aplicações executadas dentro do Docker estarão por padrão disponíveis apenas ao Host localmente para evitar que as vulnerabilidades sejam exploradas por atacantes.

A imagem mysql:5.7 será responsável por armazenar o banco de dados da aplicação, a imagem node:16-alpine será responsável pelo front-end do projeto e a imagem alpine:latest ficará responsável pelo alpine:latest deste trabalho.

O projeto será configurado para rodar localmente, e realizar as conexões internamente, não permitindo que as aplicações sejam visualizadas pelo mundo exterior. Assim não haverá problemas com tráfego mal-intencionado na rede. Esta configuração se dará através do arquivo de build do docker-compose, sendo explorada no Capítulo 5.

## 4.2 NODEJS

Na aplicação de back-end a tecnologia de desenvolvimento será o NodeJS que é um *runtime* de JavaScript que funciona em forma de execução assíncrona, permitindo a execução de código sem a necessidade de esperar o retorno de requisições [18]. O Node foi criado pensando também em aplicações web escaláveis, sendo uma plataforma open-source multi-paradigmas.

Neste trabalho, o Node será utilizado como um provedor de API, requisitando dados do banco de dados da aplicação e executando operações em SO para disponibilizar arquivos e executar operações. Algumas bibliotecas serão utilizadas em conjunto ao NodeJS para possibilitar as funcionalidades citadas acima, estas são: cors, dotenv, express, mysql2 e sequelize.

## 4.3 REACT

O React é uma biblioteca do JavaScript que serve para a criação de interfaces de usuário. O React trabalha pensando em reaproveitamento de código criando componentes e pensando numa maior facilidade para descrição de estados e de conexão entre o CSS, JavaScript e HTML [26].

O React neste trabalho, ficará responsável por gerir as rotas da aplicação web, fazer a conexão com o back-end, renderizar a UI do projeto e será também a origem de vulnerabilidades, que serão explicadas no capítulo 5. Algumas bibliotecas serão utilizadas em conjunto ao React para possibilitar as funcionalidades citadas acima, estas são: axios, web-vitals, react-dom, react-router-dom, react-scripts.

## 4.4 MYSQL

O MySQL é um bando de dados relacional open-source prático e acessível mantido pela Oracle Corporation. Este banco possuí alguns problemas de segurança, mas justamente isso também permite explorar outras vulnerabilidades no ambiente [16].

O MySQL será utilizado neste trabalho para auxiliar nos controles de acesso de usuário na aplicação mantendo informações de usuário e credenciais. Ele será outro vetor de ataques da aplicação web.

## 4.5 TECNOLOGIAS NO AMBIENTE

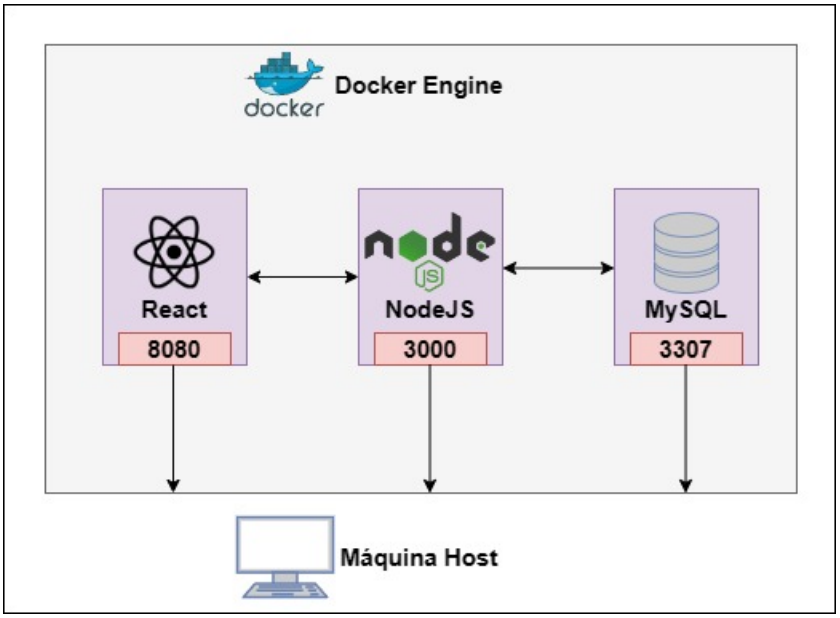


Figura 6 – Diagrama do ambiente projetado

Na Figura 6 é possível visualizar o diagrama do ambiente desenvolvido, contendo três containers docker rodando pela Docker Engine na máquina Host. Um container rodando a UI da aplicação em React, um container MySQL armazenando os dados e um container NodeJS servindo como API para disponibilizar dados e serviços.

# 5 DESENVOLVIMENTO

Este capítulo primeiramente apresentará como foi realizado a configuração inicial do ambiente com docker, react, node e mysql. Em um segundo momento, serão apresentadas as vulnerabilidades encontradas no sistema, uma breve revisão delas e como foram implementadas no sistema.

## 5.1 CONFIGURAÇÕES INICIAIS DO SISTEMA

O controle de build e disponibilidade do sistema foi configurado por um arquivo chamado *docker-compose.yml*, este arquivo, divide os serviços do projeto em três sessões, a ordem que devem ser buildados, os volumes e as conexões de rede entre eles. Na Figura 7, se tem a definição do serviço 'mysqldb', responsável pelo banco de dados do ambiente.

```
services:
  mysqldb:
    image: mysql:5.7
    restart: unless-stopped
    env_file: ./.env
    environment:
      - MYSQL_ROOT_PASSWORD=$MYSQLDB_ROOT_PASSWORD
      - MYSQL_DATABASE=$MYSQLDB_DATABASE
    ports:
      - $MYSQLDB_LOCAL_PORT:$MYSQLDB_DOCKER_PORT
    volumes:
      - db:/var/lib/mysql
    networks:
      - backend
```

Figura 7 – Configuração do serviço de MySQL

Este arquivo está definindo que a imagem Docker base para o serviço deve ser a "mysql:5.7", informar que o container deve ser reiniciado sempre que for interrompido por algum problema a menos que seja parado manualmente, define o arquivo de *environment* como "./.env", configura algumas variáveis de ambiente que precisam ser predefinidas para o serviço ser corretamente iniciado, seta as porta que o serviço será disponibilizado na máquina host, cria um volume para a persistência dos dados e configura o serviço para executar no network denominado "backend".

Na Figura 8, se tem a definição do serviço *bwa-api*, responsável pelo back-end da aplicação.

```
services:
  bwa-api:
    depends_on:
      - mysqldb
    build: ./bwa-api
    restart: unless-stopped
    env_file: ./.env
    ports:
      - $NODE_LOCAL_PORT:$NODE_DOCKER_PORT
    environment:
      - DB_HOST=mysqldb
      - DB_USER=$MYSQLDB_USER
      - DB_PASSWORD=$MYSQLDB_ROOT_PASSWORD
      - DB_NAME=$MYSQLDB_DATABASE
      - DB_PORT=$MYSQLDB_DOCKER_PORT
      - CLIENT_ORIGIN=$CLIENT_ORIGIN
    networks:
      - backend
      - frontend
```

Figura 8 – Configuração do serviço de BWA-API

No código da Figura 8 está definindo que antes do build do “bwa-api”, o banco de dados deve ser executado, informar que no diretório “./bwa-api” deve ser procurado um Dockerfile com informações do build do serviço específico, define o arquivo de *environment* como “./.env”, configura algumas variáveis de ambiente que precisam ser predefinidas para o serviço ser corretamente iniciado, seta as porta que o serviço será disponibilizado na máquina host e configura o serviço para executar nos networks *backend* e *frontend*.

Na Figura 9, se tem a definição do serviço *bwa-ui*, responsável pelo front-end da aplicação.

```
services:
  bwa-ui:
    depends_on:
      - bwa-api
    build:
      context: ./bwa-ui
      args:
        - REACT_APP_API_BASE_URL=$CLIENT_API_BASE_URL
    ports:
      - $REACT_LOCAL_PORT:$REACT_DOCKER_PORT
    networks:
      - frontend
```

Figura 9 – Configuração do serviço de BWA-UI

No código da Figura 9 fica definido a ordem de *build* do sistema, e o *bwa-ui* deve ser montado após o *bwa-api*, seu build deve acontece com base no Dockerfile encontrado no path

"./bwa-ui" levando como argumento o "$CLIENT_APLI_BASE_URL", tendo a porta setada com base na variável de ambiente do docker-compose e configurando o serviço para executar no network "frontend".

Na Figura 10, os networks e volumes são definidos.

```
volumes:
  db:

networks:
  backend:
  frontend:
```

Figura 10 – Configuração de volumes e networks

Por fim, o arquivo *docker-compose.yml* representado na Figura 10, nomeia e define os volumes e networks do ambiente.

O MySQL por ter um uso mais padrão teve seu build completamente definido através do arquivo *docker-compose.yml*, em contraponto, o back-end e front-end da aplicação possuem necessidades específicas e por isso, um arquivo separado foi criado para cada um deles responsável por configurar seus *ENTRYPOINTS* assim como a imagem Docker base para a aplicação e transferir os arquivos de execução para dentro da imagem.

```
FROM alpine:latest

RUN apk add --update nodejs npm && apk add bash && apk add bind-tools

WORKDIR /bwa-api
COPY package.json .
RUN npm install
COPY . .
CMD npm start
```

Figura 11 – Dockerfile do back-end

Na Figura 11 é possível ver o Dockerfile do back-end definindo a imagem Docker base como "alpine:latest", as imagens *alpine* são mais leves e por isso foram escolhidas para este projeto. Nesta distribuição linux é então instalado o *NodeJS*, o *Bash* e o *bind-tools*. Em seguida o WORKDIR é definida para "/bwa-api", os arquivos do back-end são copiados para a pasta e o *Node* é inicializado.

```
FROM node:16-alpine

RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app

COPY package.json .
RUN npm install

COPY . ./

CMD ["npm", "run", "start"]
```

Figura 12 – Dockerfile do front-end

Na Figura 12 está o código Dockerfile do front-end definindo a imagem Docker base como “node:16-alpine”, copiando os arquivos do front-end para dentro do WORKDIR “/usr/src/app”, instalando as dependências e inicializando o serviço.

Com isso, as configurações de containers do ambiente estão concluídas. Agora, os serviços passam a ser o objeto de interesse de desenvolvimento. No back-end o arquivo “server.js” define o *setup*, seu conteúdo será explicado nos próximos parágrafos.

Primeiramente, ele configura as variáveis de ambiente baseados no arquivo “.env”, importa o *express* e a biblioteca *cors* para permitir as chamadas http ao servidor e realiza a criação da conexão com o banco ao importar o *db*.

O *express* é ajustado para realizar suas operações junto ao *cors*, que foi previamente configurado para expor alguns de seus *headers*, essa exposição foi feita para permitir a identificação correta de tipos de arquivos no front-end. Com o *express* definido, as rotas são importadas de outro arquivo, chamado “routes.js”.

É realizada uma migração de dados, com os dados de usuários base do sistema e por fim, o app *express* é disponibilizada por meio da porta encontrada nas variáveis de ambiente.

O arquivo “./bwa-api/app/models/index.js” cria a conexão com o banco de dados usando a biblioteca de ORM chamada *Sequelize*, também cria um objeto contendo o Sequelize e a conexão é criado, neste, é inserido um model de Usuários, que será definido através do arquivo “./user/model.js”.

Esse model de User é responsável pela criação da tabela de usuários no banco de dados, informando as tabelas e tipos. A tabela do model é bem simples, possuindo apenas 4 colunas: name, password, email e admin.

O *Router* do *Express* é utilizado para definir o tipo de requisição e o redirecionamento de funções. No arquivo de cada rota, existe a chamada para um segundo arquivo responsável pelo controle das funções, as receber uma requisição o arquivo direciona para a função responsável baseado na rota e no tipo de *request*. A função então realiza as operações necessárias e retornar um *payload*.

Para construção da UI do front-end foi utilizado o *React* e *ReactDOM*. Uma raiz de

renderização é criada através do “ReactDOM.createRoot()” e tem um *App* como *component* renderizado na aplicação. Já as chamadas http no front-end são feitas através da biblioteca *Axios*.

## 5.2 IMPLEMENTAÇÃO DAS VULNERABILIDADES

Com um ambiente construído e pronto para realizar atividades e operações que um sistema real faria, vulnerabilidades podem começar a ser implantadas. Este capítulo abordará a implementação das 10 vulnerabilidades no ambiente construído, fazendo um pequeno informe sobre as CWEs, a origem das falhas e como elas foram introduzidas na aplicação.

As vulnerabilidades serão divididas em duas sessões de 5 vulnerabilidades cada no sistema, a ideia é que as primeiras vulnerabilidades sejam menos nocivas ao ambiente e as últimas mais críticas. As primeiras ficarão na sessão Genesis e as segundas na sessão Apocalypse. O início da sessão Genesis servirá principalmente para o usuário entender o sistema e explorar as rotas, os códigos encontrados no *client* e testar algumas *injections*.

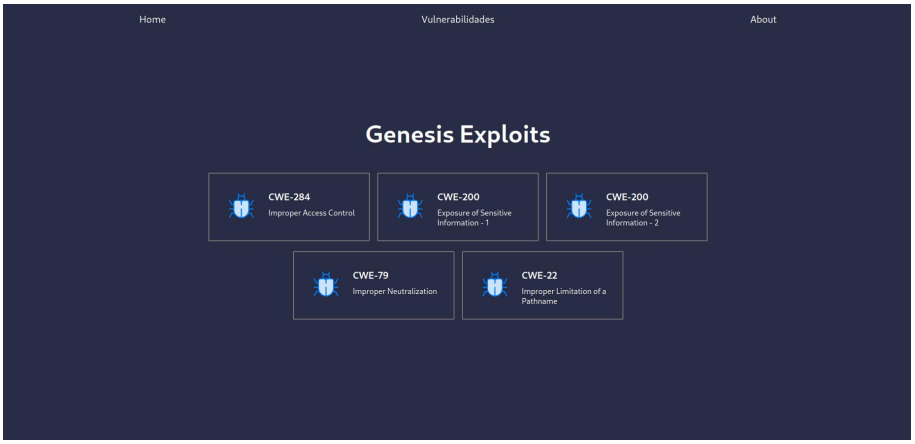


Figura 13 – Sessão Genesis


Figura 14 – Sessão Apocalypse

Nas Figuras 13 e 14 é possível observar as sessões Genesis e Apocalypse do trabalho respectivamente. Para acessar a tela com a vulnerabilidade e/ou dica sobre a vulnerabilidade, basta clicar em um dos tópicos CWE que aparecem nas sessões.

### 5.2.1 CWE-284: Improper Access Control

A CWE-284 é uma vulnerabilidade definida sobre quando um sistema falha em impor condições de proteção a recursos que deveriam ser destinados a usuários específicos, então quando um usuário não logado ou com acesso insuficiente consegue acessar uma página restrita e conseguir coletar os dados desta página, uma CWE-284 é explorada.

Nesta CWE também contam os casos onde o sistema permite o uso de senhas fracas ou quando o sistema permite escalação de privilégios, mas este trabalho se baseará numa página sem a implementação de validação de acesso privilegiado.

Para a criação desta vulnerabilidade não houve grandes dificuldades, uma vez que bastava criar uma rota sem realizar verificações de identidade com dados sensíveis e que deveriam em um primeiro momento serem destinadas a um administrado, para isso, a rota "/admin-control" foi designada, contendo uma página com informações de contas administrativas.

### 5.2.2 CWE-200: Exposure of Sensitive Information to an Unauthorized Actor

A CWE-200 representa uma categoria de vulnerabilidades onde acontece vazamento de informações confidenciais. Vazamentos que podem acontecer durante a transmissão dos dados ou pelos dados não serem devidamente protegido em uma aplicação. Este trabalho conta com duas vulnerabilidades da categoria CWE-200.

Para no sistema haver essas vulnerabilidades alguns arquivos ficaram disponíveis no front-end. O *path* que leva a um dos arquivos foi adicionado ao "disallow" do "robots.txt". O que parece uma medida de segurança, para não permitir a indexação dos arquivos por robôs, também realiza o vazamento de dados, informando uma URL acessíveis com dados sensíveis.

Além disso, no HTML de algumas páginas da aplicação, foram adicionados comentários vazando credenciais de segurança.

### 5.2.3 CWE-79: Improper Neutralization of Input During Web Page Generation

A categoria CWE-79 baseada na injeção de código em aplicações web, essa injeção de código pode originar um exploit de XSS (Cross-site scripting).

Contra esse tipo de ataque aplicações React possuem algumas defesas por padrão. Por padrão, as variáveis simples dentro React são carregadas como *string* independentemente se tiverem tags HTML no corpo do texto, isso garante um nível de segurança ao React conta DOM-base XSS exploits, porém isso ainda não torna as aplicações React totalmente seguras contra React.

Em um cenário onde se pretende que um HTML seja posteriormente renderizado na página, é possível passar uma propriedade no elemento HTML chamada “dangerouslySetInnetHTML”, a propriedade tem esse nome justamente pelo perigo que envolve utilizar ela em uma página [4]. Esta será usada para permitir um usuário escreva textos e elementos na página em tempo real, ao mesmo tempo, isso permite que exploits XSS sejam executados na página, o que é o objetivo dessa implementação. Na Figura 13 é possível ver o código desenvolvido vulnerável a XSS.

```
<div id="content">
  <h1>Escreva seu texto</h1>
  <div dangerouslySetInnerHTML={{"__html": aboutUserText}} />
  <div class='input-box-xss'>
    <input type='text' value={aboutUserText} onChange={this.handleChange}
       ↪ placeholder='Seu␣texto'/>
    {isRendered && (
      <div class="blank" dangerouslySetInnerHTML={{ __html: aboutUserText }} />
    )}
  </div>
</div>
```

Figura 15 – Código implementado vulnerável a XSS

### 5.2.4 CWE-22: Improper Limitation of a Pathname to a Restricted Directory

A categoria da CWE-22 representas as falhas de sistema e aplicação em limitar e restringir o acesso a diretórios e/ou pathnames. Uma falha como essa pode permitir que um usuário tenha informações restritas da aplicação e/ou sistema, consiga recuperar arquivos e em casos mais graves até mesmo realizar modificações no sistema.

Ocorrem geralmente quando existem entradas externas para a criação do *path* de um determinado diretório, porém a entrada não é neutralizada e valores como “../” conseguem participar do *path*. Isso pode permitir, por exemplo, que diretórios indevidos sejam acessados como entradas do estilo “../../<archieve_name>”.

Justamente essa vulnerabilidade será implementada no projeto. Uma página no front-end foi criada, nela uma lista de nomes de arquivos com descrições de animais foi disposta e ela pede que o usuário informe o nome de um arquivo para download. A entrada esperada seria algo como “abelha.txt”, mas como o nome é usado na construção do *path* do diretório sem corretas neutralizações de valores, o usuário pode recuperar outros arquivos de sistema.

No front-end, o campo inserido pelo usuário é adicionado ao fim da *string* “../../img/” e enviado como requisição ao back-end. No back-end esse valor é recuperado e adicionado ao *path* atual em que este arquivo está rodando, então o arquivo correspondente a este *path* é disponibilizado para download. O código pode ser visto na Figura 16.

```
exports.download = (req, res) => {
  const filename = req.query.filename
  const filePath = path.resolve(__dirname, filename)

  res.setHeader('Content-Disposition', `attachment; filename=${filename}`)

  res.download(filePath, (err) => {
    if (err) {
      console.error(`Erro ao fazer o download do arquivo:`, err)
      res.status(404).send(`Arquivo nao encontrado`)
  })
}
```

Figura 16 – Código implementado vulnerável a Path Transversal

### 5.2.5 CWE-400: Uncontrolled Resource Consumption

A CWE-400 agrupa as vulnerabilidades de aplicação ou sistema que não fazem uma gestão adequada de recursos de execução, tornando o sistema/aplicação vulnerável a ataques que consumam completamente seus recursos. Os recursos podem ser consumidos individualmente ou em conjunto, e variam entre consumo de CPU, memória RAM, armazenamento físico ou até mesmo a banda de internet. Muitas vezes quando um ou mais desses recursos é esgotado, o serviço para de funcionar corretamente por um período ou indefinidamente.

Entre as vulnerabilidades que causam negação de serviço estão as ReDoS (Regular expression Denial of Service). A negação de serviço por um *regex* é possível pela forma com as expressões regulares confirmam se uma string equivale a uma máquina de estados. Para entender melhor, o *regex* implementado no trabalho será explicado.

Uma validação de e-mail é normalmente feita com uma expressão regular como esta: "/[^@]*[^@]*/". Na Figura 17 é possível observar a máquina de estados para a expressão regular. Nesta existem 3 estados, A, B e $, enquanto os caracteres forem quaisquer exceto "@" a máquina permanece no estado A, quando aparecer um "@" ela vai para o estado B e recebe quantos caracteres existirem desde que não sejam "@" e então ela finaliza fazendo a transição para o estado $.

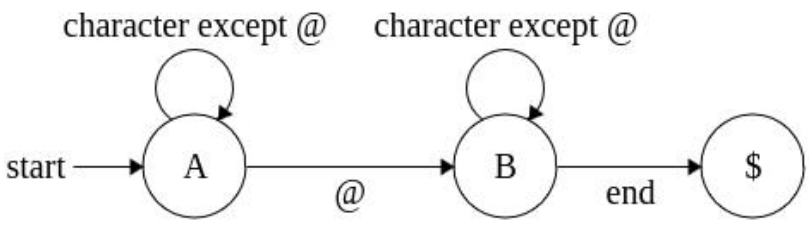


Figura 17 – Máquina de estado para a expressão /[^@]*[^@]*/

Nisto, a operação da máquina de estado é bem simples e não exige muitos recursos

computacionais, porém essa expressão não consegue englobar qualquer endereço de e-mail, uma vez que segundo a especificação formal de endereços de e-mail é possível que um endereço tenha mais de um "@" desde que este caractere esteja entre aspas [25]. Se um serviço tentar garantir que isso seja válido através de uma expressão regular, ele usará algo como '/("[^"]*"|[^@])*@[^@]*/'.

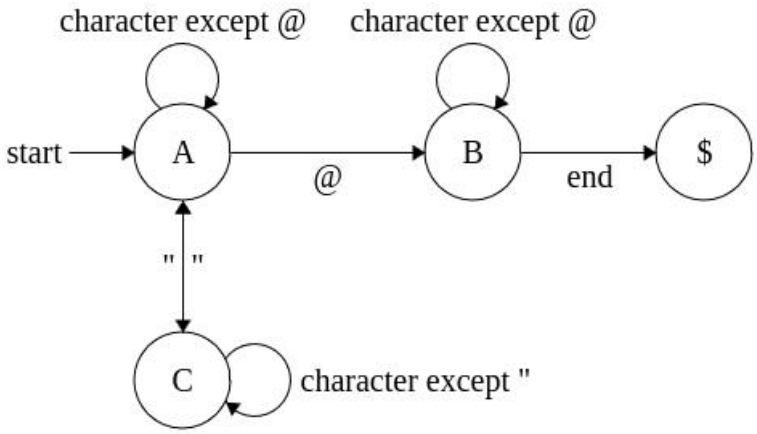


Figura 18 – Máquina de estado para a expressão /("[^"]*"|[^@])*@[^@]*/

Com essa nova expressão regular, observada na Figura 18, agora novos e-mails serão aceitos como endereços válidos, porém um não determinismo foi adicionado na expressão, criando uma árvore que possibilita que uma aspa tanto continue no estado A, quando vá para um estado C. Existem abordagens que solucionam esse problema, porém em JavaScript a máquina de estados simplesmente entende que existem vários caminhos e decide um para testar. No caso de sucesso a máquina retorna que o valor é válido, porém quando a *string* não bate com o esperado a máquina de estados voltando realizando um *backtracking*. Este *backtrack* possui um risco de escalabilidade exponencial de consumo de recursos já que no pior dos casos ele testará todas as possibilidades até retorna que não existe um correspondente para a expressão esperada.

A implementação desta vulnerabilidade foi estruturada na criação de usuário do ambiente, que ao receber nome, e-mail e senha do usuário, verifica se o e-mail é válido utilizando o regex '/("[^"]*"|[^@])*@[^@]*/' e então o cria.

### 5.2.6 CWE-89: Improper Neutralization of Special Elements

A categoria CWE-89 define as vulnerabilidades por falta de neutralização e/ou sanitização de caracteres especiais nos valores de entrada usados especificamente para SQL Injections. Pela falta de sanitização, quando os valores são adicionados a consulta SQL, um atacante consegue alterar a consulta para uma operação com comportamento divergente do esperado. A alteração pode comprometer o banco de dados, alterando valores, resgatando informações ou até deletando informações.

Para esta vulnerabilidade ser adicionado ao projeto, uma tela de login de usuário foi criada e com base no Sequelize, foi implementado uma *query* de seleção com base e-mail e senha do usuário, esta *query* pode ser vista na Figura 19.

```
db.conn.query(`SELECT name, admin FROM users WHERE email = '${email}' AND
    ↪ password = '${password}'`, {
  type: db.Sequelize.QueryTypes.SELECT
})
```

Figura 19 – Código implementado vulnerável a SQL Injection

### 5.2.7 CWE-285: Missing Authorization

Semelhante a CWE-284, mas ainda distinta, a CWE-285 é a categoria de vulnerabilidades onde a autorização aplicada em um serviço é insuficiente ou incorreta, permitindo a autorização a um usuário não autorizado. Quando um recurso fica a disposição de um usuário sem a permissão supostamente necessária para a utilização deste, esse sistema tem uma vulnerabilidade CWE-285.

Para esta vulnerabilidade, a rota “set-admin” foi criada, esta página fica escondida, mas a validação de user dela é muito simples e não exige que o usuário seja um administrador, então é possível acessar ela e utilizar os recursos a partir do momento que o usuário conseguiu um login.

```
const cookie = document.cookie
const cookieSession = 'session=active'
const hasCookie = cookie.indexOf(cookieSession) !== -1
.
.
.
if (!hasCookie) {
  return ("no-permission")
}

return(set-admin)
```

Figura 20 – Código implementado vulnerável a CWE-284

Na Figura 20 é possível ver um pseudocódigo de como a vulnerabilidade foi implementada, com base numa validação de login do serviço, ele valida se o usuário está ou não logado, porém não verifica se o usuário é administrador para dar o acesso à página que precisaria de acesso administrativo.

### 5.2.8 CWE-565: Reliance on Cookies

A categoria CWE-565 acontece quando a aplicação depende dos dados nos Cookies para a realização de operações críticas, mas a aplicação não garante que os Cookies são do usuário da sessão. Os cookies por padrão deveriam ser utilizados para armazenar preferências do usuário e alguns dados, mas utilizar eles para conceder acessos a funções torna essas funções vulneráveis uma vez que eles ficam armazenados no *client*. Com isso, o sistema se torna vulnerável a falsificações de identidade, acesso não autorizado e manipulações de sessão.

Duas páginas nessa aplicação estão vulneráveis a esta falha, a "/set-admin" e a "/admin-functions". Elas se baseiam nos cookies para renderizar páginas de perfil administrativo, a diferença é que a "/set-admin" precisa que o usuário esteja logado e a "/admin-functions" precisa que o usuário seja administrador. Essa adição é baseada na verificação de um segundo cookie boolean chamado "admin".

### 5.2.9 CWE-94: Code Injection

As Code Injections são vulnerabilidades que permitem a execução de código em um sistema e/ou aplicativo. Sistemas normalmente possuem várias entradas de dados, através das URLs, de consultadas por inputs, no upload de arquivo, entre outros. Quando uma dessas entradas não é corretamente sanitizada, o sistema pode se tornar vulnerável a uma Injection, a CWE-94 trata em específico das Injections que permitem execução arbitrária de códigos.

Dessa categoria de vulnerabilidade, a RCE (Remote Code Execution) foi escolhida para implementação. Na máquina **alpine** do back-end, foram instaladas ferramentas que permitem realizar consultas de DNS, entre elas consultas através do comando "dig". Uma página no front-end foi criada com um input que permite o usuário informar um site que deseja realizar a consulta de DNS, esse input é enviado pela URL para o back-end que insere então o nome do site num comando e devolve o resultado como um arquivo. O código pode ser visto na Figura 21.

```
const filename = req.query.filename
const command = `result=$(dig ${filename}) && echo "$result" > '/bwa-api/img/
    ↪ result.txt'`
```

Figura 21 – Código implementado vulnerável a RCE

# 6 EXPLORAÇÃO DE VULNERABILIDADES

Este capítulo abordará a forma de explorar cada um das vulnerabilidades desenvolvidas neste trabalho. O trabalho foi planejado para o aprendizado inicial de segurança ofensiva em aplicações web, por isso, evitou-se criar vulnerabilidades que necessitem de ferramentas específicas para sua exploração, dando foco em falhas que possam ser testadas manualmente ou com auxílio de simples scripts.

Como o nome da CWE de cada página está nas tabelas de vulnerabilidade, o usuário que utilizar o ambiente já deve saber o tipo de vulnerabilidade que deve procurar na página, além disso, todas as páginas possuem dicas de como explorar a falha em questão.

## 6.1 CWE-284: IMPROPER ACCESS CONTROL

Sendo a CWE-284 uma vulnerabilidade que falha em impor condições de limitação de recursos, é interessante procurar por operações na página ou informações no código do *client*. Através do código da Figura 22, é possível identificar que o ambiente possui 15 rotas, 13 acessíveis no momento e 2 inacessíveis. Dentre as acessíveis, uma página com título "admin-control" está disponível para acesso.

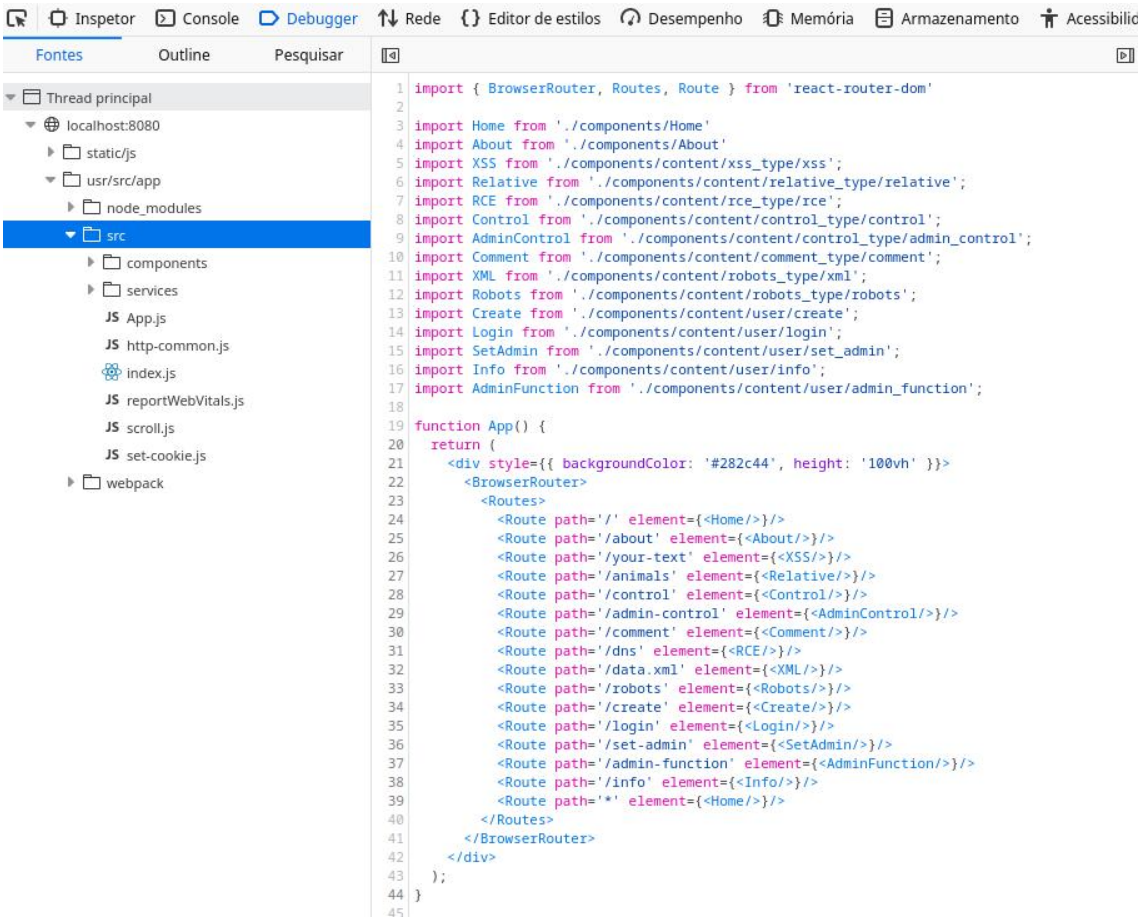


Figura 22 – Rotas encontradas no código da aplicação

Na Figura 23 então temos então sucesso ao encontrar uma página que deveria ter acesso administrativo, por informar quais contas do sistema são administrativas.

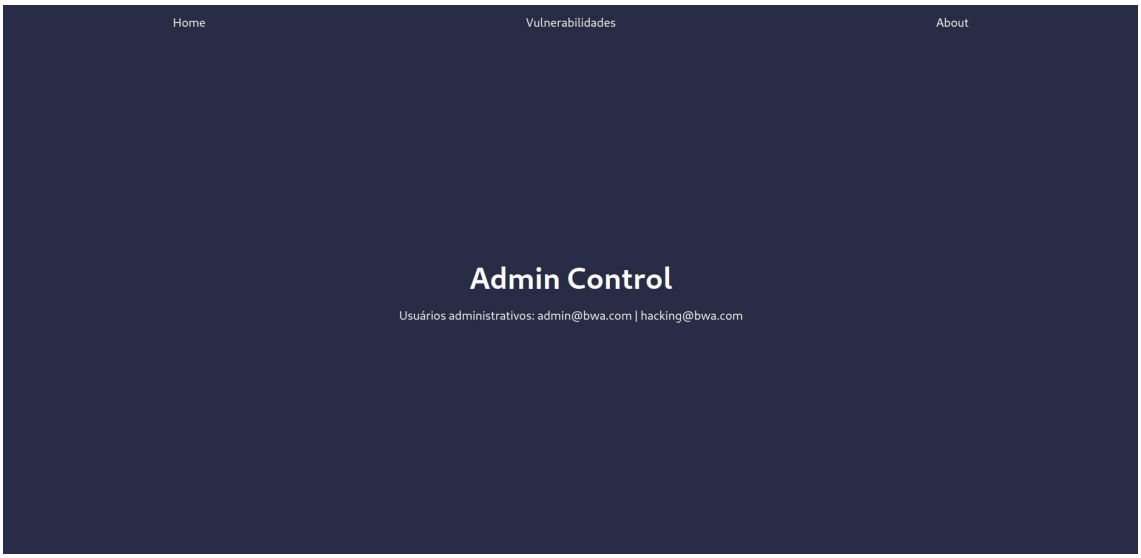


Figura 23 – Página com informações administrativas

Para mitigar essa falha, bastaria adicionar uma validação de identidade/autorização e disponibilizar a página apenas aos usuários permitidos dentro deste sistema.

## 6.2 CWE-200: EXPOSURE OF SENSITIVE INFORMATION TO AN UNAUTHORIZED ACTOR

Procurar por informações expostas é possível por buscas manuais ou por ferramentas automatizadas. Nesta busca, foi utilizada a ferramenta OWASP Zap na opção “Automated Scan” para realizar uma varredura ativa sobre o URL “http://127.0.0.1:8080/” usando “Ajax spider” e “Traditional spider”.

Na Figura 24 se identifica um comentário deixado contendo credenciais do banco de dados, comentário encontrado pelo scan do OWASP Zap. Para mitigar este tipo de falhas pode ser adicionado alguma ferramenta no pipeline do projeto que retire os comentários, agindo como uma um Lint.

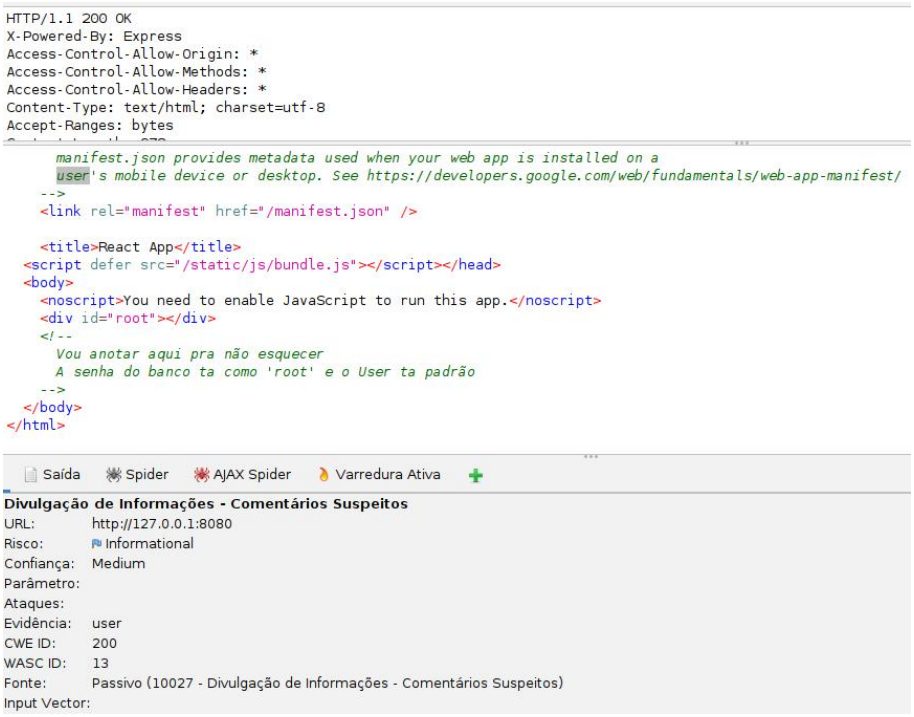


Figura 24 – Dados vazados através do HTML

Na Figura 25 se identifica ainda que o “robots.txt”, arquivo responsável por não permitir a indexação de rotas por robôs, acaba vazando uma rota, ao acessar a rota, encontra-se um XML contendo dados sensíveis de um usuário.

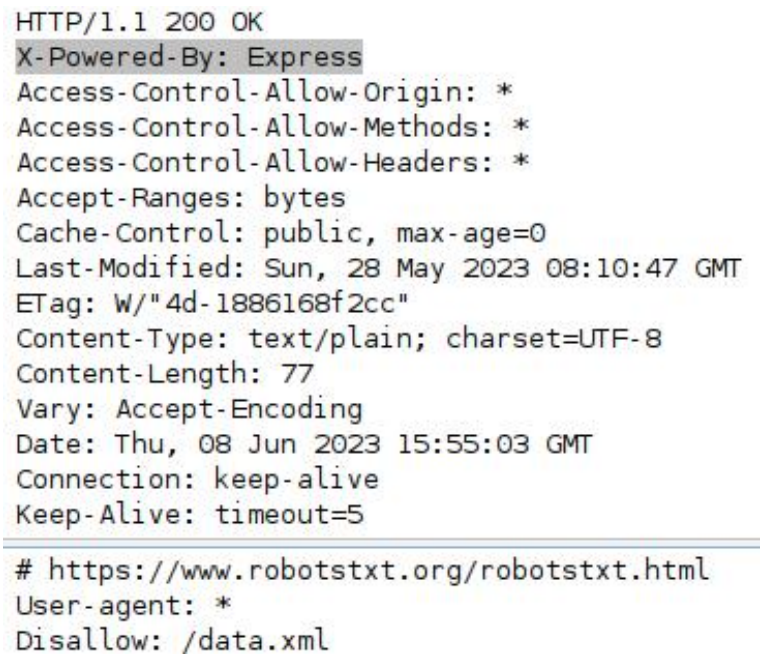


Figura 25 – Dados vazados através do robots.txt

## 6.3 CWE-79: IMPROPER NEUTRALIZATION OF INPUT DURING WEB PAGE GENERATION

A página contendo a CWE-79 contém apenas um input que renderiza o que foi escrito como um texto na tela. Observa-se que ao escrever “Teste<br>um” acontece uma quebra de linha na disposição das palavras, logo a página renderiza não apenas textos, mas também tags HTML. Tendo isso em vista, basta inserir no input um código com objetivo específico como “<img src="x"onerror="alert(’XSS Reflected exploit!’)"/>” e a página vai renderizar a tag. Neste caso, a exploração é possível até este ponto já que Reflected XSS em um cenário real passam por uma atividade de engenharia social. Normalmente a injeção de código acontece com uma intermediação de tag redirecionando dados, como “<img src="http://www.arbitraty-site.com/collect-data?data=userdata"/>”. Na Figura 26 é possível visualizar a tag sendo renderizada e o XSS sendo explorado.

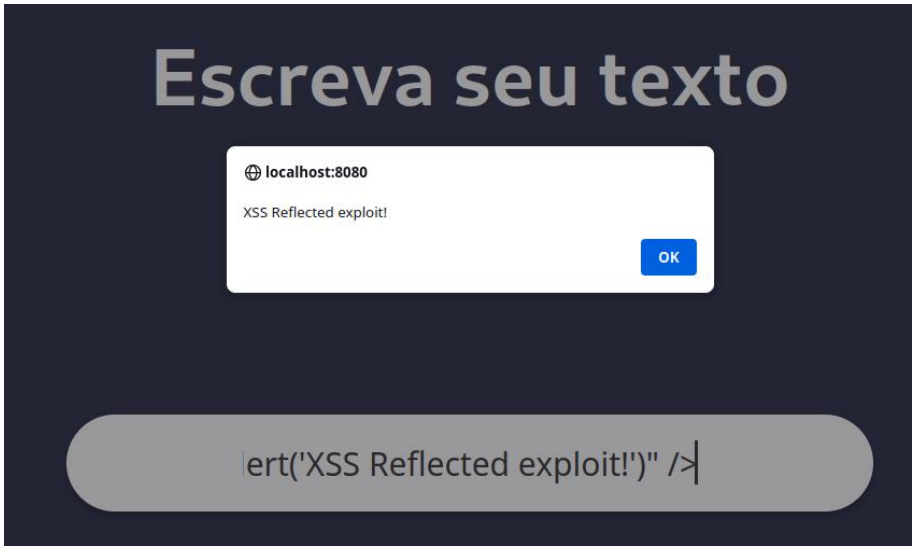


Figura 26 – Exploit de XSS no ambiente

Como comentado no capítulo anterior, o React por padrão já lida exploits básicos de XSS, então para corrigir e mitigar a CWE-79 neste ambiente basta passar a variável, que armazena o valor do input, diretamente para a *div* responsável por apresentar o texto ao invés de passar primeiramente ao *dangerouslySetInnerHTML*. Além disso, algumas bibliotecas como DOMPurify podem ser utilizadas para sanitizar as entradas do input.

## 6.4 CWE-22: IMPROPER LIMITATION OF A PATHNAME TO A RESTRICTED DIRECTORY

A página da CWE-22 conta com um input e quatro opções de arquivo para download, o usuário deve escrever o nome do arquivo que deseja fazer o download. Ao escrever “abelha” nada acontece provavelmente pelo nome precisar ser exato, já quando escreve “abelha.txt” um arquivo com a descrição do que é uma abelha é disponibilizado para download.

Agora com o intuito de descobrir se a API está rodando em uma distribuição Linux e este serviço é vulnerável a CWE-22, o path para um arquivo de configuração de distribuições Linux é tomado, neste caso o "/etc/os-release" e para continuar o teste basta ir adicionando "../" como prefixo do nome base. Após adicionar 4x o prefixo, um arquivo foi encontrado e o download foi feito, mostrando ser possível realizar um exploit de Path Transversal nesta página e a distribuição na qual a API está rodando é a "Alphine Linux v3.18".

Para ir um pouco além, sabendo que com 4 prefixos o arquivo é recuperado, a entrada "../../../../etc/passwd" é dada. Com isso um arquivo é recuperado e nele estão username, password, UID, GID, gecos, home_directory e shell dos users do servidor.

Mitigar esta vulnerabilidade pode ser feito de algumas formas, entre elas: criando uma whitelist para permitir o acesso apenas em diretórios específicos, realizando a sanitização dos dados de entrada para evitar caracteres de escape ou especiais, acessando os arquivos apenas através do caminho absoluto, limitando as permissões do usuário que executa o comando no servidor, etc.

## 6.5 CWE-400: UNCONTROLLED RESOURCE CONSUMPTION

Na página da CWE-400 existem três campos de input que podem ser testados tipos de Injection. Ferramentas de automação pode ser usadas para testar payloads de dados e comparar as respostas. Ao injetar 41 aspas no campo de e-mail, é possível notar que o tempo de resposta do servidor fica maior que o usual e a cada aspa adicional o servidor demora mais para responder. Isso se dá, pois o *backtrack* realizado pela expressão regular acaba tendo um peso de $O(2^n)$..

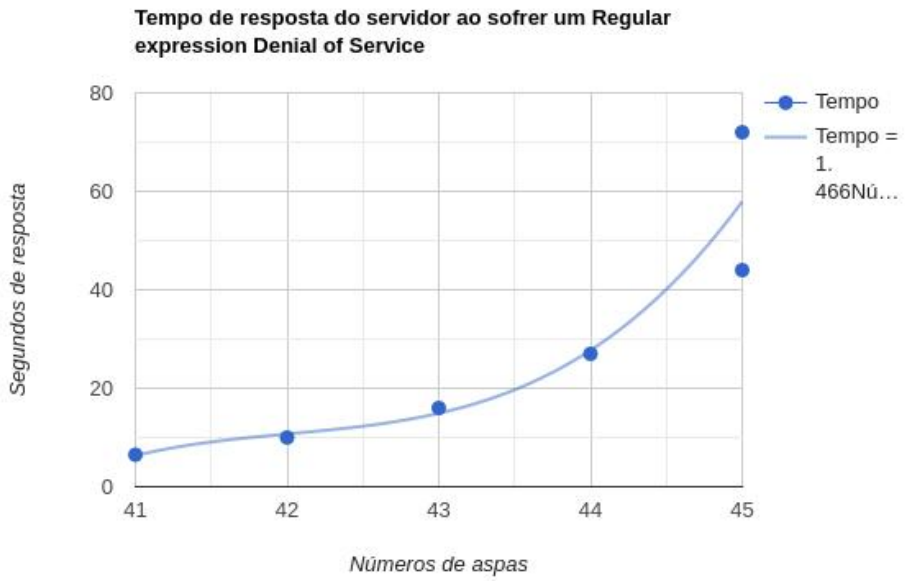


Figura 27 – Tempo de resposta do servidor contra um ataque de ReDoS

Considerando esse Big O, com 64 aspas o servidor levaria mais de 2 anos para responder. Logo para explorar esta vulnerabilidade basta realizar requisições com um número suficientemente grande de aspas para ocupar os recursos da máquina. A Figura 27 demonstra o tempo de resposta do servidor rodando em um container Docker de uma máquina com um Ryzen 5 3400g.

Esta vulnerabilidade acontece por tentar permitir que qualquer e-mail válido pela documentação seja aceito. Algumas estratégias podem ser tomadas para corrigir esta falha, entre elas, justamente desistir da ideia não convencional de aceitar este tipo de e-mail e ao invés utilizar como validador um regex mais simples, como o o /[^@]*[^@]*/.

## 6.6 CWE-89: IMPROPER NEUTRALIZATION OF SPECIAL ELEMENTS

A página destinada a CWE-89 é uma página de Login que usa e-mail e senha de usuário para efetuar o login. Nesta página existem algumas formas de realizar o Injection, a primeira seria conseguir um login baseado no id de cadastro da plataforma colocando no campo de e-mail a seguinte entrada "' OR 1=1 –%20". No servidor, a query executada seria "SELECT name, admin FROM users WHERE email = '' OR 1=1 – ' AND password = ''", esta query pegaria todos os usuários da tabela e como ela só deve retornar um usuário, ela retornará o de menor ID. O usuário com ID "1" da plataforma é "jerimundo@bwa.com.br" que não possui nenhum privilégio administrativo.

Como uma segunda etapa, pode-se juntar esta vulnerabilidade com a primeira vulnerabilidade do ambiente, nesta, os e-mails com poder administrativo ficam à disposição permitindo o atacante alterar um pouco a query e consiga o perfil desejado. Ao colocar no input a entrada "admin@bwa.com' –%20" a query retornará o perfil com o mesmo e-mail e então o atacante estará logado como admin.

## 6.7 CWE-285: MISSING AUTHORIZATION

A exploração desta página acaba sendo semelhante a da primeira vulnerabilidade, uma vez que ela apenas informa que existem outras rotas disponíveis ainda não acessadas que necessitam de um perfil de usuário administrativo. Ao percorrer as rotas, duas continuam indisponíveis, mas se o usuário fizer o login e testar novamente, uma delas estará disponível e permite tornar uma conta administrativa. Esta vulnerabilidade pode ser usada para liberar a outra página ainda indisponível.

Se for implementado um sistema de autorização nesta página, verificando a validade de um usuário pelo back-end esta vulnerabilidade pode ser atenuada.

## 6.8 CWE-565: RELIANCE ON COOKIES

A página da CWE-565 avisa que necessita de um perfil administrativo, e para explorar esta vulnerabilidade, o atacante não deve estar logado com uma conta administrativa e deve procurar outros meios de acessar esta página a partir de um usuário comum. Para explorar então, o atacante deve criar uma conta e acessar esta página.

Quando um usuário é criado, os cookies do client são setados para disponibilizar os acessos ao usuário, porém um dos cookies permite realizar a validação de perfil de usuário no *client* verificando se ele é administrador, este cookie em usuário padrão vem setado como “0”. Ao mudar o valor de cookie de “0” para “1”, a página se torna acessível.

A validação aqui está sendo feita no front-end sendo baseada em cookies, para controlar este risco, a verificação do perfil administrativo deve ser feita pelo back-end e a disponibilização dos dados e valores deve ser só enviada após essa verificação, não deixando o código previamente no front-end.

## 6.9 CWE-94: CODE INJECTION

A página da CWE-94 permite que o usuário faça uma consulta de DNS apenas informando o URL do site no input da página, então ao escrever “www.google.com” o servidor retornará um arquivo contendo informações do DNS do Google. Se neste input for colocado a entrada “www.google.com; echo "value";”, quando o arquivo com as informações do DNS retornar, ele também contará com a mensagem “value” na última linha do arquivo. Isso significa que ao escrever o comando no final do input, esse comando é executado no servidor e por isso outros comandos também podem ser executados.

Como essa vulnerabilidade permite execução remota de código, o servidor está exposto a inúmeros tipos de ataque, que permitem monitoria do tráfego, roubo de informações, infecção por outros malwares e até mesmo ataques de negação de serviço. Uma entrada como “www.google.com; rm -rf /;” deletaria todos os arquivos que o usuário padrão tiver permissão para deletar no servidor, o que resultaria numa negação de parte dos serviços do ambiente.

Para minimizar este risco várias ações devem ser tomadas. O perfil que executa o comando deve estar limitado em sua execução para não realizar operações indevidas, os valores devem ser sempre sanitizados para que nenhum valor fora do padrão entre no sistema, sanitização esta que deve ser feita principalmente no back-end. Não é seguro permitir que um usuário faça uma entrada no terminal, por isso se existir outra forma de implementar o mesmo serviço é interessante para poder remover o *eval()*. Existem algumas bibliotecas que auxiliam nisso como a *safe-eval*, mas ela por si não remove totalmente os riscos.

# 7 CONSIDERAÇÕES FINAIS

O aprendizado da área de segurança computacional pode ser desafiador, por isso ambientes para testes de vulnerabilidades são tão importantes. Com esses ambientes pode-se testar técnicas, correções, ferramentas, scripts, entre tantas outras coisas. Por este motivo este trabalho foi desenvolvido aspirando facilitar o percurso inicial na área. Além disso, ambientes em formato de imagem para Máquinas Virtuais já existem em certa quantidade, mas ambientes preparados para rodar em Docker e com isso economizar recursos são mais escassos.

O trabalho elencou o estado da arte das vulnerabilidades em aplicações web, analisando os tópicos da OWASP Top Ten:2021 e usando a mesma para selecionar vulnerabilidades pertinentes no estudo de segurança ofensiva para pessoas que estão iniciando na área da segurança da informação. As vulnerabilidades foram selecionadas com o intuito de permitir exploração ao nível de cliente, de API e de servidor, exemplos das vulnerabilidades respectivamente desses tipos no ambiente são: Reflected cross-site scripting, Regular expression Denial of Service e Remote Code Execution. Estando disponíveis em vários níveis, as falhas propositalmente colocadas no ambiente passam pelo escopo de erro humano, erro no planejamento do projeto, erro na seleção e uso das tecnologias envolvidas no projeto.

Mesmo sendo um ambiente para prática e aprendizado de técnicas ofensivas, o texto contempla formas de solucionar as vulnerabilidades possibilitando que quem o use possa alterar os códigos e realizar novamente os testes de vulnerabilidade a fim de aumentar o entendimento sobre cada falha individualmente. O código é Open Source e pode ser encontrado neste link: https://github.com/Garthu/broken-web-application.

Por ser desenvolvido em Docker, a execução do ambiente tornou-se pouco depende das configurações da máquina host, sendo facilmente executado em qualquer máquina que possua o Docker. Como visto no desenvolvimento, comparado aos ambientes em Virtual Machine citadas no trabalho, o ambiente em Docker usa menos recursos de CPU e memória, e ocupa menos recursos de armazenamento por ser relativamente leve uma vez que não necessita da virtualização de um novo kernel. Para comparação, o ambiente de teste em Docker ocupa cerca de 1Gb, por tanto é cinco vezes menor que uma ISO Ubuntu. Sua tecnologia ainda possibilita o uso do sistema com apenas dois comando “docker-compose build” e “docker-compose up”, e com isso, o tempo no uso do ambiente é gasto na exploração das vulnerabilidades e não na configuração do sistema.

Como trabalhos futuros, sugere-se:

- Expansão do ambiente adicionando novas vulnerabilidades;
- Migração do ambiente de dois contêineres para um único permitindo o registro no Docker Hub;
- Adicionar gameficação ao ambiente para auxiliar a identificação de sucesso na exploração das vulnerabilidades.

# REFERÊNCIAS


[1] Josep Albors. *Exploit: a chave para aproveitar uma vulnerabilidade*. WeLiveSecurity, jan. de 2023. URL: https://www.welivesecurity.com/br/2023/01/04/exploits/ (acesso em 31/05/2023).

[2] Rafael Oliveira de ÁVILA e Rafael Pinto da SILVA. "Brasil informacional: a segurança cibernética como desafio à segurança nacional". Em: (2013).

[3] Shivam Bathla. *A09:2021-Security Logging and Monitoring Failures*. https://medium.com/@shivam_bathla/a09-2021-security-logging-and-monitoring-failures-88c1c349f5a6, Last accessed on 2022-11-30. 2021.

[4] Doğacan Bilgili. *Using dangerouslySetInnerHTML in a React application*. LogRocket Blog, jul. de 2021. URL: https://blog.logrocket.com/using-dangerouslysetinnerhtml-in-a-react-application/ (acesso em 06/06/2023).

[5] João Henrique da Silva Carneiro. "Relação entre os motivos e o uso problemático da internet com o bem-estar : um estudo com jovens adultos". Em: (jan. de 2022). URL: http://hdl.handle.net/10400.14/37197.

[6] Theo Combe, Antony Martin e Roberto Di Pietro. "To Docker or Not to Docker: A Security Perspective". Em: *IEEE Cloud Computing* 3.5 (2016), pp. 54–62. DOI: 10.1109/MCC.2016.100.

[7] Jeremy Druin. *mutillidae*. https://github.com/webpwnized/mutillidae. 2022.

[8] Birhanu Eshete, Adolfo Villafiorita e Komminist Weldemariam. "Early Detection of Security Misconfiguration Vulnerabilities in Web Applications". Em: *2011 Sixth International Conference on Availability, Reliability and Security*. 2011, pp. 169–174. DOI: 10.1109/ARES.2011.31.

[9] g0tmi1k. *About Vulnhub*. https://www.vulnhub.com/about/, Last accessed on 2023-06-01. 2023.

[10] Gabriel Galdino. *Qual a diferença entre Vulnerabilidade, Ameaça e Risco?* DEV Community, out. de 2022. URL: https://dev.to/gabogaldino/qual-a-diferenca-entre-vulnerabilidade-ameaca-e-risco-1ijn (acesso em 31/05/2023).

[11] Pedro Leite Galvão. "Analysis and Aggregation of Vulnerability Databases with Code-Level Data". Em: (2022).

[12] M Hassan et al. "Quantitative assessment on broken access control vulnerability in web applications". Em: *International Conference on Cyber Security and Computer Science 2018*. 2018.

[13] Raphael Hiesgen et al. *The Race to the Vulnerable: Measuring the Log4j Shell Incident*. 2022. DOI: 10.48550/ARXIV.2205.02544. URL: https://arxiv.org/abs/2205.02544.

[14] Santiago Ibarra-Fiallos et al. “Effective Filter for Common Injection Attacks in Online Web Applications”. Em: *IEEE Access* 9 (2021), pp. 10378–10391. DOI: 10.1109/ACCESS.2021.3050566.

[15] Bahruz Jabiyev et al. “Preventing server-side request forgery attacks”. Em: *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. 2021, pp. 1626–1635.

[16] Lv Junyan, Xu Shiguo e Li Yijie. “Application Research of Embedded Database SQLite”. Em: *2009 International Forum on Information Technology and Applications*. Vol. 2. 2009, pp. 539–543. DOI: 10.1109/IFITA.2009.408.

[17] Eric Kahuha. *LXC vs Docker: Which Container Platform Is Right for You?* Earthly Blog, fev. de 2022. URL: https://earthly.dev/blog/lxc-vs-docker/.

[18] Li Liang et al. “Express supervision system based on NodeJS and MongoDB”. Em: *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*. 2017, pp. 607–612. DOI: 10.1109/ICIS.2017.7960064.

[19] Vikalp Nagori. “Quantum Computing posing a Challenge to Businesses”. Em: (2023).

[20] Denise Lemes Fernandes Neves et al. “A segurança da informação de encontro às conformidades da LGPD”. Em: *Revista Processando o Saber* 13 (jun. de 2021), pp. 186–198. URL: https://fatecpg.edu.br/revista/index.php/ps/article/view/171.

[21] Nurbojatmiko et al. “Security Vulnerability Analysis of the Sharia Crowdfunding Website Using OWASP-ZAP”. Em: *2022 10th International Conference on Cyber and IT Service Management (CITSM)*. 2022, pp. 1–5. DOI: 10.1109/CITSM56380.2022.9935837.

[22] OWASP. *OWASP Mutillidae II*. https://owasp.org/www-project-mutillidae-ii/, Last accessed on 2022-11-30. 2017.

[23] OWASP. *Welcome to the OWASP Top 10 - 2021*. https://owasp.org/Top10/, Last accessed on 2022-07-12. 2021.

[24] Bruno Penso. *Docker e Containers, afinal o que são?* Medium, ago. de 2020. URL: https://brunopenso.medium.com/docker-e-containers-afinal-o-que-s%C3%A3o-42a68d84aab4 (acesso em 01/06/2023).

[25] Pete Resnick. *Internet Message Format*. IETF, out. de 2008. URL: https://datatracker.ietf.org/doc/html/rfc5322#section-3.4.1 (acesso em 06/06/2023).

[26] Ugo Roveda. *REACT: O QUE É, COMO FUNCIONA E PORQUE USAR E COMO APRENDER*. https://kenzie.com.br/blog/react/, Last accessed on 2022-11-30. 2020.

[27] Sudip Sengupta. *A08:2021 – Software and Data Integrity Failures- Explained*. https://crashtest-security.com/owasp-software-data-integrity-failures/, Last accessed on 2022-11-30. 2022.

[28] Cyolo Team. *Identification And Authentication Failures And How To Prevent Them*. https://cyolo.io/blog/identification-and-authentication-failures-and-how-to-prevent-them/, Last accessed on 2022-11-30. 2022.

[29] TryHackMe. *TryHackMe*. https://tryhackme.com/r/about, Last accessed on 2023-06-01. 2023.

[30] Virgilio Viegas e Oben Kuyucu. “Security Testing and Attack Simulation Tools”. Em: *IT Security Controls: A Guide to Corporate Standards and Frameworks*. Berkeley, CA: Apress, 2022, pp. 263–290. ISBN: 978-1-4842-7799-7. DOI: 10.1007/978-1-4842-7799-7_9. URL: https://doi.org/10.1007/978-1-4842-7799-7_9.

[31] Yago Dyogennes Bezerra Vieira. “Utilização de pentest na prevenção de ataques cibernéticos às organizações”. Em: (2022). URL: http://lattes.cnpq.br/3553920177544450.

[32] R. R. Yadav, E. T. G. Sousa e G. R. A. Callou. “Performance Comparison Between Virtual Machines and Docker Containers”. Em: *IEEE Latin America Transactions* 16.8 (2018), pp. 2282–2288. DOI: 10.1109/TLA.2018.8528247.

# APÊNDICE A – CÓDIGOS DESENVOLVIDOS

## Códigos do ambiente

### bwa-api/app/controllers/rce.controller.js

```
1  const path = require('path')
2
3  exports.dns = (req, res) => {
4    const filename = req.query.filename
5    const filePath = path.resolve(__dirname, '../../img/result.txt')
6    const command = `result=$(dig ${filename}) && echo "$result" > '/
         bwa-api/img/result.txt'`
7
8    require('child_process').exec(command, (err, stdout, stderr) => {
9      res.setHeader('Content-Disposition', `attachment;
           filename=result.txt`)
10     res.setHeader('Access-Control-Allow-Headers', 'Content-Type,
           Content-Disposition')
11     res.setHeader('Access-Control-Allow-Origin', '*')
12     res.setHeader('Content-Type', 'application/octet-stream')
13
14     if (err) {
15       console.error('Erro na execucao', err)
16     }
17
18     res.download(filePath, (err) => {
19       if (err) {
20         console.error('Erro ao fazer o download do arquivo:', err)
21         res.status(404).send('Arquivo nao encontrado')
22       }
23     })
24   })
25 }
```

### bwa-api/app/controllers/relative.controller.js

```
1  const path = require('path')
2
3  exports.download = (req, res) => {
4    const filename = req.query.filename
5    const filePath = path.resolve(__dirname, filename)
6
7    res.setHeader('Content-Disposition', `attachment; filename=${
         filename}`)
8    res.setHeader('Access-Control-Allow-Headers', 'Content-Type,
         Content-Disposition')
9    res.setHeader('Access-Control-Allow-Origin', '*')
10   res.setHeader('Content-Type', 'application/octet-stream')
11
12   res.download(filePath, (err) => {
13     if (err) {
14       console.error('Erro ao fazer o download do arquivo:', err)
15       res.status(404).send('Arquivo nao encontrado')
16     }
17   })
18 }
```

**bwa-api/app/controllers/user.controller.js**

```
1  const path = require('path')
2  const { db } = require('../models/index')
3
4  exports.create = (req, res) => {
5    const name = req.query.name
6    const password = req.query.password
7    const email = req.query.email
8
9    const regex = /("[^"]*"|[^@])*@([^@]*)/
10
11   if (name == null || password == null
12       || email == null || !regex.test(email)) {
13     res.status(400).send('Dados de entrada inv lidos')
14   } else {
15     const new_user = {
16       name: name,
17       password: password,
18       email: email,
19       admin: 0
20     }
21
22     db.Users.create(new_user)
23
24     res.status(200).send('Usu rio criado')
25   }
26 }
27
28 exports.login = (req, res) => {
29   console.log(req)
30   const password = req.query.password
31   const email = req.query.email
32
33   if (password == null || email == null) {
34       res.status(400).send('Dados de entrada inv lidos')
35   } else {
36     db.conn.query(`SELECT name, admin FROM users WHERE email = '${
         email}' AND password = '${password}'`, {
37       type: db.Sequelize.QueryTypes.SELECT
38     })
39     .then((result) => {
40       if (result[0] == null) {
41         res.status(401).send('Unauthorized')
42       } else {
43         session = {
44             name: result[0].name,
45             session: 'active',
46             admin: result[0].admin
47         }
48         res.status(200).send(session)
49       }
50     })
```

```
51    }
52  }
53
54  exports.setPassword = (req, res) => {
55    const email = req.query.email
56    const password = req.query.password
57
58    if (email == null) {
59        res.status(400).send('Dados de entrada inv lidos')
60    } else {
61      db.conn.query(`UPDATE users SET password = '${password}' WHERE
            email = '${email}'`, {
62        type: db.Sequelize.QueryTypes.UPDATE
63      })
64      .then((result) => {
65        if (result[1] == 1) {
66          res.status(200).send('Senha alterada')
67        } else {
68          res.status(400).send('N  o foi poss vel atualizar a senha')
69        }
70      })
71    }
72  }
73
74  exports.setAdmin = (req, res) => {
75    const email = req.query.email
76
77    if (email == null) {
78        res.status(400).send('Dados de entrada inv lidos')
79    } else {
80      db.conn.query(`UPDATE users SET admin = 1 WHERE email = '${email
            }'`, {
81        type: db.Sequelize.QueryTypes.UPDATE
82      })
83      .then((result) => {
84        if (result[1] == 1) {
85          res.status(200).send('Usu rio agora    admin')
86        } else {
87          res.status(400).send('N  o foi poss vel atualizar o User')
88        }
89      })
90    }
91  }
```

**bwa-api/app/controllers/xss.controller.js**

```
1  exports.createTest = (req, res) => {
2      res.send({'Test':'Test'})
3  };
```

**bwa-api/app/data/users.json**

```
1  {
2      "users": [
3          {
```

```
            "name": "jerimundo",
            "password": "jerimundo",
            "email": "jerimundo@bwa.com",
            "admin": 0
        },
        {
            "name": "admin",
            "password": "admin",
            "email": "admin@bwa.com",
            "admin": 1
        },
        {
            "name": "hacking",
            "password": "hacking",
            "email": "hacking@bwa.com",
            "admin": 1
        }
    ]
}
```

**bwa-api/app/models/index.js**

```
const Sequelize = require('sequelize')

const conn = new Sequelize(process.env.DB_NAME,
  process.env.DB_USER, process.env.DB_PASSWORD, {
    host: process.env.DB_HOST,
    dialect: process.env.DIALECT,
    port: process.env.DB_PORT
  }
)

const db = {}

db.Sequelize = Sequelize
db.conn = conn

db.Users = require('./users.model.js')(conn, Sequelize)

module.exports = { db }
```

**bwa-api/app/models/migration.js**

```
module.exports = async function createData(db) {
  try {
    const fs = require('fs')

    await db.conn.sync({ force: true })

    let baseUsers = {}

    fs.readFile('/bwa-api/app/data/users.json', 'utf-8', async (err,
        data) => {
      if (err) {
        console.error('Erro: ', err)
```

```
12        }
13
14        baseUsers = JSON.parse(data)
15
16        for (const user of baseUsers.users) {
17          await db.Users.bulkCreate([
18            {
19              name: user.name,
20              password: user.password,
21              email: user.email,
22              admin: user.admin
23            }
24          ])
25        }
26      })
27    } catch (err) {
28      console.error('erro', err)
29    }
30  }
```

**bwa-api/app/routes/rce.route.js**

```
1  module.exports = app => {
2      const rce = require('../controllers/rce.controller')
3
4      var router = require("express").Router()
5
6      router.get("/", rce.dns)
7
8      app.use('/rce', router)
9    };
```

**bwa-api/app/routes/relative.route.js**

```
1  module.exports = app => {
2      const relative = require('../controllers/relative.controller')
3
4      var router = require("express").Router()
5
6      router.get("/", relative.download)
7
8      app.use('/relative', router)
9    };
```

**bwa-api/app/routes/routes.js**

```
1  module.exports = app => {
2    require('./relative.route')(app)
3    require('./rce.route')(app)
4    require('./user.route')(app)
5  };
```

**bwa-api/app/routes/user.route.js**

```
1  module.exports = app => {
2      const user = require('../controllers/user.controller')
```

```
3
4       var router = require("express").Router()
5
6       router.post("/", user.create)
7
8       router.get("/login", user.login)
9
10      router.post("/set-admin", user.setAdmin)
11
12      router.post("/set-password", user.setPassword)
13
14      app.use('/user', router)
15    };
```

**bwa-api/app/routes/xss.route.js**

```
1  module.exports = app => {
2      const xss = require('../controllers/xss.controller')
3
4      var router = require("express").Router();
5
6      // Create a new Tutorial
7      router.get("/", xss.createTest);
8
9      app.use('/xss_route', router);
10   };
```

**bwa-api/.env**

```
1  DB_HOST=localhost
2  DB_USER=root
3  DB_PASSWORD=root
4  DB_NAME=bwa-db
5  DB_PORT=3306
6  DIALECT=mysql
7
8  NODE_DOCKER_PORT=8080
```

**bwa-api/Dockerfile**

```
1  FROM alpine:latest
2
3  RUN apk add --update nodejs npm && apk add bash && apk add bind-tools
4
5  WORKDIR /bwa-api
6  COPY package.json .
7  RUN npm install
8  COPY . .
9  CMD npm start
```

**bwa-api/package.json**

```
1  {
2    "name": "broken-web-applications",
3    "version": "1.0.0",
4    "description": "",
```

```
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "dotenv": "^16.0.3",
    "express": "^4.18.2",
    "mongoose": "^7.0.4",
    "mysql2": "^3.3.2",
    "nodemon": "^2.0.22",
    "sequelize": "^6.31.1"
  }
}
```

**bwa-api/server.js**

```
require('dotenv').config()

const express = require('express')
const cors = require('cors')
const { db } = require('./app/models/index')
const migration = require('./app/models/migration')
const app = express()

var corsOptions = {
    exposedHeaders: ['Content-Disposition']
}

app.use(cors(corsOptions))

app.use(express.json())

app.get("/", (req, res) => {
    res.json({ message: "Welcome to bwa." })
})

require('./app/routes/routes')(app)

migration(db)

const PORT = process.env.NODE_DOCKER_PORT || 8080
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}.`)
})
```

**bwa-ui/public/robots.txt**

```
# https://www.robotstxt.org/robotstxt.html
User-agent: *
Disallow: /data.xml
```

**bwa-ui/src/components/content/comment_type/comment.js**

```
import React, { Component } from 'react'
import { smoothScroll}  from '../../../scroll'
import BaseHeader from '../../BaseHeader'

class Comment extends Component {
  componentDidMount() {
      let currentIndex = 0;
      var currentSection = document.querySelectorAll('.section');
      smoothScroll(currentSection, currentIndex);
  }

  render () {
    return (
        <body>
            <header id="main-header">
              <BaseHeader/>
            </header>
            <div class="section" id="start-section">
              <div id="content">
                <h2 class="h2-text">Cuidado com seus coment rios,
                    algumas coisas n o deveriam ser ditas<br/>mesmo
                    que sejam verdade</h2>
                <p></p>
              </div>
            </div>
            <div class="section">
              <div id="content">
                  <div class="center">
                    <h1>Explora  o</h1>
                    <p>
                      Coment rios pode expor informa  es
                          sens veis. Procure onde<br/>
                      usu rios normalmente n o olham
                    </p>
                  </div>
              </div>
            </div>
        </body>
    )
  }
}


export default Comment;
```

**bwa-ui/src/components/content/control_type/admin_control.js**

```
import React, { Component } from 'react'
import { smoothScroll}  from '../../../scroll'
import BaseHeader from '../../BaseHeader'

class AdminControl extends Component {
  render () {
    return (
```

```
8           <body>
9               <header id="main-header">
10                <BaseHeader/>
11              </header>
12              <div class="section" id="start-section">
13                <div id="content">
14                  <h1>Admin Control</h1>
15                  <p>Usu rios administrativos: admin@bwa.com |
                        hacking@bwa.com</p>
16                </div>
17              </div>
18          </body>
19      )
20    }
21  }
22
23
24  export default AdminControl;
```

**bwa-ui/src/components/content/control_type/control.js**

```
1   import React, { Component } from 'react'
2   import { smoothScroll}  from '../../../scroll'
3   import BaseHeader from '../../BaseHeader'
4
5   class Control extends Component {
6     componentDidMount() {
7         let currentIndex = 0;
8         var currentSection = document.querySelectorAll('.section');
9         smoothScroll(currentSection, currentIndex);
10    }
11
12    render () {
13      return (
14          <body>
15              <header id="main-header">
16                <BaseHeader/>
17              </header>
18              <div class="section" id="start-section">
19                <div id="content">
20                  <h2 class="h2-text">Se voc  colocar uma chave debaixo
                        do tapete permitir  que um ladr o encontre-a</
                        h2>
21                  <p>- Tim Cook</p>
22                </div>
23              </div>
24              <div class="section">
25                <div id="content">
26                    <div class="center">
27                      <h1>Explora  o</h1>
28                      <p>
29                        Nem todos os paths de um site ficam acess veis
                              atrav s de bot es<br/>
30                        procure por outras rotas que podem<br/>conter
```

```
                                informa  es  sens veis
31                        </p>
32                      </div>
33                    </div>
34                  </div>
35                </body>
36      )
37    }
38  }
39
40
41  export default Control;
```

**bwa-ui/src/components/content/rce_type/rce.js**

```
1   import React, { Component } from 'react'
2   import { smoothScroll}  from '../../../scroll'
3   import BaseHeader from '../../BaseHeader'
4   import RCEDataService from '../../../services/rce.services'
5
6   class RCE extends Component {
7     constructor(props) {
8       super(props)
9       this.inputReference = React.createRef();
10    }
11
12    componentDidMount() {
13      let currentIndex = 0;
14      var currentSection = document.querySelectorAll('.section');
15      smoothScroll(currentSection, currentIndex);
16    }
17
18    sendData = () => {
19      const filename = this.inputReference.current.value
20
21      RCEDataService.get(filename)
22      .then((response) => {
23        if (response.status === 200) {
24          const { data, headers } = response;
25
26          if (headers.hasOwnProperty('content-disposition')) {
27            const disposition = headers['content-disposition'];
28            const fileNameMatch = disposition.match(/filename=(?:"([^"
                ]+)"|([^;\n]+))/i);
29
30            const fileName = fileNameMatch[1] || fileNameMatch[2];
31
32            const blob = new Blob([data], { type: headers['content-type
                '] });
33
34            const url = window.URL.createObjectURL(blob);
35
36            const link = document.createElement('a');
37            link.href = url;
```

```
38            link.download = fileName;
39            link.click();
40
41            window.URL.revokeObjectURL(url);
42          } else {
43            console.error('Cabe alho "content-disposition" ausente na
                  resposta.');
44          }
45        }
46      })
47      .catch((error) => {
48        console.error('Erro ao fazer o download do arquivo:', error);
49      });
50    }
51
52    render () {
53      return (
54        <body>
55          <header id="main-header">
56            <BaseHeader/>
57          </header>
58          <div class="section" id="start-section">
59            <div id="content">
60              <h1>Coloque o link de um site para pesquisar o DNS</h1>
61              <div class='input-box'>
62                <input id='my-input' ref={this.inputReference} type='
                      text' placeholder='www.google.com'/>
63                <button class='send-button' onClick={this.sendData}>
                      Enviar</button>
64              </div>
65            </div>
66          </div>
67          <div class="section">
68            <div id="content">
69              <div class="center">
70                <h1>Explorda  o</h1>
71                <p>
72                  Muitas vezes entradas de dados em sites n o s o
                        sanitizadas<br/>
73                  tente uma Remote Code Execution
74                </p>
75              </div>
76            </div>
77          </div>
78        </body>
79      )
80    }
81  }
82
83
84  export default RCE;
```

**bwa-ui/src/components/content/relative_type/relative.js**

```
import React, { Component } from 'react'
import { smoothScroll}  from '../../../scroll'
import BaseHeader from '../../BaseHeader'
import RelativeDataService from '../../../services/relative.services'

class Relative extends Component {
  constructor(props) {
    super(props)
    this.inputReference = React.createRef();
  }

  componentDidMount() {
    let currentIndex = 0;
    var currentSection = document.querySelectorAll('.section');
    smoothScroll(currentSection, currentIndex);
  }

  sendData = () => {
    const inputValue = this.inputReference.current.value
    const filename = '../../img/' + inputValue

    RelativeDataService.get(filename)
    .then((response) => {
      if (response.status === 200) {
        const { data, headers } = response;

        if (headers.hasOwnProperty('content-disposition')) {
          const disposition = headers['content-disposition'];
          const fileNameMatch = disposition.match(/filename=(?:"([^"
              ]+)"|([^;\\n]+))/i);

          const fileName = fileNameMatch[1] || fileNameMatch[2]

          const blob = new Blob([data], { type: headers['content-type
              '] });

          const url = window.URL.createObjectURL(blob);

          const link = document.createElement('a');
          link.href = url;
          link.download = fileName;
          link.click();

          window.URL.revokeObjectURL(url);
        } else {
          console.error('Cabe alho "content-disposition" ausente na
              resposta.');
        }
      }
    })
    .catch((error) => {
      console.error('Erro ao fazer o download do arquivo:', error);
    });
  }
```

```
52
53    render () {
54      return (
55        <body>
56          <header id="main-header">
57            <BaseHeader/>
58          </header>
59          <div class="section" id="start-section">
60            <div id="content">
61              <h1>Descri  o de animais - Escolha um:</h1>
62              <p>
63                abelha.txt<br/>
64                besouro.txt<br/>
65                gafanhoto.txt<br/>
66                joaninha.txt<br/>
67              </p>
68              <div class='input-box'>
69                <input id='my-input' ref={this.inputReference} type='
                      text' placeholder='Digite aqui'/>
70                <button class='send-button' onClick={this.sendData}>
                      Enviar</button>
71              </div>
72            </div>
73          </div>
74          <div class="section">
75            <div id="content">
76              <div class="center">
77                <h1>Explora  o</h1>
78                <p>
79                  Muitas vezes entradas de dados em sites n o s o
                        sanitizadas<br/>
80                  tente um Relative Path Transversal
81                </p>
82              </div>
83            </div>
84          </div>
85        </body>
86      )
87    }
88  }
89
90
91  export default Relative;
```

**bwa-ui/src/components/content/robots_type/robots.js**

```
1  import React, { Component } from 'react'
2  import { smoothScroll}  from '../../../scroll'
3  import BaseHeader from '../../BaseHeader'
4
5  class Robots extends Component {
6    componentDidMount() {
7        let currentIndex = 0;
8        var currentSection = document.querySelectorAll('.section');
```

```
9         smoothScroll(currentSection, currentIndex);
10    }
11
12    render () {
13      return (
14          <body>
15              <header id="main-header">
16                <BaseHeader/>
17              </header>
18              <div class="section" id="start-section">
19                <div id="content">
20                  <h2 class="h2-text">Ser  mesmo que os rob s  n o
                        olham  l  ?<br/></h2>
21                  <p></p>
22                </div>
23              </div>
24              <div class="section">
25                <div id="content">
26                    <div class="center">
27                      <h1>Explora  o</h1>
28                      <p>
29                        Um arquivo    respons vel por  n  o  permitir a
                              indexa  o de rotas<br/>
30                        tente olhar  l  .
31                      </p>
32                    </div>
33                </div>
34              </div>
35          </body>
36      )
37    }
38  }
39
40
41  export default Robots;
```

**bwa-ui/src/components/content/robots_type/xml.js**

```
1   import React, { Component } from 'react'
2
3   class XML extends Component {
4     render () {
5       return (
6           <body>
7             <pre class="blank">
8               &lt;?xml version="1.0" encoding="UTF-8"?&gt;<br/>
9               &lt;dadosPessoais&gt;<br/>
10                &lt;nome&gt;Broken Aplication&lt;/nome&gt;<br/>
11                &lt;cpf&gt;123.456.789-00&lt;/cpf&gt;<br/>
12                &lt;endereco&gt;<br/>
13                  &lt;rua&gt;Rua Fict cia , 123&lt;/rua&gt;<br/>
14                  &lt;cidade&gt;Cidade  Fict  cia&lt;/cidade&gt;<br/>
15                  &lt;estado&gt;Estado  Fict  cio&lt;/estado&gt;<br/>
16                &lt;/endereco&gt;<br/>
```

```
17                &lt;telefone&gt;(00) 1234-5678&lt;/telefone&gt;<br/>
18                &lt;email&gt;broken.aplication@bwa.com&lt;/email&gt;<br/
                      >
19                &lt;cartaoCredito&gt;<br/>
20                  &lt;numero&gt;**** **** **** 1234&lt;/numero&gt;<br/>
21                  &lt;validade&gt;12/24&lt;/validade&gt;<br/>
22                  &lt;cvv&gt;123&lt;/cvv&gt;<br/>
23                &lt;/cartaoCredito&gt;<br/>
24              &lt;/dadosPessoais&gt;<br/>
25            </pre>
26          </body>
27      )
28    }
29  }
30
31
32  export default XML;
```

**bwa-ui/src/components/content/table/table_one.js**

```
1   import React from 'react'
2
3   const TableOne = () => {
4     return (
5       <div>
6         <div class="grid">
7         <a href='/control'>
8             <div class="box">
9               <img class='imgs' src="https://cdn-icons-png.flaticon.com
                    /512/1224/1224811.png" alt=" cone "/>
10              <h2>CWE-284</h2>
11              <p>Improper Access Control</p>
12            </div>
13          </a>
14          <a href='/comment'>
15            <div class="box">
16              <img class='imgs' src="https://cdn-icons-png.flaticon.com
                    /512/1224/1224811.png" alt=" cone "/>
17              <h2>CWE-200</h2>
18              <p>Exposure of Sensitive Information - 1</p>
19            </div>
20          </a>
21          <a href='/robots'>
22            <div class="box">
23              <img class='imgs' src="https://cdn-icons-png.flaticon.com
                    /512/1224/1224811.png" alt=" cone "/>
24              <h2>CWE-200</h2>
25              <p>Exposure of Sensitive Information - 2</p>
26            </div>
27          </a>
28        </div>
29        <div class="grid-two">
30        <a href='/your-text'>
31            <div class="box">
```

```
32              <img class='imgs' src="https://cdn-icons-png.flaticon.com
                    /512/1224/1224811.png" alt=" cone "/>
33              <h2>CWE-79</h2>
34              <p>Improper Neutralization</p>
35            </div>
36          </a>
37          <a href='/animals'>
38            <div class="box">
39              <img class='imgs' src="https://cdn-icons-png.flaticon.com
                    /512/1224/1224811.png" alt=" cone "/>
40              <h2>CWE-22</h2>
41              <p>Improper Limitation of a Pathname</p>
42            </div>
43          </a>
44        </div>
45      </div>
46    )
47  }
48
49  export default TableOne;
```

**bwa-ui/src/components/content/table/table_two.js**

```
1   import React from 'react'
2
3   const TableTwo = () => {
4     return (
5       <div>
6         <div class="grid">
7         <a href='/create'>
8             <div class="box">
9               <img class='imgs' src="https://cdn-icons-png.flaticon.com
                    /512/1224/1224811.png" alt=" cone "/>
10              <h2>CWE-400</h2>
11              <p>Uncontrolled Resource Consumption</p>
12            </div>
13          </a>
14          <a href='/login'>
15            <div class="box">
16              <img class='imgs' src="https://cdn-icons-png.flaticon.com
                    /512/1224/1224811.png" alt=" cone "/>
17              <h2>CWE-89</h2>
18              <p>Improper Neutralization of Special Elements</p>
19            </div>
20          </a>
21          <a href='/info'>
22            <div class="box">
23              <img class='imgs' src="https://cdn-icons-png.flaticon.com
                    /512/1224/1224811.png" alt=" cone "/>
24              <h2>CWE-285</h2>
25              <p>Missing Authorization</p>
26            </div>
27          </a>
28        </div>
```

```
29        <div class="grid-two">
30        <a href='/admin-function'>
31            <div class="box">
32              <img class='imgs' src="https://cdn-icons-png.flaticon.com
                    /512/1224/1224811.png" alt=" cone "/>
33              <h2>CWE-565</h2>
34              <p>Reliance on Cookies</p>
35            </div>
36          </a>
37          <a href='/dns'>
38            <div class="box">
39              <img class='imgs' src="https://cdn-icons-png.flaticon.com
                    /512/1224/1224811.png" alt=" cone "/>
40              <h2>CWE-94</h2>
41              <p>Code Injection</p>
42            </div>
43          </a>
44        </div>
45      </div>
46    )
47  }
48
49  export default TableTwo;
```

**bwa-ui/src/components/content/user/admin_function.js**

```
1   import React, { Component } from 'react'
2   import { smoothScroll}  from '../../../scroll'
3   import BaseHeader from '../../BaseHeader'
4   import Users from '../../../services/user.services'
5
6   class AdminFunction extends Component {
7     constructor(props) {
8       super(props)
9       this.emailReference = React.createRef()
10      this.passwordReference = React.createRef()
11      this.state = {
12        showSucess: false,
13        showFailure: false,
14        hasCookie: false,
15        isAdmin: false
16      }
17    }
18
19    componentDidMount() {
20      let currentIndex = 0
21      var currentSection = document.querySelectorAll('.section')
22      smoothScroll(currentSection, currentIndex)
23
24      const cookie = document.cookie
25      const cookieSession = 'session=active'
26      const adminCookieSession = 'admin=1'
27      const hasCookie = cookie.indexOf(cookieSession) !== -1
28      const isAdmin = cookie.indexOf(adminCookieSession) !== -1
```

```
29
30      this.setState({ hasCookie })
31      this.setState({ isAdmin })
32    }
33
34    sendData = () => {
35      const emailReference = this.emailReference.current.value
36      const passwordReference = this.passwordReference.current.value
37
38      const data = {
39        email: emailReference,
40        password: passwordReference
41      }
42
43      Users.setPassword(data)
44      .then((response) => {
45        if (response.status === 200) {
46          this.setState({ showSucess: true })
47          this.setState({ showFailure: false })
48        }
49      })
50      .catch((err) => {
51        console.error('Erro: ', err)
52        this.setState({ showSucess: false })
53        this.setState({ showFailure: true })
54      })
55    }
56
57    render () {
58      const { showSucess, showFailure, isAdmin, hasCookie } = this.state
59
60      if (!hasCookie || !isAdmin) {
61        return (
62          <body>
63            <header id="main-header">
64              <BaseHeader/>
65            </header>
66            <div class="section" id="start-section">
67              <div id="content">
68                <h1>P gina de acesso administrativo</h1>
69              </div>
70            </div>
71            <div class="section">
72              <div id="content">
73                <div class="center">
74                  <h1>Explora  o</h1>
75                  <p>
76                    Talvez o site n o verifique bem o tipo de usu rio
77                  </p>
78                </div>
79              </div>
80            </div>
81          </body>
82        )
```

```
83      }
84
85      return (
86        <body>
87          <header id="main-header">
88            <BaseHeader/>
89          </header>
90          <div class="section" id="start-section">
91            <div id="content">
92              <h1>Definir senha de usu rio</h1>
93              <div class='input-box'>
94                <input class="input-login" ref={this.emailReference}
                      type='text' placeholder='E-mail'/>
95              </div>
96              <div class='input-box'>
97                <input ref={this.passwordReference} type='text'
                      placeholder='Senha'/>
98                <button class='send-button' onClick={this.sendData}>
                      Enviar</button>
99              </div>
100             {showSucess && <p class="p-success">Senha atualizada</p>}
101             {showFailure && <p class="p-failure">Usu rio  n  o
                    encontrado</p>}
102           </div>
103         </div>
104       </body>
105     )
106   }
107 }
108
109
110 export default AdminFunction;
```

**bwa-ui/src/components/content/user/create.js**

```
1  import React, { Component } from 'react'
2  import { smoothScroll}  from '../../../scroll'
3  import BaseHeader from '../../BaseHeader'
4  import Users from '../../../services/user.services'
5
6  class Create extends Component {
7    constructor(props) {
8      super(props)
9      this.nameReference = React.createRef();
10     this.emailReference = React.createRef();
11     this.passwordReference = React.createRef();
12     this.state = {
13         showSucess: false,
14         showFailure: false
15     }
16   }
17
18   componentDidMount() {
19     let currentIndex = 0;
```

```
20      var currentSection = document.querySelectorAll('.section');
21      smoothScroll(currentSection, currentIndex);
22    }
23
24    sendData = () => {
25      const nameReference = this.nameReference.current.value
26      const emailReference = this.emailReference.current.value
27      const passwordReference = this.passwordReference.current.value
28
29      const data = {
30          name: nameReference,
31          email: emailReference,
32          password: passwordReference
33      }
34
35      Users.create(data)
36      .then((response) => {
37        if (response.status === 200) {
38          this.setState({ showSucess: true })
39          this.setState({ showFailure: false })
40        }
41      })
42      .catch((err) => {
43        console.error('Erro: ', err)
44        this.setState({ showFailure: true })
45        this.setState({ showSucess: false })
46      })
47    }
48
49    render () {
50      const { showSucess, showFailure } = this.state
51
52      return (
53        <body>
54          <header id="main-header">
55            <BaseHeader/>
56          </header>
57          <div class="section" id="start-section">
58            <div id="content">
59              <h1>Create a User</h1>
60              <div class='input-box'>
61                <input class="input-login" ref={this.nameReference}
                      type='text' placeholder='Nome'/>
62              </div>
63              <div class='input-box'>
64                <input class="input-login" ref={this.emailReference}
                      type='text' placeholder='E-mail'/>
65              </div>
66              <div class='input-box'>
67                <input ref={this.passwordReference} type='text'
                      placeholder='Senha'/>
68                <button class='send-button' onClick={this.sendData}>
                      Enviar</button>
69              </div>
```

```
70              {showSucess && <p class="p-success">Usu rio criado</p>}
71              {showFailure && <p class="p-failure">Entrada Inv lida</p>
                    }
72            </div>
73          </div>
74          <div class="section">
75            <div id="content">
76              <div class="center">
77                <h1>Explora  o</h1>
78                <p>
79                  Muitas vezes entradas de dados em sites n o s o
                        sanitizadas<br/>
80                  tente uma ReDoS
81                </p>
82              </div>
83            </div>
84          </div>
85        </body>
86      )
87    }
88  }
89
90
91  export default Create;
```

**bwa-ui/src/components/content/user/info.js**

```
1   import React, { Component } from 'react'
2   import BaseHeader from '../../BaseHeader'
3
4   class Info extends Component {
5     render () {
6           return (
7         <body>
8           <header id="main-header">
9             <BaseHeader/>
10          </header>
11          <div class="section" id="start-section">
12            <div id="content">
13              <h1>Dica: Refa a os desafios para descobrir as novas
                      func es do seu usu rio</h1>
14            </div>
15          </div>
16        </body>
17      )
18    }
19  }
20
21
22  export default Info;
```

**bwa-ui/src/components/content/user/login.js**

```
1   import React, { Component } from 'react'
2   import { smoothScroll}  from '../../../scroll'
```

```
3  import BaseHeader from '../../BaseHeader'
4  import Users from '../../../services/user.services'
5  import setCookie from '../../../set-cookie'
6
7  class Login extends Component {
8    constructor(props) {
9      super(props)
10     this.nameReference = React.createRef();
11     this.emailReference = React.createRef();
12     this.passwordReference = React.createRef();
13     this.state = {
14         showSucess: false,
15         showFailure: false,
16         name: ''
17     }
18   }
19
20   componentDidMount() {
21     let currentIndex = 0;
22     var currentSection = document.querySelectorAll('.section');
23     smoothScroll(currentSection, currentIndex);
24   }
25
26   sendData = () => {
27     const emailReference = this.emailReference.current.value
28     const passwordReference = this.passwordReference.current.value
29
30     const data = {
31       email: emailReference,
32       password: passwordReference
33     }
34
35     Users.login(data)
36     .then((response) => {
37       if (response.status === 200) {
38         this.setState({ showSucess: true })
39         this.setState({ showFailure: false })
40         this.setState({ name: response.data.name })
41
42         setCookie('name', response.data.name, 7)
43         setCookie('session', response.data.session, 7)
44         setCookie('admin', response.data.admin, 7)
45       }
46     })
47     .catch((err) => {
48       console.error('Erro: ', err)
49       this.setState({ showSucess: false })
50       this.setState({ showFailure: true })
51     })
52   }
53
54   render () {
55     const { showSucess, showFailure, name } = this.state
56
```

```
57      return (
58        <body>
59          <header id="main-header">
60            <BaseHeader/>
61          </header>
62          <div class="section" id="start-section">
63            <div id="content">
64              <h1>Login Page</h1>
65              <div class='input-box'>
66                <input class="input-login" ref={this.emailReference}
                      type='text' placeholder='E-mail'/>
67              </div>
68              <div class='input-box'>
69                <input ref={this.passwordReference} type='text'
                      placeholder='Senha'/>
70                <button class='send-button' onClick={this.sendData}>
                      Enviar</button>
71              </div>
72              {showSucess && <p class="p-success">Usu rio logado: {name
                    }</p>}
73              {showFailure && <p class="p-failure">Login inv lido</p>}
74            </div>
75          </div>
76          <div class="section">
77            <div id="content">
78              <div class="center">
79                <h1>Explora  o</h1>
80                <p>
81                  Muitas vezes entradas de dados em sites n o s o
                        sanitizadas<br/>
82                  tente uma SQL Injection
83                </p>
84              </div>
85            </div>
86          </div>
87        </body>
88      )
89    }
90  }
91
92
93  export default Login;
```

**bwa-ui/src/components/content/user/set_admin.js**

```
1  import React, { Component } from 'react'
2  import { smoothScroll}  from '../../../scroll'
3  import BaseHeader from '../../BaseHeader'
4  import Users from '../../../services/user.services'
5
6  class SetAdmin extends Component {
7    constructor(props) {
8      super(props)
9      this.emailReference = React.createRef();
```

```
10      this.state = {
11          showSucess: false,
12          showFailure: false,
13          hasCookie: false
14      }
15    }
16
17    componentDidMount() {
18      let currentIndex = 0
19      var currentSection = document.querySelectorAll('.section')
20      smoothScroll(currentSection, currentIndex)
21
22      const cookie = document.cookie
23      const cookieSession = 'session=active'
24      const hasCookie = cookie.indexOf(cookieSession) !== -1
25
26      this.setState({ hasCookie })
27    }
28
29    sendData = () => {
30      Users.setAdmin(this.emailReference.current.value)
31      .then((response) => {
32        if (response.status === 200) {
33          this.setState({ showSucess: true })
34          this.setState({ showFailure: false })
35        }
36      })
37      .catch((err) => {
38        console.error('Erro: ', err)
39        this.setState({ showSucess: false })
40        this.setState({ showFailure: true })
41      })
42    }
43
44    render () {
45      const { showSucess, showFailure, hasCookie } = this.state
46
47      if (!hasCookie) {
48        return (
49          <body>
50            <header id="main-header">
51              <BaseHeader/>
52            </header>
53            <div class="section" id="start-section">
54              <div id="content">
55                <h1>P gina de acesso administrativo</h1>
56              </div>
57            </div>
58            <div class="section">
59              <div id="content">
60                <div class="center">
61                  <h1>Explora  o</h1>
62                  <p>
63                    Talvez o site n o verifique bem o tipo de usu rio
```

```
64                  </p>
65                </div>
66              </div>
67            </div>
68          </body>
69        )
70      }
71
72      return (
73        <body>
74          <header id="main-header">
75            <BaseHeader/>
76          </header>
77          <div class="section" id="start-section">
78            <div id="content">
79              <h1>Definir usu rio como Admin</h1>
80              <div class='input-box'>
81                <input ref={this.emailReference} type='text'
                      placeholder='E-mail'/>
82                <button class='send-button' onClick={this.sendData}>
                      Enviar</button>
83              </div>
84              {showSucess && <p class="p-success">Usu rio agora
                    Admin</p>}
85              {showFailure && <p class="p-failure">Usu rio  n  o
                    encontrado</p>}
86            </div>
87          </div>
88        </body>
89      )
90    }
91  }
92
93
94  export default SetAdmin;
```

**bwa-ui/src/components/content/xss_type/xss.css**

```
1   .send-button {
2       color: black;
3     }
4
5   div {
6     color: white;
7   }
8
9   .input-box-xss {
10    display: flex;
11    align-items: center;
12    justify-content: center;
13    width: 500px;
14    height: 60px;
15    border-radius: 30px;
16    background-color: #fff;
```

```
17    box-shadow: 0px 2px 4px rgba(0, 0, 0, 0.3);
18    margin: 20px auto;
19  }
20
21  .input-box-xss input {
22    border: none;
23    outline: none;
24    text-align: center;
25    height: 100%;
26    padding: 30 20px;
27    font-size: 20px;
28    color: #333;
29    background-color: transparent;
30  }
```

**bwa-ui/src/components/content/xss_type/xss.js**

```
1  import React, { Component } from 'react'
2  import { smoothScroll}  from '../../../scroll'
3  import BaseHeader from '../../BaseHeader'
4  import './xss.css'
5
6  class XSS extends Component {
7    constructor(props) {
8      super(props);
9      this.state = {
10       aboutUserText: '',
11       isRendered: false
12     };
13   }
14
15   handleChange = (event) => {
16     this.setState({ aboutUserText: event.target.value });
17   };
18
19   handleRender = () => {
20     this.setState({ isRendered: true });
21   };
22
23   componentDidMount() {
24     let currentIndex = 0;
25     var currentSection = document.querySelectorAll('.section');
26     smoothScroll(currentSection, currentIndex);
27   }
28   render () {
29     const { aboutUserText, isRendered } = this.state;
30
31     return (
32       <body>
33         <header id="main-header">
34           <BaseHeader/>
35         </header>
36         <div class="section" id="start-section">
37           <div id="content">
```

```
38              <h1>Escreva seu texto</h1>
39              <div dangerouslySetInnerHTML={{"__html": aboutUserText}} /
                    >
40              <div class='input-box-xss'>
41                <input type='text' value={aboutUserText} onChange={
                      this.handleChange} placeholder='Seu texto'/>
42                {isRendered && (
43                  <div class="blank" dangerouslySetInnerHTML={{ __html:
                        aboutUserText }} />
44                )}
45              </div>
46            </div>
47          </div>
48          <div class="section">
49            <div id="content">
50              <div class="center">
51                <h1>Explora  o</h1>
52                <p>
53                  Muitas vezes entradas de dados em sites n o s o
                        sanitizadas<br/>
54                  tente um XSS Injection
55                </p>
56              </div>
57            </div>
58          </div>
59        </body>
60      )
61    }
62  }
63
64
65  export default XSS;
```

**bwa-ui/src/components/content/XSSContent.js**

```
1  import React from 'react'
2
3  const XSSContent = () => {
4      return (
5          <div class="grid">
6          <div class="box">
7            <img class='imgs' src="https://cdn-icons-png.flaticon.com
                 /512/1224/1224811.png" alt=" cone "/>
8            <h2>T tulo 1</h2>
9            <p>Descri  o curta 1</p>
10         </div>
11         <div class="box">
12           <img class='imgs' src="https://cdn-icons-png.flaticon.com
                 /512/1224/1224811.png" alt=" cone "/>
13           <h2>T tulo 2</h2>
14           <p>Descri  o curta 2</p>
15         </div>
16         <div class="box">
17           <img class='imgs' src="https://cdn-icons-png.flaticon.com
```

```
                /512/1224/1224811.png" alt=" cone "/>
18            <h2>T tulo 3</h2>
19            <p>Descri  o curta 3</p>
20          </div>
21          <div class="box">
22            <img class='imgs' src="https://cdn-icons-png.flaticon.com
                /512/1224/1224811.png" alt=" cone "/>
23            <h2>T tulo 4</h2>
24            <p>Descri  o curta 4</p>
25          </div>
26          <div class="/relative">
27            <img class='imgs' src="https://cdn-icons-png.flaticon.com
                /512/1224/1224811.png" alt=" cone "/>
28            <h2>T tulo 5</h2>
29            <p>Descri  o curta 5</p>
30          </div>
31          <a href='/xss'>
32            <div class="box">
33              <img class='imgs' src="https://cdn-icons-png.flaticon.com
                  /512/1224/1224811.png" alt=" cone "/>
34              <h2>T tulo 6</h2>
35              <p>Descri  o curta 6</p>
36            </div>
37          </a>
38        </div>
39      )
40  }
41
42  export default XSSContent;
```

### bwa-ui/src/components/About.js

```
1   import React from 'react'
2   import BaseHeader from './BaseHeader';
3
4   const About = () => {
5     return (
6       <body>
7         <header>
8           <BaseHeader/>
9         </header>
10        <div class="container">
11          <div class='section'>
12          <main>
13            <div class="center">
14              <h1>Broken Web Applications</h1>
15              <p>Este projeto foi desenvolvido por: Samuel Cardoso</p>
16            </div>
17          </main>
18          </div>
19        </div>
20      </body>
21    )
22  }
```

```
23
24 export default About;
```

### bwa-ui/src/components/BaseHeader.js

```
1  import React from 'react'
2
3  const BaseHeader = () => {
4    return (
5      <nav>
6        <ul>
7          <li><a href="/">Home</a></li>
8          <li><a href="/">Vulnerabilidades</a></li>
9          <li><a href="/about">About</a></li>
10       </ul>
11     </nav>
12   )
13 }
14
15 export default BaseHeader;
```

### bwa-ui/src/components/Body.css

```
1  * {
2      margin: 0;
3      padding: 0;
4      box-sizing: border-box;
5    }
6
7    html, body {
8      height: 100%;
9      background-color: #282c44;
10   }
11
12   header {
13     background-color: #282c44;
14     color: #fff;
15     padding: 20px;
16     height: 80px;
17   }
18
19   nav ul {
20     list-style: none;
21     display: flex;
22     justify-content: space-around;
23   }
24
25   nav a {
26     color: #fff;
27     text-decoration: none;
28     font-size: 20px;
29   }
30
31   .center {
32     margin: 30vh;
```

```
33      display: flex;
34      flex-direction: column;
35      justify-content: center;
36      align-items: center;
37      text-align: center;
38    }
39
40    .blank {
41      color:#fff;
42      font-size: 20px;
43      text-align: center;
44      padding: 10%;
45    }
46
47    h1 {
48      font-size: 50px;
49      text-align: center;
50      margin-bottom: 20px;
51      color: #fff;
52    }
53
54    .h2-text {
55      font-size: 30px;
56      text-align: center;
57      margin-bottom: 20px;
58      color: #ffffff;
59    }
60
61    p {
62      font-size: 20px;
63      margin-bottom: 50px;
64      color: #fff;
65    }
66
67    .buttons {
68      display: flex;
69    }
70
71    button {
72      border-radius: 30px;
73      padding: 10px 20px;
74      margin: 0 10px;
75      font-size: 20px;
76      color: #fff;
77      text-align: center;
78      text-decoration: none;
79      display: inline-block;
80      transition-duration: 0.4s;
81      cursor: pointer;
82      min-width: 200px;
83    }
84
85    .blue {
86      background-color: #2196F3;
```

```
87      border: 1px solid transparent;
88    }
89
90    .transparent {
91      background-color: transparent;
92      border: 1px solid #fff;
93    }
94
95    .input-box {
96      display: flex;
97      align-items: center;
98      justify-content: center;
99      width: 500px;
100     height: 60px;
101     border-radius: 30px;
102     background-color: #fff;
103     box-shadow: 0px 2px 4px rgba(0, 0, 0, 0.3);
104     margin: 20px auto;
105   }
106
107   .input-box input {
108     border: none;
109     outline: none;
110     width: calc(100% - 220px);
111     height: 100%;
112     padding: 0 20px;
113     font-size: 20px;
114     color: #333;
115     background-color: transparent;
116   }
117
118   .input-box input.input-login {
119     border: none;
120     outline: none;
121     width: calc(100% - 0px);
122     height: 100%;
123     padding: 0 20px;
124     font-size: 20px;
125     background-color: transparent;
126   }
127
128   .input-box button {
129     border: none;
130     outline: none;
131     border-radius: 30px;
132     padding: 10px 20px;
133     margin: 0 10px;
134     font-size: 20px;
135     color: #282c44;
136     text-align: center;
137     text-decoration: none;
138     display: inline-block;
139     transition-duration: 0.4s;
140     cursor: pointer;
```

```
141       min-width: 200px;
142       background-color: white;
143       border: 1px solid #282c44;
144     }
145
146     .input-box button:hover {
147       background-color: #282c44;
148       color: white;
149     }
150
151     .p-success {
152       color: #2196F3;
153     }
154
155     .p-failure {
156       color: rgb(181, 55, 55);
157     }
```

**bwa-ui/src/components/Body.js**

```
1  import React from 'react'
2  import './Body.css'
3  import VulnOpt from './VulnOpt'
4
5  const Body = () => {
6    function handleClick() {
7      const secao2 = document.querySelector('#start-section');
8      secao2.scrollIntoView({ behavior: 'smooth' });
9    }
10
11   return (
12     <body>
13       <div class="container">
14         <div class='section' id="home-section">
15         <main>
16           <div class="center">
17             <h1>Broken Web Applications</h1>
18             <p>Para acessar as aplica  es</p>
19             <div class="buttons">
20               <button onClick={handleClick} class="blue">Iniciar</
                     button>
21               <button class="transparent">Github</button>
22             </div>
23           </div>
24         </main>
25         </div>
26         <VulnOpt/>
27       </div>
28     </body>
29   )
30 }
31
32 export default Body;
```

**bwa-ui/src/components/Header.js**

```
1  import React from 'react'
2
3  const Header = () => {
4    const handleLinkClick = (event) => {
5      event.preventDefault();
6
7      const targetId = event.currentTarget.getAttribute('href');
8      const targetElement = document.querySelector(targetId);
9
10     targetElement.scrollIntoView({ behavior: 'smooth' });
11   }
12
13
14   return (
15     <nav>
16       <ul>
17         <li><a href="#home-section" onClick={handleLinkClick}>Home</a>
               </li>
18         <li><a href="#start-section" onClick={handleLinkClick}>
               Vulnerabilidades</a></li>
19         <li><a href="/about">About</a></li>
20       </ul>
21     </nav>
22   )
23 }
24
25 export default Header;
```

**bwa-ui/src/components/Home.js**

```
1  import React from 'react'
2  import Body from './Body'
3
4  const Home = () => {
5    return (
6      <Body/>
7    )
8  }
9
10 export default Home;
```

**bwa-ui/src/components/Nav.js**

```
1  import React from 'react'
2  import { Link } from 'react-router-dom'
3
4  const Nav=()=>{
5    return (
6      <div>
7        <ul className='nav-ul'>
8          <li><Link to='/'>Produtos</Link></li>
9          <li><Link to='/add'>Produtos</Link></li>
10       </ul>
```

```
11      </div>
12    )
13  }
14
15  export default Nav;
```

### bwa-ui/src/components/VulnOpt.css

```
1   html,
2   body {
3       height: 100%;
4   }
5
6   .section {
7       height: 100vh;
8       padding-top: 80px;
9   }
10
11  .section.active {
12      display: block;
13  }
14
15  a {
16      text-decoration: none;
17  }
18
19  #main-header {
20      position: fixed;
21      top: 0;
22      left: 0;
23      right: 0;
24      height: 80px;
25      z-index: 1000;
26    }
27
28  .imgs {
29      height: 120px;
30  }
31
32  #content {
33    position: relative;
34    justify-content: center;
35    align-items: center;
36    top: 50%;
37    left: 50%;
38    display: table;
39    transform: translate(-50%, -50%);
40    text-align: center;
41  }
42
43
44  .grid {
45    display: grid;
46    grid-template-columns: repeat(3, 1fr);
```

```
47    grid-gap: 15px;
48    margin-top: 10px;
49  }
50
51  .grid-two {
52    justify-content: center;
53    display: flex;
54    grid-template-columns: repeat(2, 0fr);
55    grid-gap: 15px;
56    margin-top: 20px;
57  }
58
59  .box {
60    background-color: transparent;
61    border-radius: 2px;
62    border: 2px solid #737373;
63    text-align: left;
64    padding: 35px 20px 35px 20px;
65    width: 328px;
66    height: 139px;
67    }
68
69  .box img {
70    float: left;
71    padding: 35px 30px 35px 20px;
72    margin-top: -30px;
73  }
74
75  .box h2 {
76    color: white;
77    font-size: 1.2em;
78    margin: 0 0 10px;
79  }
80
81  .box p {
82    font-size: 1em;
83    margin: 0;
84  }
```

**bwa-ui/src/components/VulnOpt.js**

```
1  import React, { Component } from 'react'
2  import { smoothScroll}  from '../scroll'
3  import TableOne from './content/table/table_one';
4  import TableTwo from './content/table/table_two';
5  import Header from './Header';
6  import './VulnOpt.css'
7
8  class VulnOpt extends Component {
9    componentDidMount() {
10     let currentIndex = 0;
11     var currentSection = document.querySelectorAll('.section');
12     smoothScroll(currentSection, currentIndex);
13   }
```

```
14
15    render () {
16      return (
17        <body>
18          <header id="main-header">
19            <Header/>
20          </header>
21          <div class="section" id="start-section">
22            <div id="content">
23            <h1>Genesis Exploits</h1>
24              <p></p>
25              <TableOne/>
26            </div>
27          </div>
28          <div class="section">
29            <div id="content">
30              <h1>Apocalypse Exploits</h1>
31              <p></p>
32              <TableTwo/>
33            </div>
34          </div>
35        </body>
36      )
37    }
38  }
39
40
41  export default VulnOpt;
```

**bwa-ui/src/services/rce.services.js**

```
1  import http from '../http-common'
2
3  class RCEDataService {
4    get(filepath) {
5      return http.get(`/rce?filename=${filepath}`)
6    }
7  }
8
9  export default new RCEDataService();
```

**bwa-ui/src/services/relative.services.js**

```
1  import http from '../http-common'
2
3  class RelativeDataService {
4    get(filepath) {
5      return http.get(`/relative?filename=${filepath}`)
6    }
7  }
8
9  export default new RelativeDataService();
```

**bwa-ui/src/services/user.services.js**

```
import http from '../http-common'

class Users {
  create(data) {
    const name = data.name
    const password = data.password
    const email = data.email

    return http.post(`/user?name=${name}&password=${password}&email=${email}`)
  }

  login(data) {
    const password = data.password
    const email = data.email

    return http.get(`/user/login?email=${email}&password=${password}`)
  }

  setAdmin(email) {
    return http.post(`/user/set-admin?email=${email}`)
  }

  setPassword(data) {
    const password = data.password
    const email = data.email

    return http.post(`/user/set-password?email=${email}&password=${password}`)
  }
}

export default new Users();
```

**bwa-ui/src/App.js**

```
import { BrowserRouter, Routes, Route } from 'react-router-dom'

import Home from './components/Home'
import About from './components/About'
import XSS from './components/content/xss_type/xss';
import Relative from './components/content/relative_type/relative';
import RCE from './components/content/rce_type/rce';
import Control from './components/content/control_type/control';
import AdminControl from './components/content/control_type/admin_control';
import Comment from './components/content/comment_type/comment';
import XML from './components/content/robots_type/xml';
import Robots from './components/content/robots_type/robots';
import Create from './components/content/user/create';
import Login from './components/content/user/login';
import SetAdmin from './components/content/user/set_admin';
import Info from './components/content/user/info';
import AdminFunction from './components/content/user/admin_function';
```

```
18
19 function App() {
20   return (
21     <div style={{ backgroundColor: '#282c44', height: '100vh' }}>
22       <BrowserRouter>
23         <Routes>
24           <Route path='/' element={<Home/>}/>
25           <Route path='/about' element={<About/>}/>
26           <Route path='/your-text' element={<XSS/>}/>
27           <Route path='/animals' element={<Relative/>}/>
28           <Route path='/control' element={<Control/>}/>
29           <Route path='/admin-control' element={<AdminControl/>}/>
30           <Route path='/comment' element={<Comment/>}/>
31           <Route path='/dns' element={<RCE/>}/>
32           <Route path='/data.xml' element={<XML/>}/>
33           <Route path='/robots' element={<Robots/>}/>
34           <Route path='/create' element={<Create/>}/>
35           <Route path='/login' element={<Login/>}/>
36           <Route path='/set-admin' element={<SetAdmin/>}/>
37           <Route path='/admin-function' element={<AdminFunction/>}/>
38           <Route path='/info' element={<Info/>}/>
39           <Route path='*' element={<Home/>}/>
40         </Routes>
41       </BrowserRouter>
42     </div>
43   );
44 }
45
46 export default App;
```

**bwa-ui/src/App.test.js**

```
1 import { render, screen } from '@testing-library/react';
2 import App from './App';
3
4 test('renders learn react link', () => {
5   render(<App />);
6   const linkElement = screen.getByText(/learn react/i);
7   expect(linkElement).toBeInTheDocument();
8 });
```

**bwa-ui/src/http-common.js**

```
1 import axios from "axios";
2
3 export default axios.create({
4   baseURL: process.env.REACT_APP_API_BASE_URL || 'http://
        localhost:3000',
5   headers: {
6     "Content-type": "application/json"
7   }
8 });
```

**bwa-ui/src/index.css**

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

**bwa-ui/src/index.js**

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

reportWebVitals();
```

**bwa-ui/src/scroll.js**

```
function smoothScroll() {
  window.addEventListener('wheel', function(event) {
    if (event.deltaY > 0) {
      window.scrollBy({
        top: window.innerHeight,
        behavior: 'smooth'
      });
    } else if (event.deltaY < 0) {
      window.scrollBy({
        top: -window.innerHeight,
        behavior: 'smooth'
      });
    }
  });
}

export { smoothScroll };
```

**bwa-ui/src/set-cookie.js**

```
function setCookie(name, value, exp) {
  const expDate = new Date();
  expDate.setTime(expDate.getTime() + (exp * 24 * 60 * 60 * 1000));

  const expDateBR = expDate.toGMTString();

  document.cookie = `${name}=${value}; expires=${expDateBR}; path=/`;
}

export default setCookie;
```

# APÊNDICE B – ARTIGO SBC

# Ambiente para Aprendizado e Prática de Técnicas de Segurança Ofensiva em Aplicações Web Vulneráveis

**Samuel Cardoso**

[1]Curso de Bacharelado em Ciência da Computação - Departamento de Informática e Estatística - Universidade Feraderal de Santa Catarina (UFSC) - 88040-900 Florianópolis - SC - Brasil

samuelcardosodejesus@hotmail.com

***Abstract.*** *This work discusses the state of the art of information security based on the OWASP Top Ten:2021, lists ten vulnerabilities that still have an impact on web applications in the year 2023 and develops an environment for learning and practive offensive security techniques through a web applications with these vulnerabilities. This environment is developed in Docker so that the installation of the environment is simplified, with this, the time spent with the environment is totally focused on exploiting the vulnerabilities and not on installing the system.*

***Resumo.*** *Este trabalho discorre sobre o estado da arte da segurança da informação com base na OWASP Top Ten:2021, elenca dez vulnerabilidades que ainda impactam as aplicações web no ano de 2023 e desenvolve um ambiente para aprendizado e prática de técnicas de segurança ofensiva por meio de uma aplicação web com estas vulnerabilidades. Este ambiente é desenvolvido em Docker para que a instalação do ambiente seja simplificada, com isso, o tempo gasto com o ambiente fica totalmente focado na exploração das vulnerabilidade e não na instalação do sistema.*

## 1. Introdução

Para prevenção contra infortúnios, muitas empresas estão utilizando técnicas ofensivas para monitorar o nível de segurança e aperfeiçoar suas defesas, já que é mais fácil corrigir uma vulnerabilidade ao ter o conhecimento dela [Vieira 2022]. Existem, porém, complicações e barreiras no aprendizado de segurança ofensiva, uma vez que há uma linha tênue entre legalidade e ilegalidade quando se realiza testes em sites de terceiros.

Visto a lacuna existente na área para aplicar o conhecimento teórico de forma prática, este trabalho se propõe a desenvolver um ambiente para aprendizado e prática de técnicas ofensiva em aplicações web vulneráveis. Ambiente este, desenvolvido em Docker para facilitar a execução do mesmo em múltiplos Sistemas Operacionais, permitindo o estudo dessas técnicas sem necessidade de alocar tantos recursos computacionais como outros mecanismos de virtualização fazem [Yadav et al. 2018].

O ambiente proposto é constituído de uma imagem Docker contendo uma aplicação web que ficará disponível para acesso local, a aplicação conterá algumas falhas selecionadas, que em sua maioria podem ser encontradas no OWASP TOP 10:2021. O ambiente foi construído em Docker para facilitar a instalação e otimizar a ocupação de recursos computacionais, uma vez que a maioria dos ambientes com vulnerabilidade estão disponíveis como imagens para Máquinas Virtuais.

OWASP Top 10 que é um framework da OWASP, uma organização sem fins lucrativos, que traz informações sobre as falhas mais utilizadas em um determinado ano. Neste trabalho o OWASP Top 10 utilizado será do ano de 2021 [OWASP 2021]. Desta forma, foram selecionadas falhas da OWASP Top 10:2021, pois como são frequentemente encontradas falhas novas de segurança [Hiesgen et al. 2022], algumas técnicas passam a ser mais usadas e outras caem em desuso. Por isso, é importante para quem estuda tais técnicas, praticar em falhas atuais.

## 2. Ambientes de teste

Parte das dificuldades encontradas na área de aprendizado de segurança cibernética podem ser sanadas pela utilização de ambientes de teste. O mesmo não tem o mesmo valor que uma aplicação real, mas pode ser muito útil para pessoas que querem testar novas ferramentas, explorar novas vulnerabilidades ou iniciar o seu estudo na área da segurança da informação.

Um ambiente de testes, consegue isolar os testes para a própria máquina local e isso na segurança da informação é muito importante, já que não é possível para *pentesters* ou profissionais que estejam testando vulnerabilidades, testarem aplicações reais sem as devidas permissões. Ao mesmo tempo, realizar determinados testes em aplicações reais pode gerar custos desnecessários e perdas econômicas no caso da indisponibilidade do serviço, acontecimento este que poderia ser mitigado utilizando um ambiente próprio para a validação de vulnerabilidades da aplicação.

Profissionais de segurança da área defensiva da segurança da informação podem utilizar destes ambientes para entender melhor certas falhas e com base nisto aperfeiçoar seus próprios sistemas sem ter que colocar em um primeiro momento seus sistemas a prova. Assim, um ambiente *Sandbox* para testes de vulnerabilidade se torna útil não apenas aos profissionais que querem se aperfeiçoar no ataque de sistemas, mas também aos profissionais de cibersegurança defensiva que querem melhorar a segurança de suas aplicações.

É importante que estes ambientes, quando disponíveis para execução em máquinas de usuário, sejam executados localmente de forma isolada, uma vez que as vulnerabilidades encontradas nas aplicações podem resultar em vulnerabilidades mais graves de rede e/ou sistema operacional. Alguns bons exemplos de ambientes e plataformas que aulixiam no aprendizado de tecnicas de segurança ofensiva, são:

A Mutillidae II, uma aplicação web open-source que oferece mais de 40 vulnerabilidades e desafios com vulnerabilidades encontradas desde a OWASP TOP Ten de 2007 até a de 2017. O ambiente conta com tutoriais e dicas para auxiliar no aprendizado e possui várias falhas simples de sempre exploradas para estimular o aprendizado da área [OWASP 2017].

A VulnHub, uma plataforma que disponibiliza máquinas virtuais propositalmente vulneráveis para auxiliar no aprendizado da área de segurança da informação. Essas máquinas virtuais ficam à disposição no site para download e são normalmente divididas por dificuldade, tendo níveis variados para auxiliar desde iniciantes na área de segurança até profissionais mais experientes [g0tmi1k 2023].

A TryHackMe, uma plataforma não apenas de disponibilização de laboratórios

de segurança, mas também de treinamentos de segurança, sobre as áreas de análise de vulnerabilidade, testes de penetração, bases da segurança e área correlatadas. Uma plataforma que não foca em apenas um tipo de usuário, mas disponibiliza materias, laboratórios e treinamentos para vários níveis de expertise em segurança da informação [TryHackMe 2023].

A ImmersiveLabs, uma plataforma de aprendizado de segurança, contudo é uma plataforma com uma abordagem mais prática e mais voltada para execução de laboratórios de segurança virtuais com inúmeros temas e objetivos, alguns até mesmo com objetivos de prática forense. Parte dos Capture the Flags criados pela ImmersiveLabs são baseados em situações reais [Viegas and Kuyucu 2022].

Existem inúmeros ambientes de aprendizado de segurança cibernética ofensiva, parte deles é disponibilizado de forma online, parte deles são disponibilizados como o download de máquinas virtuais. Este trabalho desenvolverá um ambiente semelhante à aplicação Mutillidae II, mas seu desenvolvimento será executado em outras tecnologias para serem disponíveis via imagens Docker, tornando o ambiente mais leve, uma vez que não necessita da virtualização de um SO, usando a Docker Engine para compartilhar os recursos do Kernel [Penso 2020].

## 3. Vulnerabilidades

Vulnerabilidades são falhas encontradas em um serviço, sistema, rede, aplicativo ou em algum processo que permita a exploração e por conseguinte a execução de operações indevidas. Existem inúmeros motivos para a existência de uma vulnerabilidade, envolvendo falhas introduzidas por programadores, configurações impróprias de sistemas, a não atualização de serviços, entre outros. Seus impactos, assim como são motivos, são muito variáveis, alguns resultam vazamentos de dados outros podem chegar a conceder o controle total de um sistema a um atacante [Galdino 2022].

As ameaças podem ser entendidas como todo e qualquer vetor de ação que utilize uma vulnerabilidade para a realização de uma operação indevida. Então não apenas atacantes mal-intencionados ou malwares são considerados ameaças, mas também usuários, administradores de sistema e até mesmo complicações em ambientes físicos, como um impacto em um disco de armazenamento ou um furacão, pois estes são potenciais ameaças ao pleno funcionamento de um sistema [Galdino 2022].

O nome dado para a técnica, código ou ferramenta utilizada para explorar uma vulnerabilidade é *exploit*. Os objetivos dos exploits variam conforme a vulnerabilidade que exploram, e eles são divididos entre os exploits conhecidos e os desconhecidos, que tão são chamados de *0-day*. Contra o primeiro grupo de exploits existem normalmente medidas a serem tomadas para mitigar as vulnerabilidades, já contra o segundo grupo não apenas a mitigação é dificultada como a própria identificação da vulnerabilidade [Albors 2023].

### 3.1. OWASP

A OWASP é uma organização sem fins lucrativos que como projeto open-source aspira auxiliar no desenvolvimento de aplicações e ambientes mais seguros. Dentre seus projetos existe a OWASP TOP:10 focado em informar aos profissionais de segurança e demais

pessoas interessadas no estado atual da segurança mundial quais tipos de vulnerabilidades tem sido mais exploradas, chamando atenção para estas falhas mais comumente exploradas por invasores e mostrando assim aspectos novos a serem observados com mais atenção nas aplicações. É importante frisar que vários tipos de vulnerabilidades são difíceis de serem quantificadas e monitoradas, por isso a OWASP não se baseia apenas em números de detecções, mas também no estado da arte, publicações e estudo de vulnerabilidades atuais [OWASP 2021].

No ano de 2021 a OWASP TOP:10 publicou um novo informativo, classificando distintamente os tipos de vulnerabilidades antes previstos, criando categorias e alterando também o ranking de cada tópico. Neste ano os tópicos se baseiam muito na adequação as leis que envolvem a proteção de dados dos usuários e com base neles, parte do trabalho será desenvolvido. Nos próximos parágrafos serão apresentados os tópicos do OWASP TOP:TEN de 2021 respectivamente.

Broken Access Control é um tipo de vulnerabilidade caracterizada pelo acesso a um ativo indevidamente por um usuário, sistema ou dispositivo não autorizado [Hassan et al. 2018]. Em sistemas as permissões de acesso deveriam ser bem definidas. Quando um sistema, que precisa segregar funções ou recursos, não implementa corretamente as permissões baseadas em acesso, independe se por meio de sessão, autorização ou autenticação, esse sistema se torna vulnerável. Os ataques de Broken Access podem se dar por acessos direto em URL, manipulação de parâmetros, previsibilidade de sessões, privilégios excessivos, falta de verificação de autorização, entre outros.

Cryptographic Failures são as vulnerabilidades que expõem dados sensíveis de uma aplicação por conta de uma criptografia fraca ou inexistente [Nagori 2023]. Os dados podem ir desde informações médicas até senhas ou números de cartão de crédito. Nas aplicações modernas os dados passam por manipulações de dados em repouso e em trânsito, tornando ainda mais necessário um controle rígido de criptografia para que essas vulnerabilidades sejam dificilmente aproveitadas. Este tipo de vulnerabilidade estava em terceiro lugar no ano de 2017, mas subiu para segundo lugar no ano de 2021 [OWASP 2021], pois problemas de segurança causados por senhas *hard-coded* tem se tornado muito comuns.

Injection é a classe de vulnerabilidades que engloba as falhas que ocorrem no momento que um dado é injetado em uma aplicação e consegue com isso alterar o comportamento planejado da aplicação, tornando, por exemplo, uma simples *string* que serviriam como nome em um código malicioso executável [Ibarra-Fiallos et al. 2021]. Essas falhas geralmente ocorrem por uma falta de validação das entradas ou uma neutralização imprópria das entradas das aplicações.

Insecure Design é a classe de vulnerabilidades existentes quando um projeto possui problemas de segurança de forma que mesmo que as melhores práticas de desenvolvimento sejam utilizadas o sistema ainda continuará a ser inseguro. Então a causa da falha já faz parte do design escolhido para o projeto que podem levar a, por exemplo, vazamento de informações da aplicação [Nurbojatmiko et al. 2022].

Security Misconfiguration em resumo são as falhas de segurança motivadas por falta de configuração ou configuração padrão em aplicações e/ou serviços [Eshete et al. 2011]. Quando um banco de dados é configurado com valores padrão e ex-

ecutado em porta padrão, ele carrega inúmeros problemas e falhas de segurança. Falhas que podem ser facilmente encontradas por códigos maliciosas que procuram automaticamente serviços executando em determinadas portas. Nas Security Miscunfigurations também estão faltas por falta de configurações de header.

Vulnerable and Outdated Components é a classe que engloba as vulnerabilidades que envolvem as falhas por conta de componentes de software datados ou que possuem falhas conhecidas adicionadas ao serviço em questão [Galvão 2022]. Durante o desenvolvimento de novos serviços é difícil criar tudo do zero, sendo comum, que sejam utilizados componentes prontos de terceiro, o problema é que quando um componente com falha é adicionado em uma aplicação, essa aplicação obviamente herda a vulnerabilidade que pode ou não ter algum peso no contexto da aplicação.

Identification and Authentication Failures é a classe que aloca as falhas que envolvem controle de autenticação, como quebra em gerenciamento de sessão de usuários por má implementação do serviço [Team 2022]. Essas falhas podem ter origens como: não tratamento para força bruta, permissão de senhas fracas, permissão de usuários default, processos ineficiente de recuperação de credenciais, e reutilização de identificadores de sessão. Muitas ferramentas são desenvolvidas para automatizar a exploração dessas vulnerabilidades, no caso de não tratamento de força bruta é possível explorar com ataques de dicionário.

Software and Data Integrity Failures é a classe de vulnerabilidades que se aproveitam da falta de confirmação de integridade do software e de dados [Sengupta 2022]. Quando em um ambiente, podem levar a corrupção de dados, manipulação de informação, violação de privacidade, entre outros. Pode ocorrer, por exemplo, quando uma aplicação usa fontes externas não confiáveis, quando não existe um pipeline seguro no CI/CD, quando um invasor pode obter acesso a dados e/ou estruturas que possam ser vistas e alteradas.

O item Security Logging and Monitoring Failures da OWASP TOP Ten vai além de apenas classificar falhas, mas visa abordar os tópicos de resposta e monitoramento dos ataques e ações rotineiras realizadas em um ambiente [Bathla 2021]. Esta etapa se mistura com vários itens do OWASP TOP Ten, uma vez que para um correto monitoramento é necessário ter os logs armazenados seguramente, portanto um design correto para isso é necessário assim como confirmar a integralidade dos logs para se necessário usar os mesmos como provas. Dentro dessa classe de falhas é importante entender que além de logs insuficientes nas aplicações, logs com dados sensíveis também podem gerar problemas de segurança e logs com dados demais podem dificultar o monitoramento.

Server-Side Request Forgery como o nome diz, é uma vulnerabilidade causada pela falsificação de uma requisição do lado do servidor e ela acontece quando o servidor recupera o conteúdo de uma requisição, mas por não validar o URL acaba não garantindo que a requisição chegue ao destino esperado, esta falha tem um CWE mapeado que é a CWE-918 [Jabiyev et al. 2021]. Esta vulnerabilidade consegue enviar as requisições a destinatários mesmo quando protegidos por firewall, VPN ou outro ACL.

## 4. Seleção de vulnerabilidades

Sendo o OWASP Top Ten uma referência em classificação de vulnerabilidades mais relevantes em criticidade e quantidade de um intervalo de tempo, o projeto se torna um

bom ponto de partida para a seleção de vulnerabilidades. Selecionar vulnerabilidades relacionadas a OWASP enfatiza um estudo de impacto imediato no estado da arte da segurança cibernética.

O trabalho visa montar um ambiente para prática de técnicas de segurança ofensiva, por tanto, se faz necessário a existência de instruções básicas de como explorá-las. A OWASP traz uma abordagem sistemática das vulnerabilidades, deixando as informações de forma estruturada, porém, importante ainda frisar que esse trabalho não se propõe a utilizar apenas vulnerabilidades documentadas pela OWASP e sim abordar os tópicos selecionados como um ponto de partida e quando de interesse comum, recorrer às vulnerabilidades mapeadas.

Alguns tópicos do OWASP Top Ten podem não ser tão instigantes de serem desenvolvidas neste trabalho, uma vez que tenham suas falhas mais comuns, tenham um envolvimento maior com segurança defensiva e compliance, como as vulnerabilidades do tópico A09:2021-Security Logging and Monitoring Failures. Com base nas informações deste capítulo foram escolhidas 9 CWEs (Common Weakness Enumeration) nos tópicos da OWASP para se transformarem em 10 vulnerabilidades do ambiente, estas são CWEs dividas nos tópicos do:

- A01:2021 - Broken Access Control
  - CWE-284: Improper Access Control - Nível de criticidade: média
  - CWE-285: Improper Authorization - Nível de criticidade: alta
- A02:2021 - Cryptographic Failures
  - CWE-200: Exposure of Sensitive Information to an Unauthorized Actor - Nível de criticidade: alta
- A03:2021 - Injection
  - CWE-79: Improper Neutralization of Input During Web Page Generation - Criticidade média
  - CWE-400: Uncontrolled Resource Consumption - Nível de criticidade: crítica
  - CWE-89: Improper Neutralization of Special Elements used in an SQL - Nível de criticidade: alta
  - CWE-94: Improper Control of Generation of Code - Nível de criticidade: crítica
- A05:2021 - Security Misconfiguration
  - CWE-22: Improper Limitation of a Pathname to a Restricted Directory - Nível de criticidade: alta
- A08:2021 - Software and Data Integrity Failures
  - CWE-784: Reliance on Cookies without Validation and Integrity Checking in a Security Decision - Nível de criticidade: alta

## 5. Tecnologias de Desenvolvimento

Existe uma similaridade entre as falhas das aplicações da OWASP Broken Web Applications e o ambiente proposto neste trabalho, os tipos de falhas encontrados nesta máquina são muito semelhantes às falhas desenvolvidas neste ambiente, porém a grande diferença é que este ambiente será desenvolvido em Docker com a intenção de facilitar a instalação e diminuir a ocupação de dados alocados em máquina. Enquanto uma imagem ISO Ubuntu

tem cerca de 5Gb e a OWASP BWA tem 1.7Gb, todos os containers docker desenvolvidos neste trabalhos juntos ocupam cerca de 1Gb.

O Docker é um sistema que automatiza a implementação e gerenciamento de *containers*. O Docker diferente dos sistemas de virtualização não necessita de Hardware virtualizado e nem necessariamente de uma instalação completa de um S.O. dentro do *container* [Combe et al. 2016]. Para seu funcionamento ele usa algumas *features* do *kernel* como o Cgroups e o namespaces, essas features são utilizadas para rodar os processos independentemente. Este modelo permite uma melhor organização da infraestrutura e, em simultâneo, mantém os ambiente seguros já que eles são separados.

Neste trabalho, o Node será utilizado como um provedor de API, requisitando dados do banco de dados da aplicação e executando operações em SO para disponibilizar arquivos e executar operações. Algumas bibliotecas serão utilizadas em conjunto ao NodeJS para possibilitar as funcionalidades citadas acima, estas são: cors, dotenv, express, mysql2 e sequelize.

O React neste trabalho, ficará responsável por gerir as rotas da aplicação web, fazer a conexão com o back-end, renderizar a UI do projeto e será também a origem de vulnerabilidades, que serão explicadas no capítulo 5. Algumas bibliotecas serão utilizadas em conjunto ao React para possibilitar as funcionalidades citadas acima, estas são: axios, web-vitals, react-dom, react-router-dom, react-scripts.

O MySQL será utilizado neste trabalho para auxiliar nos controles de acesso de usuário na aplicação mantendo informações de usuário e credenciais. Ele será outro vetor de ataques da aplicação web.

Na Figura 1 é possível visualizar o diagrama do ambiente desenvolvido, contendo três containers docker rodando pela Docker Engine na máquina Host. Um container rodando a UI da aplicação em React, um container MySQL armazenando os dados e um container NodeJS servindo como API para disponibilizar dados e serviços.

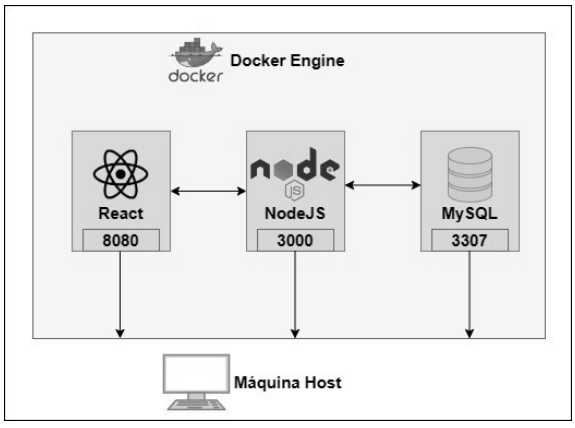


**Figure 1. Diagrama do ambiente projetado**

## 6. Desenvolvimento

Este capítulo primeiramente apresentará como foi realizado a configuração inicial do ambiente com docker, react, node e mysql. Em um segundo momento, serão apresentadas as

vulnerabilidades encontradas no sistema, uma breve revisão delas e como foram implementadas no sistema.

O controle de build e disponibilidade do sistema foi configurado por um arquivo chamado *docker-compose.yml*, este arquivo, divide os serviços do projeto em três sessões, a ordem que devem ser buildados, os volumes e as conexões de rede entre eles. Na Figura 2, se tem a definição do serviço 'mysqldb', responsável pelo banco de dados do ambiente.

```
services:
 mysqldb:
   image: mysql:5.7
   restart: unless-stopped
   env_file: ./.env
   environment:
     - MYSQL_ROOT_PASSWORD=$MYSQLDB_ROOT_PASSWORD
     - MYSQL_DATABASE=$MYSQLDB_DATABASE
   ports:
     - $MYSQLDB_LOCAL_PORT:$MYSQLDB_DOCKER_PORT
   volumes:
     - db:/var/lib/mysql
   networks:
     - backend
```

**Figure 2. Configuração do serviço de MySQL**

Este arquivo está definindo que a imagem Docker base para o serviço deve ser a "mysql:5.7", informar que o container deve ser reiniciado sempre que for interrompido por algum problema a menos que seja parado manualmente, define o arquivo de *environment* como "./.env", configura algumas variáveis de ambiente que precisam ser predefinidas para o serviço ser corretamente iniciado, seta as porta que o serviço será disponibilizado na máquina host, cria um volume para a persistência dos dados e configura o serviço para executar no network denominado "backend".

```
services:
 bwa-api:
   depends_on:
     - mysqldb
   build: ./bwa-api
   restart: unless-stopped
   env_file: ./.env
   ports:
     - $NODE_LOCAL_PORT:$NODE_DOCKER_PORT
   environment:
   networks:
     - backend
     - frontend
```

**Figure 3. Configuração do serviço de BWA-API**

No código da Figura 3 está definindo que antes do build do “bwa-api”, o banco de dados deve ser executado, informar que no diretório “./bwa-api” deve ser procurado um Dockerfile com informações do build do serviço específico, define o arquivo de *environment* como “./.env”, configura algumas variáveis de ambiente que precisam ser predefinidas para o serviço ser corretamente iniciado, seta as porta que o serviço será disponibilizado na máquina host e configura o serviço para executar nos networks *backend* e *frontend*.

```
services:
  bwa-ui:
    depends_on:
      - bwa-api
    build:
      context: ./bwa-ui
      args:
        - REACT_APP_API_BASE_URL=$CLIENT_API_BASE_URL
    ports:
      - $REACT_LOCAL_PORT:$REACT_DOCKER_PORT
    networks:
      - frontend
```

**Figure 4. Configuração do serviço de BWA-UI**

No código da Figura 4 fica definido a ordem de *build* do sistema, e o *bwa-ui* deve ser montado após o *bwa-api*, seu build deve acontece com base no Dockerfile encontrado no path “./bwa-ui” levando como argumento o “$CLIENT_APLI_BASE_URL”, tendo a porta setada com base na variável de ambiente do docker-compose e configurando o serviço para executar no network “frontend”.

```
volumes:
  db:

networks:
  backend:
  frontend:
```

**Figure 5. Configuração de volumes e networks**

Por fim, o arquivo *docker-compose.yml* representado na Figura 5, nomeia e define os volumes e networks do ambiente.

O MySQL por ter um uso mais padrão teve seu build completamente definido através do arquivo *docker-compose.yml*, em contraponto, o back-end e front-end da aplicação possuem necessidades específicas e por isso, um arquivo separado foi criado para cada um deles responsável por configurar seus *ENTRYPOINTS* assim como a imagem Docker base para a aplicação e transferir os arquivos de execução para dentro da imagem.

Na Figura 6 é possível ver o Dockerfile do back-end definindo a imagem Docker base como “alpine:latest”, as imagens *alpine* são mais leves e por isso foram escolhidas

para este projeto. Nesta distribuição linux é então instalado o *NodeJS*, o *Bash* e o *bind-tools*. Em seguida o WORKDIR é definida para "/bwa-api", os arquivos do back-end são copiados para a pasta e o *Node* é inicializado.

```
FROM alpine:latest

RUN apk add --update nodejs npm && apk add bash && apk add bind-
   red↪ tools

WORKDIR /bwa-api
COPY package.json .
RUN npm install
COPY . .
CMD npm start
```

**Figure 6. Dockerfile do back-end**

Na Figura 7 está o código Dockerfile do front-end definindo a imagem Docker base como "node:16-alpine", copiando os arquivos do front-end para dentro do WORKDIR
"/usr/src/app", instalando as dependências e inicializando o serviço.

```
FROM node:16-alpine

RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app

COPY package.json .
RUN npm install

COPY . ./

CMD ["npm", "run", "start"]
```

**Figure 7. Dockerfile do front-end**

Com isso, as configurações de containers do ambiente estão concluídas. Agora, os serviços passam a ser o objeto de interesse de desenvolvimento. No back-end o arquivo "server.js" define o *setup*, seu conteúdo será explicado nos próximos parágrafos.

Primeiramente, ele configura as variáveis de ambiente baseados no arquivo ".env", importa o *express* e a biblioteca *cors* para permitir as chamadas http ao servidor e realiza a criação da conexão com o banco ao importar o *db*.

O *express* é ajustado para realizar suas operações junto ao *cors*, que foi previamente configurado para expor alguns de seus *headers*, essa exposição foi feita para permitir a identificação correta de tipos de arquivos no front-end. Com o *express* definido, as rotas são importadas de outro arquivo, chamado "routes.js".

É realizada uma migração de dados, com os dados de usuários base do sistema e por fim, o app *express* é disponibilizada por meio da porta encontrada nas variáveis de ambiente.

O arquivo "./bwa-api/app/models/index.js" cria a conexão com o banco de dados usando a biblioteca de ORM chamada *Sequelize*, também cria um objeto contendo o Sequelize e a conexão é criado, neste, é inserido um model de Usuários, que será definido através do arquivo "./user/model.js".

Esse model de User é responsável pela criação da tabela de usuários no banco de dados, informando as tabelas e tipos. A tabela do model é bem simples, possuindo apenas 4 colunas: name, password, email e admin.

O *Router* do *Express* é utilizado para definir o tipo de requisição e o redirecionamento de funções. No arquivo de cada rota, existe a chamada para um segundo arquivo responsável pelo controle das funções, as receber uma requisição o arquivo direciona para a função responsável baseado na rota e no tipo de *request*. A função então realiza as operações necessárias e retornar um *payload*.

Para construção da UI do front-end foi utilizado o *React* e *ReactDOM*. Uma raiz de renderização é criada através do "ReactDOM.createRoot()" e tem um *App* como *component* renderizado na aplicação. Já as chamadas http no front-end são feitas através da biblioteca *Axios*.

## 7. Implementação das vulnerabilidades

As vulnerabilidades serão divididas em duas sessões de 5 vulnerabilidades cada no sistema, a ideia é que as primeiras vulnerabilidades sejam menos nocivas ao ambiente e as últimas mais críticas. As primeiras ficarão na sessão Genesis e as segundas na sessão Apocalypse. O início da sessão Genesis servirá principalmente para o usuário entender o sistema e explorar as rotas, os códigos encontrados no *client* e testar algumas *injections*.

### 7.1. CWE-284: Improper Access Control

A CWE-284 é uma vulnerabilidade definida sobre quando um sistema falha em impor condições de proteção a recursos que deveriam ser destinados a usuários específicos, então quando um usuário não logado ou com acesso insuficiente consegue acessar uma página restrita e conseguir coletar os dados desta página, uma CWE-284 é explorada.

Nesta CWE também contam os casos onde o sistema permite o uso de senhas fracas ou quando o sistema permite escalação de privilégios, mas este trabalho se baseará numa página sem a implementação de validação de acesso privilegiado.

Para a criação desta vulnerabilidade não houve grandes dificuldades, uma vez que bastava criar uma rota sem realizar verificações de identidade com dados sensíveis e que deveriam em um primeiro momento serem destinadas a um administrado, para isso, a rota "/admin-control" foi designada, contendo uma página com informações de contas administrativas.

### 7.2. CWE-200: Exposure of Sensitive Information to an Unauthorized Actor

A CWE-200 representa uma categoria de vulnerabilidades onde acontece vazamento de informações confidenciais. Vazamentos que podem acontecer durante a transmissão dos

dados ou pelos dados não serem devidamente protegido em uma aplicação. Este trabalho conta com duas vulnerabilidades da categoria CWE-200.

Para no sistema haver essas vulnerabilidades alguns arquivos ficaram disponíveis no front-end. O *path* que leva a um dos arquivos foi adicionado ao "disallow" do "robots.txt". O que parece uma medida de segurança, para não permitir a indexação dos arquivos por robôs, também realiza o vazamento de dados, informando uma URL acessíveis com dados sensíveis.

Além disso, no HTML de algumas páginas da aplicação, foram adicionados comentários vazando credenciais de segurança.

### 7.3. CWE-79: Improper Neutralization of Input During Web Page Generation

A categoria CWE-79 baseada na injeção de código em aplicações web, essa injeção de código pode originar um exploit de XSS (Cross-site scripting).

Contra esse tipo de ataque aplicações React possuem algumas defesas por padrão. Por padrão, as variáveis simples dentro React são carregadas como *string* independentemente se tiverem tags HTML no corpo do texto, isso garante um nível de segurança ao React conta DOM-base XSS exploits, porém isso ainda não torna as aplicações React totalmente seguras contra React.

Em um cenário onde se pretende que um HTML seja posteriormente renderizado na página, é possível passar uma propriedade no elemento HTML chamada "dangerously-SetInnetHTML", a propriedade tem esse nome justamente pelo perigo que envolve utilizar ela em uma página [Bilgili 2021]. Esta será usada para permitir um usuário escreva textos e elementos na página em tempo real, ao mesmo tempo, isso permite que exploits XSS sejam executados na página, o que é o objetivo dessa implementação. Foi adicionada uma div que recebe um "$dangerouslySetInnerHTML = \_html : aboutUserText/ >$", assim, ao receber a string do usuário, o html é renderizado e a página se torna vulnerável.

### 7.4. CWE-22: Improper Limitation of a Pathname to a Restricted Directory

A categoria da CWE-22 representas as falhas de sistema e aplicação em limitar e restringir o acesso a diretórios e/ou pathnames. Uma falha como essa pode permitir que um usuário tenha informações restritas da aplicação e/ou sistema, consiga recuperar arquivos e em casos mais graves até mesmo realizar modificações no sistema.

Ocorrem geralmente quando existem entradas externas para a criação do *path* de um determinado diretório, porém a entrada não é neutralizada e valores como "../" conseguem participar do *path*. Isso pode permitir, por exemplo, que diretórios indevidos sejam acessados como entradas do estilo "$../../ < archieve\_name >$".

Justamente essa vulnerabilidade será implementada no projeto. Uma página no front-end foi criada, nela uma lista de nomes de arquivos com descrições de animais foi disposta e ela pede que o usuário informe o nome de um arquivo para download. A entrada esperada seria algo como "abelha.txt", mas como o nome é usado na construção do *path* do diretório sem corretas neutralizações de valores, o usuário pode recuperar outros arquivos de sistema.

No front-end, o campo inserido pelo usuário é adicionado ao fim da *string* "../../img/" e enviado como requisição ao back-end. No back-end esse valor é recuperado

e adicionado ao *path* atual em que este arquivo está rodando, então o arquivo correspondente a este *path* é disponibilizado para download. Para adicionar essa vulnerabilidade, o texto de entrada do usuário vai diretamente para o comando que cria o path, sendo este: "$constfilePath = path.resolve\_\_dirname, filename$".

### 7.5. CWE-400: Uncontrolled Resource Consumption

A CWE-400 agrupa as vulnerabilidades de aplicação ou sistema que não fazem uma gestão adequada de recursos de execução, tornando o sistema/aplicação vulnerável a ataques que consumam completamente seus recursos. Os recursos podem ser consumidos individualmente ou em conjunto, e variam entre consumo de CPU, memória RAM, armazenamento físico ou até mesmo a banda de internet. Muitas vezes quando um ou mais desses recursos é esgotado, o serviço para de funcionar corretamente por um período ou indefinidamente.

Entre as vulnerabilidades que causam negação de serviço estão as ReDoS (Regular expression Denial of Service). A negação de serviço por um *regex* é possível pela forma com as expressões regulares confirmam se uma string equivale a uma máquina de estados. Para entender melhor, o *regex* implementado no trabalho será explicado.

Uma validação de e-mail é normalmente feita com uma expressão regular como esta: "/[^@]*[^@]*/". Nesta expressão existem 3 estados, A, B e $, enquanto os caracteres forem quaisquer exceto "@" a máquina permanece no estado A, quando aparecer um "@" ela vai para o estado B e recebe quantos caracteres existirem desde que não sejam "@" e então ela finaliza fazendo a transição para o estado $.

Nisto, a operação da máquina de estado é bem simples e não exige muitos recursos computacionais, porém essa expressão não consegue englobar qualquer endereço de e-mail, uma vez que segundo a especificação formal de endereços de e-mail é possível que um endereço tenha mais de um "@" desde que este caractere esteja entre aspas [Resnick 2008]. Se um serviço tentar garantir que isso seja válido através de uma expressão regular, ele usará algo como '/("[^"]*"—[^@])*@[^@]*/'.

Com essa nova expressão regular novos e-mails serão aceitos como endereços válidos, porém um não determinismo foi adicionado na expressão, criando uma árvore que possibilita que uma aspa tanto continue no estado A, quando vá para um estado C. Existem abordagens que solucionam esse problema, porém em JavaScript a máquina de estados simplesmente entende que existem vários caminhos e decide um para testar. No caso de sucesso a máquina retorna que o valor é válido, porém quando a *string* não bate com o esperado a máquina de estados voltando realizando um *backtracking*. Este *backtrack* possui um risco de escalabilidade exponencial de consumo de recursos já que no pior dos casos ele testará todas as possibilidades até retorna que não existe um correspondente para a expressão esperada.

A implementação desta vulnerabilidade foi estruturada na criação de usuário do ambiente, que ao receber nome, e-mail e senha do usuário, verifica se o e-mail é válido utilizando o regex '/("[^"]*"—[^@])*@[^@]*/' e então o cria.

### 7.6. CWE-89: Improper Neutralization of Special Elements

A categoria CWE-89 define as vulnerabilidades por falta de neutralização e/ou sanitização de caracteres especiais nos valores de entrada usados especificamente para SQL Injec-

tions. Pela falta de sanitização, quando os valores são adicionados a consulta SQL, um atacante consegue alterar a consulta para uma operação com comportamento divergente do esperado. A alteração pode comprometer o banco de dados, alterando valores, resgatando informações ou até deletando informações.

Para esta vulnerabilidade ser adicionado ao projeto, uma tela de login de usuário foi criada e com base no Sequelize, foi implementado uma *query* de seleção com base e-mail e senha do usuário, esta *query* é a seguinte: "SELECT name, admin FROM users WHERE email = '$email' AND password = '$password'".

### 7.7. CWE-285: Missing Authorization

Semelhante a CWE-284, mas ainda distinta, a CWE-285 é a categoria de vulnerabilidades onde a autorização aplicada em um serviço é insuficiente ou incorreta, permitindo a autorização a um usuário não autorizado. Quando um recurso fica a disposição de um usuário sem a permissão supostamente necessária para a utilização deste, esse sistema tem uma vulnerabilidade CWE-285.

Para esta vulnerabilidade, a rota "set-admin" foi criada, esta página fica escondida, mas a validação de user dela é muito simples e não exige que o usuário seja um administrador, então é possível acessar ela e utilizar os recursos a partir do momento que o usuário conseguiu um login.

### 7.8. CWE-565: Reliance on Cookies

A categoria CWE-565 acontece quando a aplicação depende dos dados nos Cookies para a realização de operações críticas, mas a aplicação não garante que os Cookies são do usuário da sessão. Os cookies por padrão deveriam ser utilizados para armazenar preferências do usuário e alguns dados, mas utilizar eles para conceder acessos a funções torna essas funções vulneráveis uma vez que eles ficam armazenados no *client*. Com isso, o sistema se torna vulnerável a falsificações de identidade, acesso não autorizado e manipulações de sessão.

Duas páginas nessa aplicação estão vulneráveis a esta falha, a "/set-admin" e a "/ad- min-functions". Elas se baseiam nos cookies para renderizar páginas de perfil administrativo, a diferença é que a "/set-admin" precisa que o usuário esteja logado e a "/admin-functions" precisa que o usuário seja administrador. Essa adição é baseada na verificação de um segundo cookie boolean chamado "admin".

### 7.9. CWE-94: Code Injection

As Code Injections são vulnerabilidades que permitem a execução de código em um sistema e/ou aplicativo. Sistemas normalmente possuem várias entradas de dados, através das URLs, de consultadas por inputs, no upload de arquivo, entre outros. Quando uma dessas entradas não é corretamente sanitizada, o sistema pode se tornar vulnerável a uma Injection, a CWE-94 trata em específico das Injections que permitem execução arbitrária de códigos.

Dessa categoria de vulnerabilidade, a RCE (Remote Code Execution) foi escolhida para implementação. Na máquina **alpine** do back-end, foram instaladas ferramentas que permitem realizar consultas de DNS, entre elas consultas através do comando "dig". Uma página no front-end foi criada com um input que permite o usuário informar um site

que deseja realizar a consulta de DNS, esse input é enviado pela URL para o back-end que insere então o nome do site num comando e devolve o resultado como um arquivo. O código adicionado no *request* foi: "const command = 'result=$(dig $filename) && echo '$result' '/bwa-api/img/result.txt'"".

## References


Albors, J. (2023). Exploit: a chave para aproveitar uma vulnerabilidade.

Bathla, S. (2021). A09:2021-security logging and monitoring failures. https://medium.com/@shivam_bathla/a09-2021-security-logging-and-monitoring-failures-88c1c349f5a6, Last accessed on 2022-11-30.

Bilgili, D. (2021). Using dangerouslysetinnerhtml in a react application.

Combe, T., Martin, A., and Di Pietro, R. (2016). To docker or not to docker: A security perspective. *IEEE Cloud Computing*, 3(5):54–62.

Eshete, B., Villafiorita, A., and Weldemariam, K. (2011). Early detection of security misconfiguration vulnerabilities in web applications. In *2011 Sixth International Conference on Availability, Reliability and Security*, pages 169–174.

g0tmi1k (2023). About vulnhub. https://www.vulnhub.com/about/, Last accessed on 2023-06-01.

Galdino, G. (2022). Qual a diferença entre vulnerabilidade, ameaça e risco?

Galvão, P. L. (2022). Analysis and aggregation of vulnerability databases with code-level data.

Hassan, M., Ali, M., Bhuiyan, T., Sharif, M., and Biswas, S. (2018). Quantitative assessment on broken access control vulnerability in web applications. In *International Conference on Cyber Security and Computer Science 2018*.

Hiesgen, R., Nawrocki, M., Schmidt, T. C., and Wählisch, M. (2022). The race to the vulnerable: Measuring the log4j shell incident.

Ibarra-Fiallos, S., Higuera, J. B., Intriago-Pazmiño, M., Higuera, J. R. B., Montalvo, J. A. S., and Cubo, J. (2021). Effective filter for common injection attacks in online web applications. *IEEE Access*, 9:10378–10391.

Jabiyev, B., Mirzaei, O., Kharraz, A., and Kirda, E. (2021). Preventing server-side request forgery attacks. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 1626–1635.

Nagori, V. (2023). Quantum computing posing a challenge to businesses.

Nurbojatmiko, Lathifah, A., Bil Amri, F., and Rosidah, A. (2022). Security vulnerability analysis of the sharia crowdfunding website using owasp-zap. In *2022 10th International Conference on Cyber and IT Service Management (CITSM)*, pages 1–5.

OWASP (2017). Owasp mutillidae ii. https://owasp.org/www-project-mutillidae-ii/, Last accessed on 2022-11-30.

OWASP (2021). Welcome to the owasp top 10 - 2021. https://owasp.org/Top10/, Last accessed on 2022-07-12.

Penso, B. (2020). Docker e containers, afinal o que são?

Resnick, P. (2008). Internet message format.

Sengupta, S. (2022). A08:2021 – software and data integrity failures- explained. https://crashtest-security.com/owasp-software-data-integrity-failures/, Last accessed on 2022-11-30.

Team, C. (2022). Identification and authentication failures and how to prevent them. https://cyolo.io/blog/identification-and-authentication-failures-and-how-to-prevent-them/, Last accessed on 2022-11-30.

TryHackMe (2023). Tryhackme. https://tryhackme.com/r/about, Last accessed on 2023-06-01.

Viegas, V. and Kuyucu, O. (2022). *Security Testing and Attack Simulation Tools*, pages 263–290. Apress, Berkeley, CA.

Vieira, Y. D. B. (2022). Utilização de pentest na prevenção de ataques cibernéticos às organizações.

Yadav, R. R., Sousa, E. T. G., and Callou, G. R. A. (2018). Performance comparison between virtual machines and docker containers. *IEEE Latin America Transactions*, 16(8):2282–2288.