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

Samuel Cardoso

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

Florianópolis
2022

Samuel Cardoso

Desenvolvimento de um 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, 2022.

Prof. Jean Everson Martina, Dr
Coordenador do Curso

Banca Examinadora:

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

Fabricio Bortoluzzi
Avaliador
Universidade Federal de Santa Catarina

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

Samuel Cardoso

Desenvolvimento de um 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: Prof. Carla Merkle Westphall, Dra.

Florianópolis
2022

RESUMO

Em um momento onde a tecnologia evolui tão rapidamente, novas vulnerabilidades em aplicações web são encontradas todos os meses enquanto outras são remediadas, isso faz com que o estado da arte da segurança computacional seja atualizado constantemente. Neste contexto, este trabalho se propõe a desenvolver um ambiente para aprendizado e prática de técnicas de segurança ofensiva em aplicações web com vulnerabilidades de alto risco nos últimos anos.

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

ABSTRACT

At a time when technology evolves so quickly, new vulnerabilities in web applications are found every month while others are remedied, this means that the state of the art of computer security is constantly updated. In this context, this work proposes to develop an environment for learning and practicing offensive security techniques in web applications with high-risk vulnerabilities in recent years.

Keywords: Offensive Security. Web Applications. Vulnerabilities.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

A democratização da Internet aumentou exponencialmente o uso de Internet no último século, afetando a vida cotidiana e corporativa das pessoas [**carneiro2022**], porém o aumento do uso da Internet contém consequências. Nos últimos anos problemas com ataques cibernéticos passaram a ser cada vez mais comuns ao redor do mundo e inclusive no Brasil [**avila2013brasil**]. Em contra ponto, políticas e leis tornam-se mais rigorosas quanto ao tratamento e a proteção de dados [**Neves_Lopes_Pavani_Sales_2021**], levando a segurança da informação e a segurança computacional a um novo nível de importância.

Para prevenção contra infortúnios, muitas empresas estão utilizando técnicas de segurança ofensiva para monitorar o nível de segurança e aperfeiçoar suas defesas, já que é mais fácil corrigir uma vulnerabilidade ao ter o conhecimento dela [**vieira2018**]. 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 acerca de segurança ofensiva de forma prática, este trabalho se propõe a desenvolver um ambiente para aprendizado e prática de técnicas de segurança ofensiva em aplicações web vulneráveis. Ambiente este que será 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 [**8528247**].

O ambiente proposto é constituído de uma imagem Docker contendo uma aplicação web que ficará disponível para acesso local, a aplicação conterá algumas falhas de segurança selecionadas, que em sua maioria podem ser encontradas no OWASP TOP 10:2021. O ambiente será 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 [**url:OWASP**]. Desta forma serão selecionadas falhas da OWASP Top 10:2021, pois como são frequentemente encontradas falhas novas de segurança [**https://doi.org/10.48550/arxiv.2205.02544**], 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.

1.2 OBJETIVOS

1.2.1 Objetivos gerais

Este trabalho visa realizar a implementação de um ambiente em Docker contendo aplicações web vulneráveis para facilitar o aprendizado e a prática legal de técnicas de segurança ofensiva.

1.2.2 Objetivos específicos

Os objetivos específicos que podem ser descritos 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;
- Projetar e decidir as tecnologias utilizadas no desenvolvimento do ambiente a fim de que tenha os recursos necessários à execução das vulnerabilidades selecionadas;
- Desenvolver o ambiente 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

2.1 INTRODUÇÃO

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 um ambiente de teste. Obviamente 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 teste, 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 que eles façam estes testes em aplicações reais sem as devidas permissões. Ao mesmo tempo, nem sempre é a melhor opção realizar determinados testes na aplicação real, já que pode gerar custos desnecessários e perdas econômicas no caso do serviço ficar indisponível. Então profissionais de segurança da área defensiva da segurança da informação podem usar destes ambientes para entender melhor certas falhas e então com base nisto aperfeiçoar seus próprios sistemas sem ter que colocar em um primeiro momento seus sistemas a prova. Portanto, 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 que querem melhorar a segurança de alguma aplicação.

Hoje existem vários ambientes de segurança que podem ser utilizados para a realização de testes de vulnerabilidade desenvolvidos com diversas tecnologias, os mais comuns são como aplicações web ou imagens para máquinas virtuais. Este teste descreverá trabalhos desenvolvidos como imagens de sistemas com aplicações executando internamente, uma vez que nestes ambientes existem falhas de segurança web que podem resultar em problemas mais graves como vulnerabilidades em redes e/ou em sistemas operacionais. Tanto a OWASP BWA quando a Mutillidae II são bons exemplos de imagens com falhas muito interessantes.

2.2 OWASP BROKEN WEB APPLICATIONS

O BWA (OWASP Broken Web Applications) é um projeto da OWASP para auxiliar na exploração e aprendizado de falhas de segurança, no teste de ferramentas automatizadas, na observação de ataques em aplicações web, no teste de técnicas de avaliação manual e no teste de WAFs (Web Application Firewall) [**url:OWASPBWA**]. A Figura 1 representa a *homepage*

da imagem BWA.

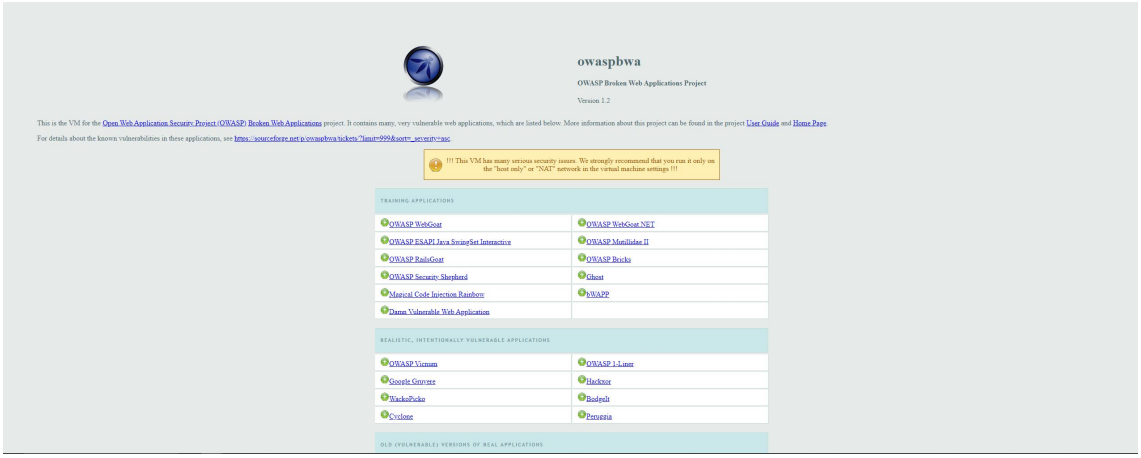

Figura 1 – Captura de tela da OWASP BWA

2.3 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 [**url:owaspmuti**]. Na Figura 2 é possível ver uma das páginas da Mutillidae II.

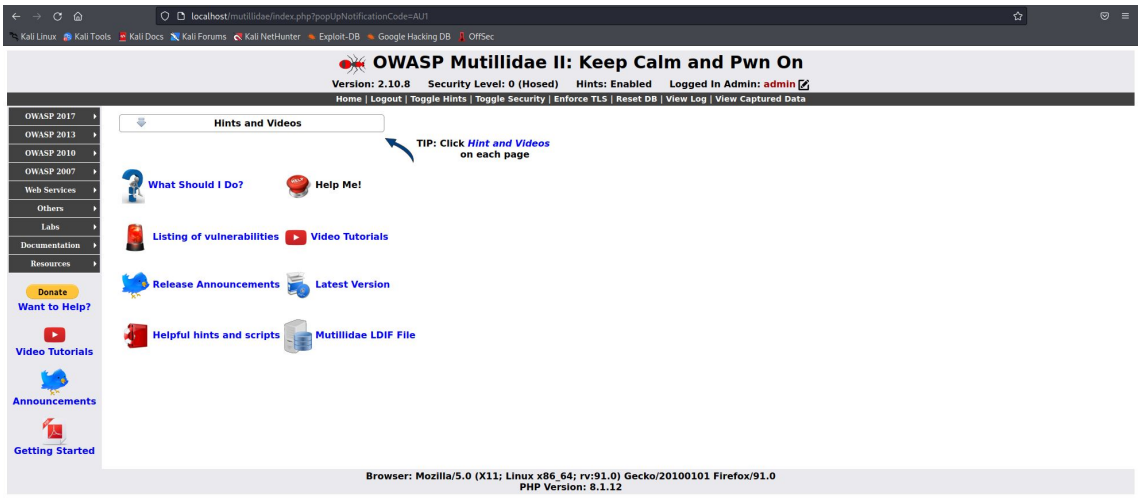

Figura 2 – Captura de tela da Mutillidae II

3 VULNERABILIDADES

3.1 INTRODUÇÃO

A OWASP é uma organização sem fins lucrativos que por projetos 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 o estado atual da segurança mundial, quais tipos de vulnerabilidades tem sido mais exploradas e com isso chamar atenção para as falhas mais comumente exploradas por invasores, 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 [**url:OWASP**].

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 3 são apresentados os 10 maiores riscos do ano de 2021 segundo a OWASP.

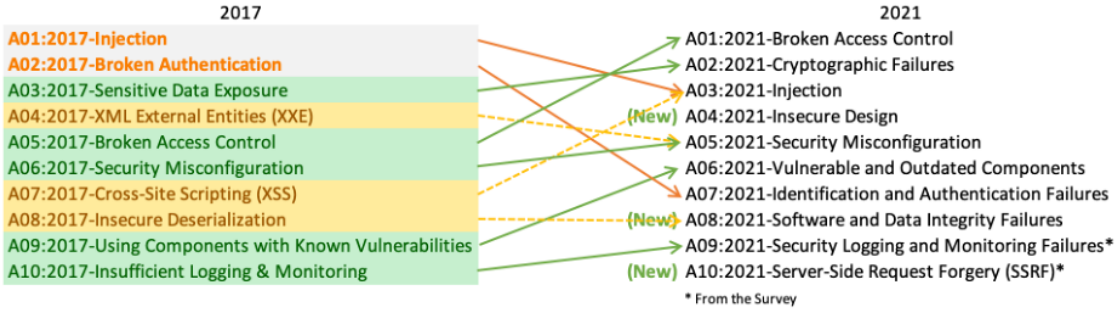

Figura 3 – Top 10 Web Application Security Risks [**url:OWASP**]

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 [**hassan2018quantitative**]. Em sistemas as permissões de acesso deveriam ser bem definidas. 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 [**Jeremy2022**] 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 [**nagoriquantum**]. 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 [**url:OWASP**], pois problemas de segurança causados por senhas *hard-coded* tem se tornado muito comuns.

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 [**9319139**]. 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 [**9935837**]. Uma das vulnerabilidades da classe é justamente a "CWE-209 Geração de mensagem de erro contendo informações confidenciais", mas 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 [**6045929**]. 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.

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 que são adicionados ao serviço em questão [**galvao2022analysis**]. 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.

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 [**authentication**]. 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.

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 [**integrity**]. 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.

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 monitoriamento dos ataques realizados em um ambiente [**logging**]. 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.

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 mapeada que é a CWE-918 [**jabiyev2021preventing**]. Esta vulnerabilidade consegue enviar as requisições a destinatários mesmo quando protegidos por firewall, VPN ou outro ACL.

4 TECNOLOGIAS DE DESENVOLVIMENTO

4.1 DOCKER

O Docker é um sistema de virtualização que se baseia no empacotamento e uso de ambientes em *containers*. O Docker diferente de outros sistemas de virtualização não necessita de Hardware virtualizado e nem necessariamente de uma instalação completa de um S.O. dentro do *container* [**7742298**]. Para seu funcionamento ele usa algumas *features* do *kernel* como o Cgroups e o namespaces, essas features são utilizadas para poder 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 4 compara a execução de imagens em um Linux Container e em Docker.

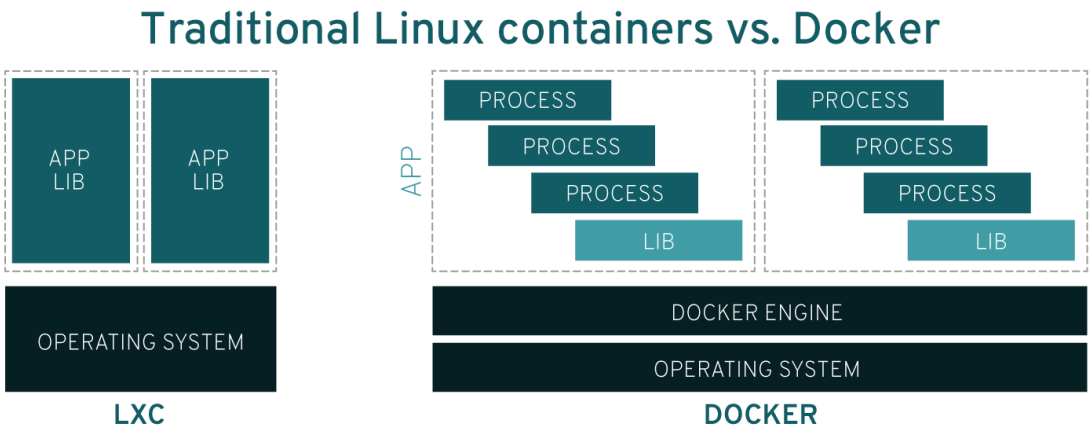

Figura 4 – Figura do site RedHat sobre containers em Linux vs Docker [**docker**]

4.2 TECNOLOGIAS DE APLICAÇÃO

Além do Docker teremos algumas tecnologias que serão utilizadas no desenvolvimento do ambiente proposto.

4.2.1 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 [**7960064**]. O Node foi criado pensando também em aplicações web escaláveis, sendo uma plataforma open-source multi-paradigmas.

4.2.2 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 [**url:react**].

4.2.3 SQLite

O SQLite é um banco de dados relacional open-source prático e acessível que consegue atuar como o próprio servidor, já que ele insere seus dados dentro de si mesmo para funcionar. O nível de segurança desse banco não é tão alto, mas justamente isso também permite explorar outras vulnerabilidades dentro do ambiente [**5231398**].

4.3 AMBIENTE LOCAL

A imagem Docker estará disponível no repositório, após o download basta a execução que através do Dockerfile fará o ambiente ser configurado para permanecer sendo executado apenas de forma local, não permitindo que as aplicações sejam visualizadas pelo mundo exterior. Assim não haverá problemas com tráfego mal-intencionado na rede.

5 PROPOSTA DE DESENVOLVIMENTO

5.1 AMBIENTE PARA APRENDIZADO E PRÁTICA DE TÉCNICAS DE SEGURANÇA OFENSIVA EM APLICAÇÕES WEB VULNERÁVEIS

Existe uma similaridade entre a OWASP Broken, a Mutillidae e o ambiente proposto neste trabalho, os tipos de falhas encontrados nestas duas máquinas são muito semelhantes às falhas que serão 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.

O ambiente será desenvolvido com base em uma imagem Linux Ubuntu 22.04, para que possam ser exploradas não apenas falhas em aplicações web, mas falhas em aplicações web que levem a falhas em nível de sistemas operacional. Mesmo que isto custe um pouco mais de recursos em contra-ponto a uma aplicação que usaria *docker-compose* para manter os serviços rodando separadamente.

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

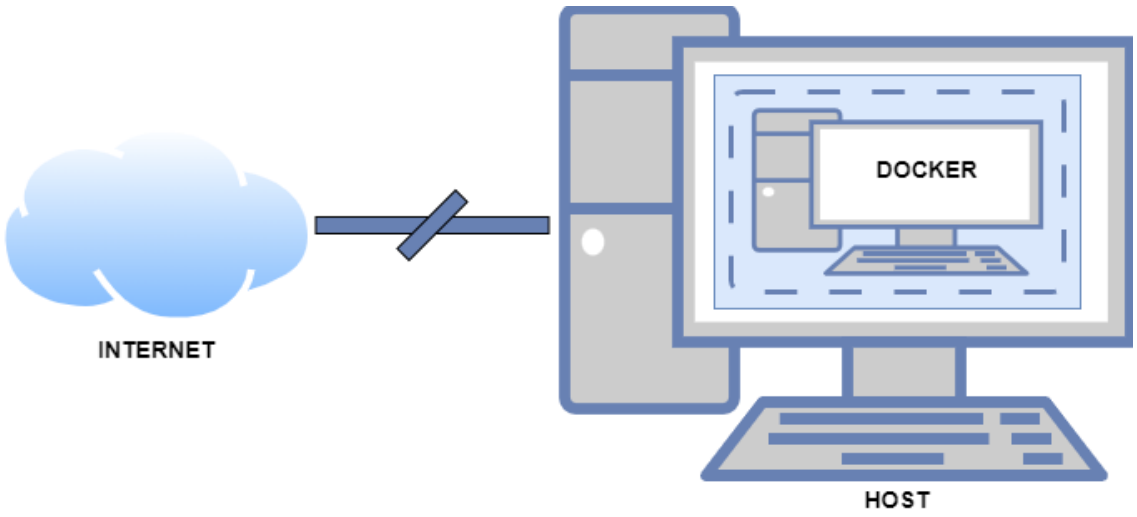

Figura 5 – Ambiente proposto

A imagem docker do Ubuntu será alterada para conter alguns serviços, sendo os principais estes três: back-end sendo desenvolvido em NodeJS, front-end desenvolvido em JavaScript com React e um banco de dados com SQLite. Alguns microsserviços também estarão disponíveis para explorar falhas pontuais e estão indicadas corretamente pelo front-end da aplicação.

5.2 VULNERABILIDADES SELECIONADAS

5.2.1 CWE-23: Relative Path Traversal

Ocorre 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 "../../../../etc/passwd".

5.2.2 CWE-319: Cleartext Transmission of Sensitive Information

Ocasionada quando o sistema realiza a transmissão de dados sensíveis ou críticos em texto plano e permite que usuários não autorizados interceptem o dado.

5.2.3 CWE-328: Use of Weak Hash

Quando o algoritmo de Hash não tem como produto um resumo forte o suficiente, ele fica vulnerável a ataques que encontrem a valor original através de ataques de pré-imagem, 2ª pré-imagem ou ataques de aniversário.

5.2.4 CWE-74: Improper Neutralization of Special Elements in Output Used by a Downstream Component

Vulnerabilidade de injeção baseada na falta de neutralização de caracteres especiais nos valores de entrada que podem alterar a forma de interpretação e execução da aplicação.

5.2.5 CWE-209: Generation of Error Message Containing Sensitive Information

Um designe inseguro de aplicação que permite que o usuário veja informações sobre o ambiente, os usuários e dados relacionados através dos erros gerados na aplicação.

5.2.6 CWE-613: Insufficient Session Expiration

Falha na aplicação que permite acesso por sessões antigas ou apenas com o ID de acesso.

5.2.7 CWE-565: Reliance on Cookies without Validation and Integrity Checking

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.

6 CONSIDERAÇÕES FINAIS

O aprendizado da área de segurança computacional pode ser muito complicado, 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 está sendo 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.

Para este ambiente foram inicialmente escolhidas 7 vulnerabilidades que ainda podem ser encontradas com certa facilidade em sistemas web pelo mundo. Será necessário realizar testes para confirmar se o ambiente fica realmente segregado a fim de que máquinas usadas para testes não sejam vítimas de ataques enquanto usam o ambiente proposto.

6.1 CRONOGRAMA

Etapa	Março	Abril	Maio	Junho	Julho
Desenvolvimento do Back-end em NodeJS	X	X			
Desenvolvilmento do Front-end em React	X	X			
Integração do Front-end com o Back-end		X	X		
Conclusão de escrita da monografia			X		
Defesa de projeto				X	
Ajustes e envio final					X